

# Extensions for Financial Services (XFS)

## XFS4IoT specification

---

### Preview version 0.3

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its Members.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



EUROPEAN COMMITTEE FOR STANDARDIZATION  
COMITÉ EUROPÉEN DE NORMALISATION  
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

## XFS4IOT API Draft 0.0.1

This defines the XFS4IoT API including but not limited to:

- \* Service Discovery
- \* Message Definition
- \* Security

### Documentation

#### Service Discovery

---

##### Service Discovery

###### Introduction

Multiple services can be supplied by multiple vendors. This standard doesn't require coordination between these different organization, or between the service publishers and the service client. It is possible to operate a system with components from multiple hardware vendors, and with third party applications, without the prior knowledge of any party.

This specification covers an environment using WebSockets to communicate between services and applications, either on a single machine or across a network.

This section covers both the process for publishing a service such that it can be discovered, and the discovery process used by the service client.

There is also a clear definition of responsibility for each component, including when there are dependencies between components. There are no shared components required to coordinate the system.

The underlying network can use any protocol that supports WebSockets such as IPv4 or IPv6. Nothing in this document requires any particular underlying protocol.

This document uses CAN, WILL, MUST, SHOULD etc. in the normal ways to distinguish between requirements, recommendations, permissions and possibilities. See <https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/how-to-write-standards.pdf>

Requirements of this spec will be formatted as follows:

This is an example requirement

All other text is commentary used to illustrate the reasoning of the requirements. Where there is any conflict the requirement text always takes priority.

###### Overview

In this standard there are two types of "end-point"; publisher and service. Each end-point, of either type, is published by a single software/hardware vendor. A publisher end-point is used for service discovery, to discover service end-points. A single service end-point can expose multiple "services", where each service typically represents a single piece of hardware. A single machine (or a single IP address) may expose multiple publisher and service end-points from different vendors. A "client" application may consume multiple services from multiple service end-points on the same machine, or across multiple machines.

On startup of the machine, any software services attempt to claim access to individual network ports using the underlying operating system mechanism. Ports are claimed sequentially from a known sequence. Each port becomes an end-point that can publish multiple services from a single vendor.

A client application will attempt to connect to each port on a machine in the known sequence to get a list of all active publisher end-points. For each publisher end-point it then exchanges JSON messages across WebSockets with URLs using a known format to recover a list of services published by that end-point. Once it has a full list of services it can use WebSocket connections to communicate with each service to perform whichever actions are required.

###### Machine Identification

Machines publishing services are identified by URLs. Machines exposing end-points can be identified by an IP address or by a DNS name.

Either the IP address or DNS name for a machine must be known by the client for the client to connect. This would probably be a configuration setting for the application and would need to be known by the organization setting up the application, but this configuration is outside the scope of this document.

###### Network Protocol

TLS security will be used to secure network connections. The only exception will be when the network connection between the client and service can be physically secured because they are both inside the same cabinet. In that case it will be possible to use clear communication without TLS encryption.

The publisher will publish all WebSocket services protected by TLS encryption. This will be identified by the wss:// protocol specifier.

The publisher may publish WebSocket services without TLS encryption, as a clear WebSocket connection, but only if the physical connection between the service and the client is physically protected. It is up to the hardware manufacturer to ensure this physical protection is sufficient. This unsecured connection will be identified by the ws:// protocol specifier.

Where TLS is used, the service will be protected by a mutually trusted server side certificate as part of the TLS protocol. This complete certificate chain must be mutually trusted by the client and service.

Establishing and managing the certificates between the service and the client is outside of the scope of this spec but trust must be in place. This might be achieved using a public third party certificate authority that issues TLS certificates. Alternatively it might be achieved using a bank's own internal CA. It shouldn't depend on a private CA or certificates issued by a vendor, which might limit access to the service.

A WSS connections with invalid certificates will be invalid and will be rejected by both the client and the service.

###### URI Format

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Communication with service publishers and services will be through distinct URLs which will use the following format

wss://machinename:portnumber/xfs4iot/v1.0/servicename

This consists of the following parts:

wss:// or ws://

The protocol id for secure WebSockets. This should be wss:// for secure connections. An insecure ws:// connection can be used when the connection is physically secured, inside an ATM enclosure.

machine name

The identification of the machine publishing end-points. This can be an IP address or DNS name.

port number

The port number discovered through the initial service discovery process

XFS4IoT

A literal string. The inclusion of this part identifies standard XFS4IoT services published on this URI. It allows the possibility of a single vendor publishing standard and non-standard proprietary services on the same port. Any standard service URI will start with this string. Any non-standard service's URI must not start with this string.

v1.0

The version of the XFS4IoT specification being used by this service. This will be updated in future versions of the specification and allows support for multiple versions of the specification on the same machine and end-point.

Note that most future changes to the XFS4IoT specification will be done in a non-breaking, backwards and forwards compatible way. For example, optional fields will be added to JSON messages when required. This means that changes to the version field of the URI will be very rare. It will only be changed if there is a breaking, incompatible change or a fundamental change to the API. Because of this there won't be any need for complex version negotiation between the client and the service. The client will simply attempt to open the version of the API that it supports.

Service Name

This will be included in the URI to allow different services to be identified on the same port. Services will normally match individual devices. The exact service name is discovered during service discovery and is vendor dependent. The format of the service name shouldn't be assumed. The only URI that doesn't include a service name is the service discovery URI.

For example, a service discovery URI might be;

- wss://terminal321.atmnetwork.corporatenet:443/xfs4iot/v1.0
- wss://192.168.21.43:5848/xfs4iot/v1.0

Service URI might be;

- wss://terminal321.atmnetwork.corporatenet:443/xfs4iot/v1.0/maincashdispenser
- wss://192.168.21.43:5848/xfs4iot/v1.0/cardreader

The URI will be case sensitive. The URI will be lower case.

## Service Publishing

Service publishers will negotiate access to resources and publish services using the following process.

### Port Sequence

Services will be published on a sequence of IP ports consisting of two ranges consisting of the port 80 and 443 followed by the ports 5846 to 5856 (inclusive.) Hence the full sequence of ports will be 12 ports as,

80 or 443, 5846, 5847, 5848, ... 5855, 5856

Port 80 will only be used with HTTP/WS. Port 443 will only be used with HTTPS/WSS. All other ports may be used with either or both HTTP/WS and HTTPS/WSS.

Port 80 and 443 are the standard ports for HTTP and HTTPS and have the advantage that they are likely to be open on firewalls. The correct port will be used to match the protocol - 80 for HTTP/WS and 443 for HTTPS/WSS. Other ports are flexible and can be used for either protocol by the Service Publisher.

The port range 5846-5856 is semi-randomly selected in the 'user' range of the port space as defined by ICANN/IANA. This range is currently unassigned by IANA.

### Free End-point Port Discovery

On startup each service publisher must attempt to connect to the first port in the port sequence. It will use the underlying OS and network stack to attempt to bind to this port.

All network access must go through the normal underlying OS mechanism. One service publisher must not block another publisher from accessing the network.

If the underlying OS reports that the port is already in use the service publisher will repeat the same process with the next port in the port sequence. This will be repeated until a port is successfully bound to, or all ports in the sequence have been tried.

If no available port can be found the service publisher will have failed to start. How this failure is handled by the service publisher is undefined.

It's important that a single organisation doesn't use up multiple ports, since this could lead to all the ports being blocked so that other publishers can't get a free port.

Any single organisation will publish all services on a single port, determined dynamically as above.

**Note:** A service publisher will only fail to find a free port if more than 12 different hardware vendors are attempting to publish services from the same machine. This should be unusual.

### Handling Incoming Connections

Once a service publisher has successfully bound to a port it must handle connection attempts. It will accept all connections from any clients without filtering attempts. Security around connections will be handled after a connection has been established.

**Note:** This document does not cover restrictions on connections to services or managing permissions for connections, such as limiting connections to certain machines or sub-nets. This would normally be under the control of the machine deployer and can be controlled through normal firewall settings and network configuration.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Incoming connection attempts will specify a specific URI using the normal WebSocket process. The service publisher will allow connections to valid URLs as defined in this spec and track which URI each connection was made to.

The initial connection will be to the URI `wss://machinename:port/xfs4iot/v1.0`. This connection will then be used to list/discover individual services using the process outlined below ([Service discovery](#)).

### Client

A client application must be able to discover and open a connection to each service that it will use. It does this in two steps; firstly, through publisher end-point discovery, then through service discovery for each service end-point. It will do this through the following process.

#### Publisher End-point Discovery

The client will enumerate end-points by attempting to open a WebSocket connection to the following URL on each port in the port sequence. (See [Port sequence](#).)

```
wss://machinename:port/xfs4iot/v1.0
```

The client will continue to enumerate publisher end-points by repeating for each port number in the port sequence until all ports have been tried.

The client will also start [service discovery](#) on the open connection. There is no requirement for the order of opening ports and discovering services. All ports connections may be created first followed by service discover, or port enumeration and service discovery may continue in parallel.

If the connection attempt to any port fails then the application will attempt error handling for network issues, machine powered off etc. The details of error handling are left up to the application.

#### Service End-point Discovery

*Once a connection has been established between the client and each publisher end-point, the client will discover the services published by sending a service discovery command and receiving events in the usual way.*

The only command sent to the publisher end-point will be "Common.GetService".

```
{
  "header": {
    "type": "command",
    "name": "Common.GetService",
    "requestId": "123"
  },
  "payload": {}
}
```

The end-point will acknowledge the command in the normal way.

The command will be followed by zero or more events. The command will complete with a completion event, in the normal way. Each event, and the completion event will contain the following fields:

```
{
  "header": {
    "type": "completion",
    "name": "Common.GetService",
    "requestId": "123"
  },
  "payload": {
    "vendorName": "<Name of hardware/software vendor>",
    "services": [
      {
        "serviceURI": "wss://machinename:port/xfs4iot/v1.0/<servicename1>",
        {
          "serviceURI": "wss://machinename:port/xfs4iot/v1.0/<servicename2>",
        }
      ]
    }
  }
}
```

The service end-point URI will be returned as serviceURI.

A secure wss:// protocol URI will be returned whenever possible.

An insecure ws:// protocol URI may be returned instead. If an insecure ws:// protocol is used then the hardware vendor will be responsible for ensuring the security of the connection.

Much of the security of the XFS4IoT specification is based around TLS encryption. Using an unencrypted ws:// protocol will have a negative impact on that security, so as far as possible a wss:// should always be used.

If an unencrypted ws:// connection is used then alternative methods should be used to keep the connection secure, perhaps by physically securing the connection.

The Publisher service will send an event to report on every URI. A single event may report on one or more URI. URI will not be repeated between events, so each URI will be reported exactly once.

A publisher service may be designed to send one URI per event, or it may group URI together into a smaller number of events. The publisher should try and send events to report on each URI as soon as each URI is known. It's possible a publisher will know the complete set of URI when they're requested and can send them all at once in one or more events. Alternatively the URI may not be known straight away (such as if an IP address or port is being dynamically allocated.) In that case the publisher service would delay sending events for unknown URI until the full URI is known.

Having each URI reported at most once means that a client can connect to each URI reported in events without having to track which URI have already been connected to. This simplifies the client. Alternatively, a client may wait for the completion event and a full set of URI before attempting to connect. This would be simpler to implement, but might be slower to start up.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

The completion event will contain every URI that the publisher service is aware of.

The publisher service will follow the above process to publish all URI that it's aware of. It will not suppress URI based on device status or service status.

For example, a device might be powered off, in the process of powering on, or powered on but have a hardware fault that makes it impossible to use. In all cases the publisher service will publish the URI anyway. The client can't assume anything about the device based on the URI. It will always need to query the service at the URI for its status to know more.

Events should be sent as soon as a URI is known by the publisher - the event doesn't mean or imply that the URI is currently available or can be connected to - that error handling must be performed by the client (see below.)

**Note:** Even if the publisher service could know that a URI was valid at the time that it sends the event, the client can't know that the URI is still valid when it attempts to use the URI. It could have failed between querying and connecting. So the client has to handle errors, timeouts and retrying when connecting to the URI.

The client may then attempt to open a WebSocket connection to each of the returned URI. The client will handle connection failures and timeouts by repeating the attempts to connect such that the service has a reasonable amount of time to start up.

Each service will endeavor to accept connections as quickly as possible during startup and restarts. Once a connection has been accepted a service may continue to report a 'starting' status until the device is physically started and ready.

Some devices are slow physically to start up, but software should be able to start relatively quickly. So, for example, a cash recycler device might be able to accept a connection within a few seconds of power being applied, but the physical hardware can take several minutes to reset. During this time the service would accept connections but report a 'starting' status.

Each connection will be used to communicate with a single service. The service will then be queried for details about that service, such as the type of service or device that it represents and the messages and interfaces that it supports. (**todo: Querying for service information needs to be documented elsewhere.**)

The connection to the service will be kept open for as long as the service is in use. Details of the service lifetime are covered elsewhere.

The returned URI is a full URI including the machine name and port. It is possible that these values will be different to the service discovery URI - each service may be on a different machine, a different IP address, and a different port. The port is also independent of the discovery port range. It can be any port number.

The service URI values will have the same version number as the service discovery URI version number. Different versions of the API will not be mixed.

If a client wants to open multiple different API version numbers then it should perform service discovery against each of the possible version URI strings.

The client may close the publisher connection once it has completed service discovery, or it may keep the connection open. This will have no effect on the behavior of services.

## Message Definition

### API Definition

In XFS4IoT Services are accessible through a WebSocket Interface. The following specification details the format of message sent to the XFS4IOT Service.

Data sent across the WebSocket stream utilize JSON as a format (<https://www.json.org/>). Each message conforms to one of the following Message Types. Message Types are documented in the following table.

Message Type	Direction	Description
Command	Client to Service	Message sent to the XFS4IOT Service to perform a Command
Response	Service to Client	Message from the XFS4IOT Service indicating if the Command is valid and queued.
Event	Service to Client	Intermediate message from the XFS4IOT Service indicating progress of the Command.
Completion	Service to Client	Final message from the XFS4IOT Service indicating the Command is complete.
Unsolicited	Service to Client	Message from the XFS4IOT Service unrelated to a Command.

All the message types follow the same JSON structure conforming of a mandatory header and payload.

- **header** : containing attributes that are common across all Message Types to allow the payload to be efficiently parsed
- **payload** : containing information that is specific to the Message Type and action.

Header and Payload are the only two attributes defined at the top level of the JSON structure as the example illustrated below.

```
{  
  "header": {  
  },  
  "payload": {  
  }  
}
```

### Header Definition

Headers are consistent across all XFS4IOT Message Types. The following table details the header content. All attributes are mandatory.

Attribute	Type	Description
Name		
requestId	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. For example "12345" or "b34800d0-9dd2-4d50-89ea-92d1b13df54b"
type	string	The message type. The is must be either "command", "response", "event", "completion" or "unsolicited"
name	string	The message name, for example "CardReader.Status"

The following example illustrates the header for a CardReader.Status command Message.

```
{  
  "header": {  
    "type": "command",  
  },  
  "payload": {  
  }  
}
```

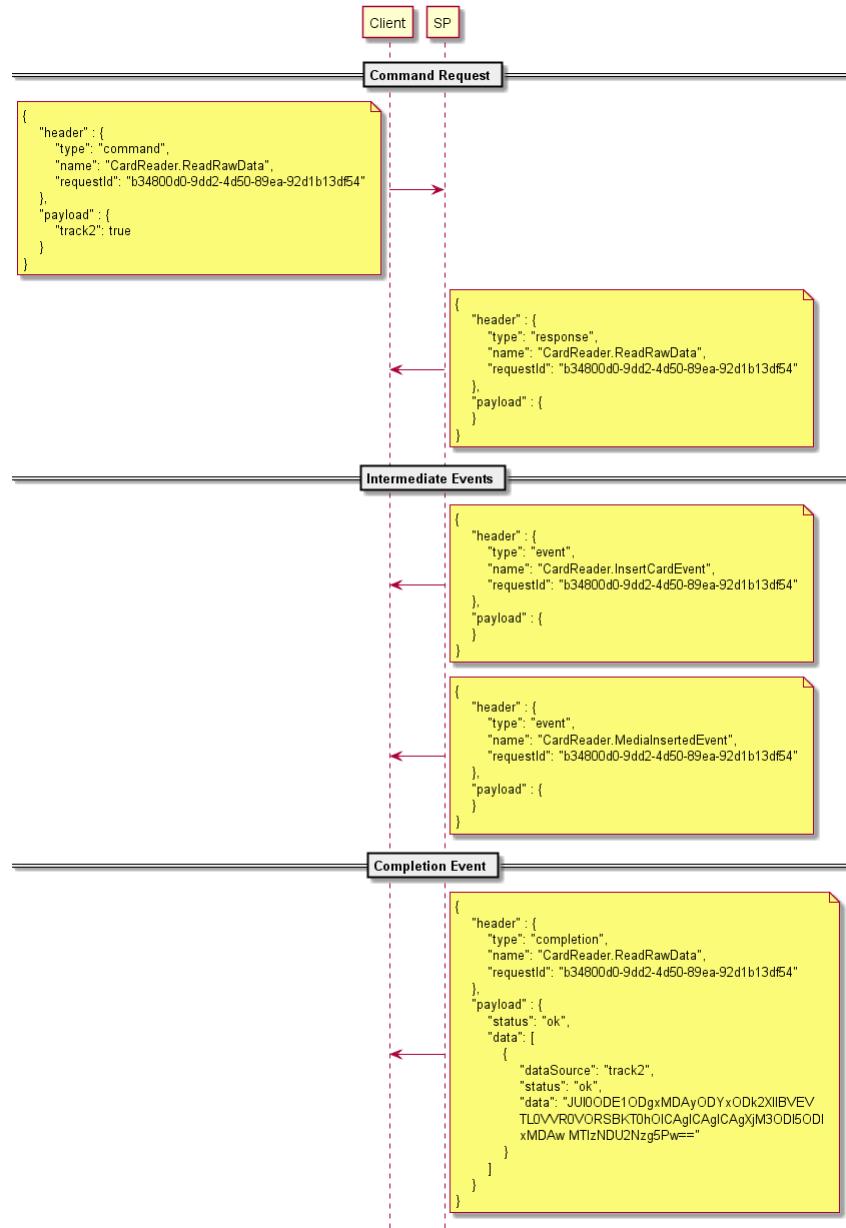
## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
        "name": "CardReader.Status",
        "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54"
    },
    "payload": {
    }
}
```

### Payload Definition

The XFS4IoT class specifications detail the Payload content for the class Command, Event, Completion and Unsolicited Messages.

### Example Message Flow



### End to End security overview

#### ## Overview

A key priority for XFS4IoT is to improve security of the whole environment where XFS is used. This means securing not only the interface between the service and the hardware, or the interface between the client and the service, but providing security all the way from one end of an operation to the other.

For example, during a cash dispense operation the transaction will first be authorised by an authorising host which represents the owner of the cash in the device. That host will communicate through various other systems to the client application, the client application will communicate with the XFS4IoT service and the service will finally communicate with the hardware. Any part of that process is vulnerable to an attack which could lead to the wrong amount of cash being dispensed. XFS4IoT has been designed to block attacks at any point between the authorising host and the dispenser hardware - it's truly end to end.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Both data 'integrity' like this, and 'confidentiality' for things like card data, are important. XFS4IoT focuses on integrity since confidentiality will be covered by TLS encryption of the network messages.

### Tokens

XFS4IoT implements end to end security using strong cryptography, such as HMAC values. Secret keys are shared between endpoints using TR34 and TR31. The use of public key RKL ensures that only the correct endpoints have access to the end to end transactions.

The data relevant to the end to end transactions is separated out from the normal clear text properties passed in the JSON message formats, and is included in different "token" properties in the messages. Each token property is defined in the relevant service class documentation. For example, there is a "dispense token" for cash dispense actions and a "present status" token to protect the information about the last presented cash operation.

Tokens are encoded as a simple string which contains all of the information needed for that token, and also the information needed to keep it secure, such as the HMAC value. Keeping all of the information together in this way ensures that a single HMAC can protect all of the data, and there's no possibility that part of the data could be changed independent of other parts.

Keeping the token as a simple string means that it's easy to handle for low-power hardware. For example, the token may need to be checked by embedded firmware, even if the service is running on a front end machine. To be fully end to end it must be checked on the hardware, so the format is kept simple.

Like all XFS4IoT data any binary or hex values, such as the HMAC value, are encoded as a base64 string.

The general format of a token is a string with a set of key=value pairs all comma seperated and UTC8 encoded. The first pair will give a random number. The last value will be an HMAC. The encoding is important so that the HMAC is consistent.

```
RANDOM=<Random>, <KEY>=<value>, HMACSHA256=<HMAC>
```

- Random number is the number initially exchanged between the end points and is used to ensure that each Token is different and to avoid replay attacks. The random number will always come at the start of the token (so that there isn't much constant leading data - this is important for security.)
- The KEY=value pairs define what the token is used for. For example, a cash dispenser might have "DISPENSE1" or "PRESENTSTATUS" keys in different Tokens. This ensures that tokens with different uses can't be reused by an attacker. "value" is the actually value of the data being protected and will be different for each operation.
- There can be multiple key=value pairs, seperated by commas. Keys can appear in any order. Key names are always upper case.
- Tokens will not contain any extra whitespace.
- The HMAC is the BASE64 encoded HMAC of all the preceeding data up to and including the last comma. The HMAC is always the last value. This makes it easy to calculate the HMAC since it can be calculated over other data, converted to BASE64, then appended to the string. The HMAC will use SHA256 as the algorithm.

For example, a dispense token might be:

```
RANDOM=254611E63B2531576314E86527338D61,  
DISPENSE1=50.00EUR,  
HMACSHA256=wDohmRIKUqm/Y9hEc65B3YYkA1mAmeUDOnSmAjGIQK0=
```

The value used to calculate the HMAC is

```
RANDOM=254611E63B2531576314E86527338D61,DISPENSE1=50.00EUR,HMACSHA256=
```

A HMAC for this data, with SHA256 and a key of 112233445566778899AABCCDDEFF, is

```
C033A199120A52A9BF63D84473AE41DD862402598099E5033A74A60231840AD
```

which BASE64 encoded is.

```
wDohmRIKUqm/Y9hEc65B3YYkA1mAmeUDOnSmAjGIQK0=
```

This would be included in a dispense command message as:

```
{  
    "header": {  
        "type": "command",  
        "name": "Dispenser.Dispense",  
        "requestId": 456  
    },  
    "payload": {  
        ...  
        "dispenseToken": "RANDOM=254611E63B2531576314E86527338D61,  
DISPENSE1=50.00EUR,  
HMACSHA256=wDohmRIKUqm/Y9hEc65B3YYkA1mAmeUDOnSmAjGIQK0="  
        ...  
    }  
}
```

The standard set of keys and values is defined in the relevant specification for each token type. Including which keys are required.

It's also permitted to include custom key values in a token. For example, a hardware dependent error code might be included. Unknown keys will be included in the HMAC calculation, but otherwise ignored. There must not be a dependency on custom keys. Care should be taken to avoid name clashes between keys, maybe by using a vendors name in the key name.

The total token length will be limited to 1024 BYTES to avoid the risk of buffer overflows. Any token longer than this will be treated as invalid data.

### Key management

To ensure strong security symmetric keys are shared between different endpoints. Ideally this will be done using public key encryption, as defined in TR34. However, in some cases it may be necessary to pre-load keys into hardware in some other secure way, for example in legacy hardware that can't support TR34. The security of the whole system is only as good as the security of these keys so it's vital that this is done in as secure a way as possible.

The keys that are loaded are covered by the TR31 specification, which defines both key data and the key details such the intended use. This includes the algorithm that the key is intended to be used with. Since the key details defines the algorithm there is no need for algorithms to be negotiated in any other way - the two end points will effectively agree the algorithm to use by loading the relevant keys.

For simplicity, communication in each direction will be handled separately. So, for example, there will be a separate key for tokens passed in each direction. (Also there will be a separate random number in each direction. See below.) The naming convention will be to use "Command" for values relating to tokens passing from the client to the service, and "Response" for tokens passing from the service to the client. Hence there is a CommandKey and a separate ResponseKey.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

The details of the key management are covered by the shared key management service class. Any service that implements end to end security will implement this service class as part of its interface.

### Unique messages and replay attacks

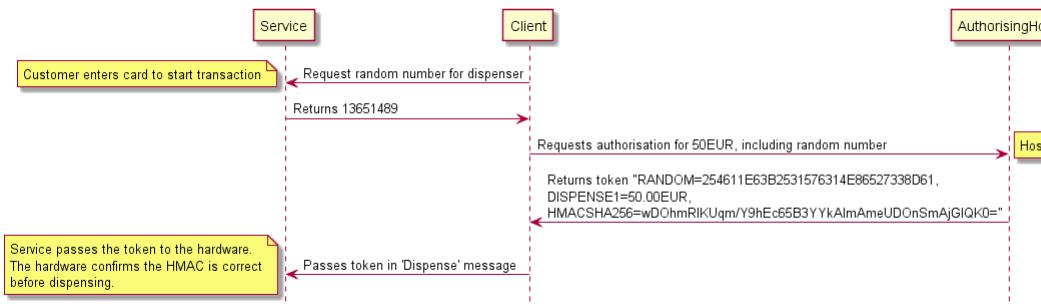
To avoid 'replay attacks', where an attacker reuses an old message to replace a new one, it's important that all individual tokens are unique and the same data is not used multiple times. For example, it's common to have a dispense token for "10EUR" so that value (and its HMAC) will be the same for many transactions and could be reused by an attacker. To avoid this a random number is included in each token and the random number will be different for each transaction. This guarantees uniqueness.

There will be a different random number for tokens passed in each direction. That is, there will be a CommandRandom and ResponseRandom number. Each random number must be generated and checked at the same end of the communication, so the CommandRandom must be generated and checked by the device hardware. The ResponseRandom number must be generated and checked by the client. (Ideally in the host HSM.)

To fully avoid replay attacks the random number must be agreed between the endpoints before it's used. An extra command on the service class is called by the client to fetch a new Command Random number. This random number is then remembered by both end points and included in each command token. Similarly, a Response Random Number must be generated by the client/host, passed to the service, and included in all Response Tokens. It should be possible to add the Response Random number to existing messages, such as a dispense command message for the cash dispenser.

### Example of classic dispense

The following example shows how end to end security is used to protect a cash dispense operation during a classic ATM transaction.



## Commands

### Common.GetServices

#### Description

Command sent to the service discovery port to find the details of the service exposed by this publisher

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
requestId <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
vendorname	string		Freeform string naming the hardware vendor
services	array		
services.serviceURI <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The URI which can be used to contact this individual service

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "vendorname": "string",
    "services": [
      {
        "serviceURI": "wss://ATM1:123/xfs4iot/v1.0/CardReader"
      }
    ]
  }
}
```

### Event Messages

- [Common.ServiceDetailEvent](#)

## Unsolicited Events

### Events

#### Common.ServiceDetailEvent

##### Description

Details of some services published by this endpoint

##### Message Header

Name	Type	Default	Description
requestId <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The original message name, for example "CardReader.Status"

##### Message Payload

Name	Type	Default	Description
vendorname	string		Freeform string naming the hardware vendor
services	array		
services.serviceURI <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The URI which can be used to contact this individual service

### Example Message (generated)

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
[  
  "headers": {  
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
    "type": "command",  
    "name": "string"  
  },  
  "payload": {  
    "vendorname": "string",  
    "services": [  
      {  
        "serviceURI": "wss://ATM1:123/xfs4iot/v1.0/CardReader"  
      }  
    ]  
  }  
]
```

**XFS4IoT**  
**All rights**

XFS4IOT Sample Messaging for Card Readers Draft 0.0.1

This service allows for the operation of the following categories of units:

- Motor driven card reader/writer
  - Swipe card reader (writing facilities only partially included)
  - Dip card reader
  - Contactless chip card readers
  - Permanent chip card readers (each chip is accessed through a unique service)

Some motor driven card reader/writers have parking stations inside and can place identification cards there. Once a card is in its parking station another card can be accepted by the card reader. Cards may only be moved out of a parking station if there is no other card present in the media read/write position, the chip I/O position, the transport, or the entry/exit slot.

The following tracks/chips and the corresponding international standards are taken into account in this document:

- Track 1 - ISO 7811
  - Track 2 - ISO 7811
  - Track 3 - ISO 7811 / ISO 4909
  - Cash Transfer Card Track 1 - (JIS I: 8 bits/char) Japan
  - Cash Transfer Card Track 3 - (JIS I: 8 bits/char) Japan
  - Front Track 1 - (JIS II) Japan
  - Watermark - Sweden
  - Chip (contacted) - ISO 7816
  - Chip (contactless) - ISO 10536, ISO 14443 and ISO 1809

## Documentation

## Form Description

This section describes the forms mechanism used to define the tracks to be read or written. Forms are contained in a single file, with one section for each defined form. The name of each section is the form name parameter in the [CardReader.ReadTrack](#) and [CardReader.WriteTrack](#) commands.

The way to specify the location of a form file is vendor dependent.

The read form defines which tracks should be read in the [CardReader.ReadTrack](#) command and what the response should be to a read failure. The read form can also be used to define logical track data, i.e. fields like "account number", "issuer identifier", and their position within the physical track data. For example, the output parameter of the [CardReader.ReadTrack](#) command with input parameter `lpstrFormName = READTRACK3GERMAN` could look like (see example 1 below):

The write form defines which track is to be written, the logical track data that is handed over in the [CardReader.WriteTrack](#) command, and how the write data is to be converted to the physical data to be written.

Reserved Keywords/Operands	Meaning
[]	Form Name
TRACK1	Keyword to identify track 1.
TRACK2	Keyword to identify track 2.
TRACK3	Keyword to identify track 3.
TRACK1_JIS1	Keyword to identify JIS I track 1.
TRACK3_JIS1	Keyword to identify JIS I track 3.
FRONTTRACK1	Keyword to identify front track 1 (in some countries this track is known as JIS II track).
FIELDSEPT1	Value of field separator of track 1.
FIELDSEPT2	Value of field separator of track 2.
FIELDSEPT3	Value of field separator of track 3.
FIELDSEPTFRONT1	Value of field separator of front track 1.
FIELDSEPT1_JIS1	Value of field separator of JIS I track 1.
FIELDSEPT3_JIS1	Value of field separator of JIS I track 3.
READ	Description of read action; the track identifier keywords are processed left to right.
WRITE	Description of write action.
ALL	Read or write the complete track.
SECURE	Do the security check via the security module (CIM86 or MM). This check is done on Track 3 only.
&	Read/write all tracks specified, abort reading on read failure.
	Read/write at least one of the tracks specified, continue reading on read failure.
FIELDSEPPOSn	Position of the nth occurrence of field separator on track. FIELDSEPPOS0 specifies the beginning of the data.
,	Separator in a list of logical fields.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Reserved Keywords/Operands

	Meaning
DEFAULT	String for default substitution of track data to be written, that is not defined explicitly by the form fields. DEFAULT also allows an application to input fewer fields than those defined by the form.
?	Reserved value for DEFAULT keyword: substitute track data to write with its value read before.
ENDTRACK	Represents the end of the data. It is used to identify fields positioned after the last field separator.

### Notes

The & and | operands may be combined in a single READ statement; for example:

- read track3 or track2, trying track3 first

```
READ= TRACK3 | TRACK2
```

- read track 3 and at least one of track2 or track1:

```
READ = TRACK3 & (TRACK2 | TRACK1)
```

or:

```
READ = TRACK2 | TRACK1 & TRACK3
```

The keywords FIELDSEPP0 and ENDTRACK are used as follows:

- read the first 2 bytes of a track:

```
FIRST = FIELDSEPP0 + 1, FIELDSEPP0 + 2
```

- read the last 2 bytes of a track:

```
LAST = ENDTRACK -- 2, ENDTRACK -- 1
```

Use of field separators in track layouts is to replace optional fields and terminate variable length fields.

Write forms are designed for updating specific fields without altering the position of the field separators.

The application may alter the position of the field separators by rewriting the card tracks (ALL option or DEFAULT option with default track data).

It is valid to define a field that spans another field separator, e.g. FIELDSEPP0+1, FIELDSEPP0+2 is valid as is FIELDSEPP0+3-4, FIELDSEPP0+1 where a field separator (e.g. FIELDSEPP0) lies within this range on the data read from the card. During a read track the field separator is returned within the track data. During a write track the application must ensure the correct number of field separators at the correct location with the correct spacing is included in the data, otherwise a *dataSyntax* error will be returned.

### Example 1 Reading tracks:

```
[READTRACK3GERMAN]
FIELDSEPT1= /* field separator of track 1 */
FIELDSEPT2= /* field separator of track 2 */
FIELDSEPT3= /* field separator of track 3 */
READ= TRACK3 & TRACK1 & TRACK2 /* all tracks must be read */

/* read logical fields as defined below; also check the security */
TRACK3= MII, COUNTRY, ISSUERID, ACCOUNT, LUHNT3, EXPIRATION, SECURE

MII= FIELDSEPP0 + 3, FIELDSEPP0 + 4
ISSUERID= FIELDSEPP0 + 5, FIELDSEPP0 + 1
ACCOUNT= FIELDSEPP0 + 1, FIELDSEPP0 + 2
LUHNT3= FIELDSEPP0 + 1, FIELDSEPP0 + 1

TRACK2= ALL /* return track 2 complete, don't return logical
fields*/
TRACK1= ALL /* return track 1 complete, don't return logical
fields*/
```

All tracks must be read ('READ'), that is, the read fails if an error occurs on reading any one of the tracks (the '&' operand). The field "major industry identifier" ('MII') is located after the first field separator ('FIELDSEPP0') and its length is two bytes. The "issuer identifier" field ('ISSUERID') is located after the MII field, with a length of eight bytes. The next field, "account number" ('ACCOUNT') is variable length; it ends before the luhn digit field ('LUHNT3') that is the last digit in front of the second field separator ('FIELDSEPP0').

### Example 2 Write a track:

```
[WRITETRACK3]
FIELDSEPT3= 

DEFAULT= ? /* fields not specified in the write form are to be left
unchanged, i.e. read and the same data written back to them */

WRITE= TRACK3

TRACK3= RETRYCOUNT, DATE

RETRYCOUNT= FIELDSEPP0 + 22, FIELDSEPP0 + 22
DATE= FIELDSEPP0 + 1, FIELDSEPP0 + 4
```

Track 3 is to be written. In the example only the retry counter and the date of the last transaction are updated, the other fields are unchanged.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

A sample of input data to be used with this form is as follows:

```
RETRYCOUNT=3\\0DATE=3132\\0\\0
```

**Example 3** Write a track:

```
[WITETRACK3ALL]
```

```
WRITE= TRACK3
```

```
TRACK3= ALL
```

Track 3 is to be written. By specifying ALL, the data passed in the [CardReader.WriteTrack](#) command is written to the physical track without formatting.

A sample of input data to be used with this form is as follows:

```
ALL=123456789123\\0\\0
```

**Example 4** Reading tracks:

```
[READTRACK2ANDFRONTTRACK1]
```

```
READ= TRACK2&FRONTTRACK1 /* track 2 and front track 1 must be read */
```

```
TRACK2= ALL
```

```
FRONTTRACK1= ALL
```

Track 2 and Front track 1 are to be read by the [CardReader.ReadTrack](#) command. By specifying '&', reading is aborted if either track fails to read. By specifying 'ALL' the physical track is read to the output data without field level formatting.

A sample of output data produced with this form is as follows:

```
TRACK2:ALL=123456789123\\0\\0FRONTTRACK1:ALL=123456789123\\0\\0\\0
```

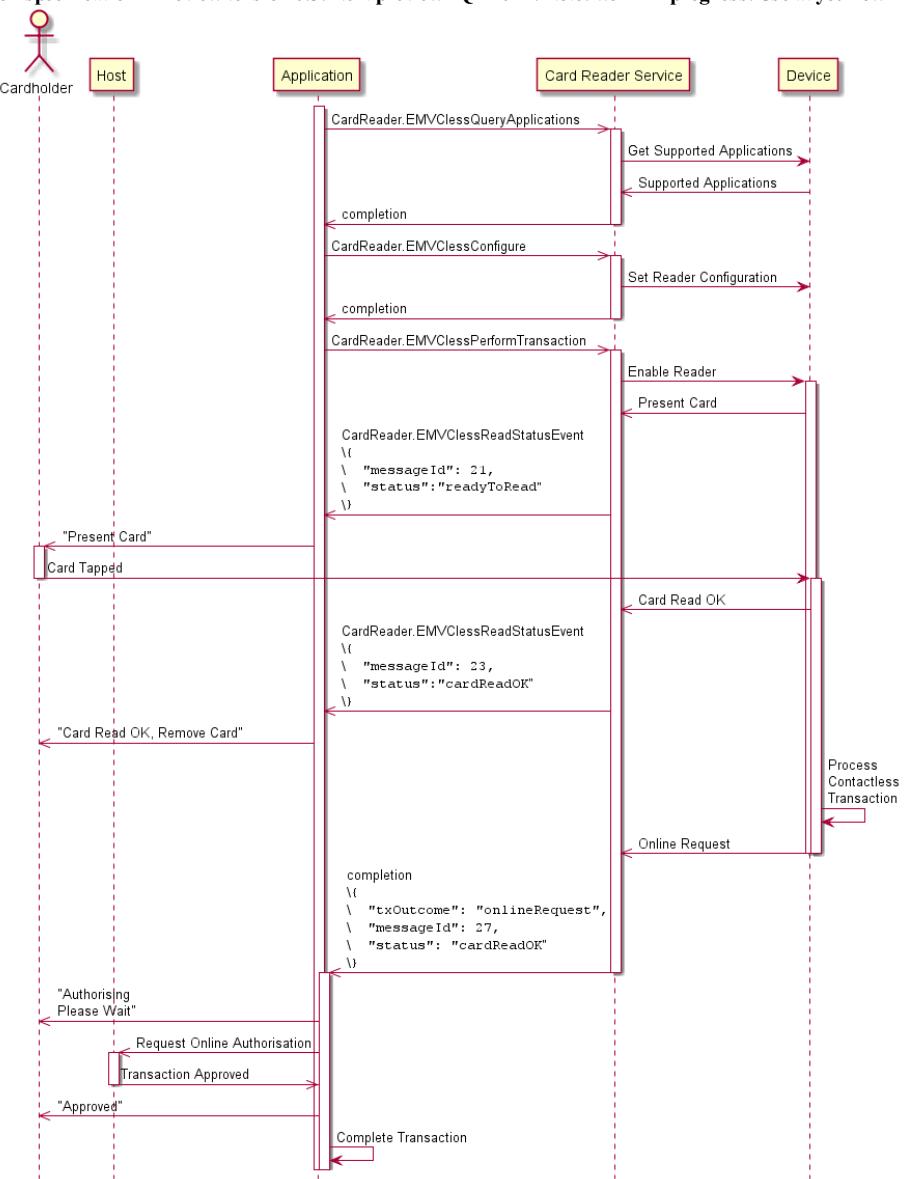
## Intelligent Contactless Sequence Diagrams

---

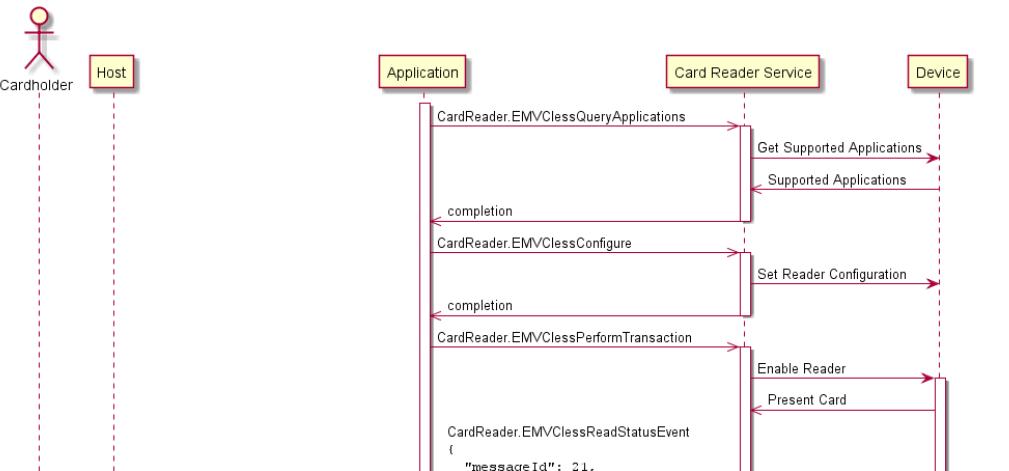
This section illustrates the sequence diagrams of EMV-like intelligent contactless transactions.

### Single Tap Transaction Without Issuer Update Processing

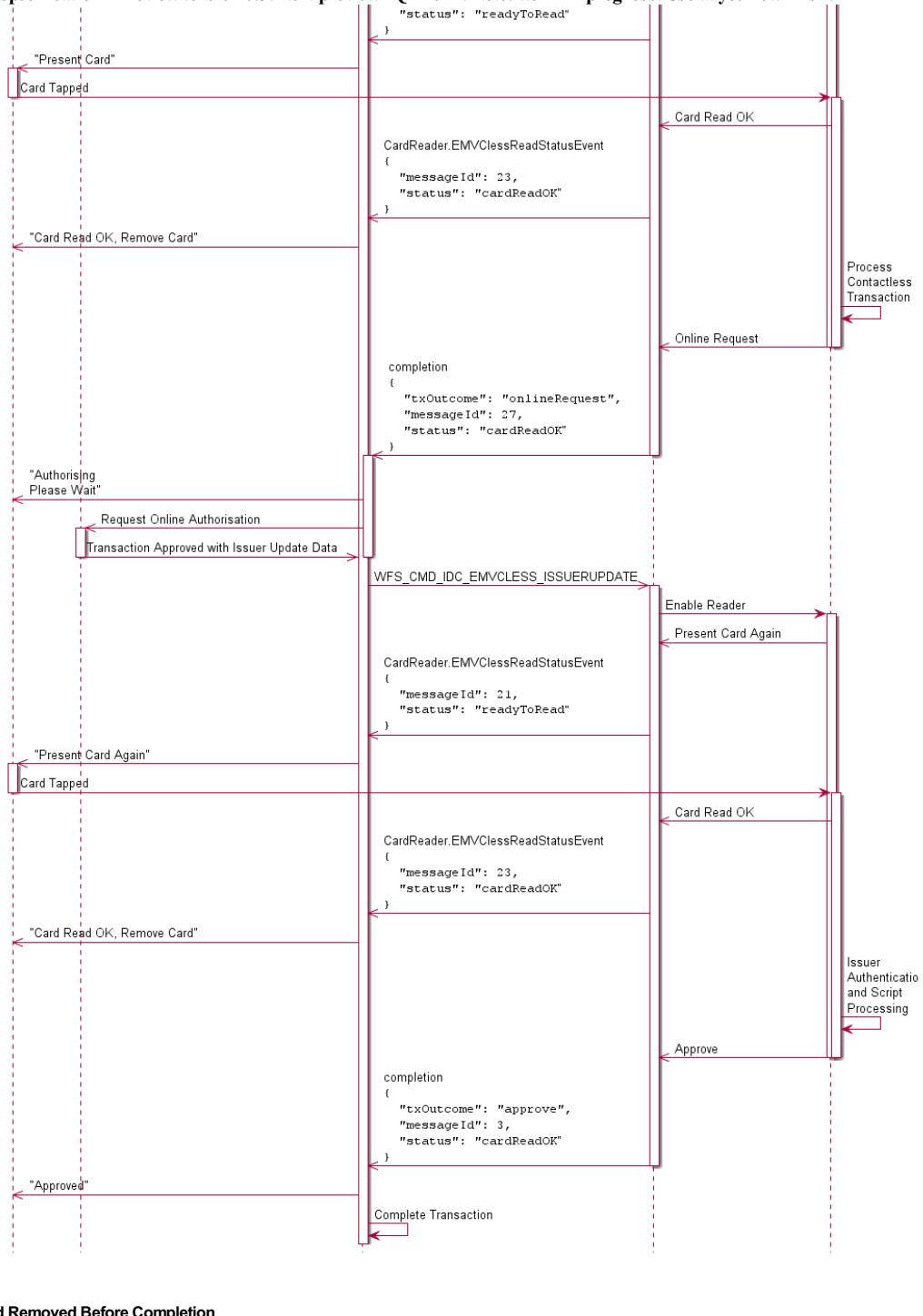
---

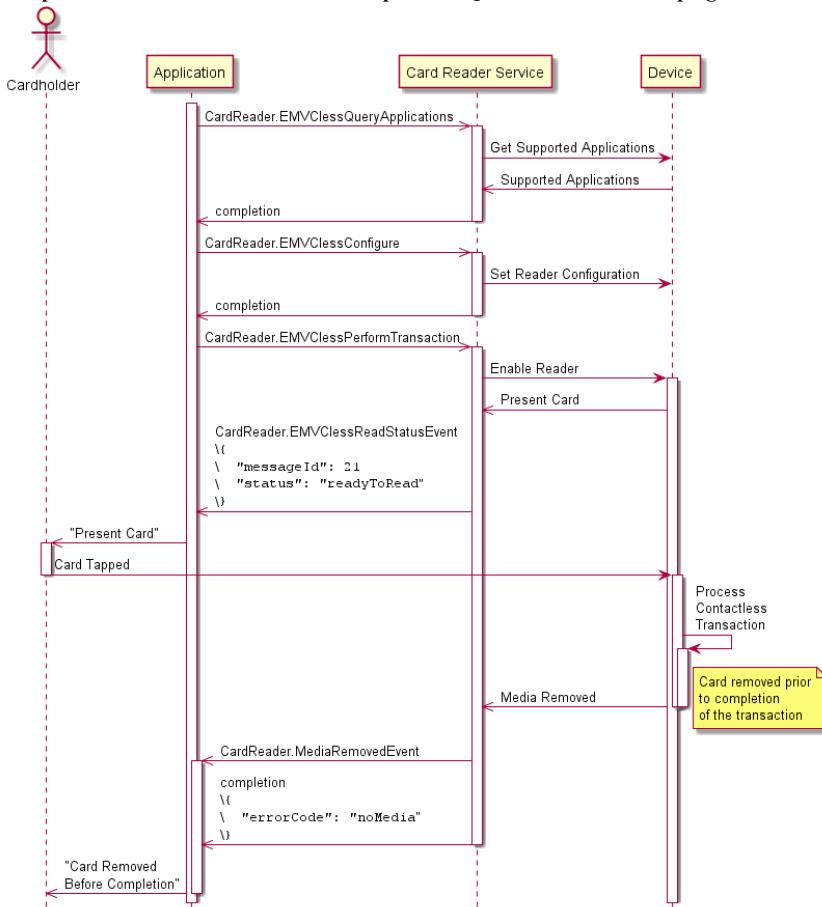


#### Double Tap Transaction With Issuer Update Processing



XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.





## Commands

### CardReader.FormList

#### Description

This command is used to retrieve the list of forms available on the device.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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)

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
forms	array		The available forms

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "forms": [
      "string"
    ]
  }
}
```

### Event Messages

## CardReader.QueryForm

#### Description

This command is used to retrieve details of the definition of a specified form.

#### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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.
formName	string		The form name for which to retrieve details.

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
formName	string		The form name.
fieldSeparatorTrack1	string		The value of the field separator of track 1.
fieldSeparatorTrack2	string		The value of the field separator of track 2.
fieldSeparatorTrack3	string		The value of the field separator of track 3.
fieldSeparatorFrontTrack1	string		The value of the field separator of front track 1.
fieldSeparatorJISITrack1	string		The value of the field separator of JIS I track 1.
fieldSeparatorJISITrack3	string		The value of the field separator of JIS I track 3.
action	string		The form action as one of the following: <b>read</b>   The form reads the card. <b>write</b>   The form writes the card.
actionDescription	string		The description of the READ or WRITE action.
secure	boolean		Whether or not to do a security check.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
track1Fields	array		The field names for track 1.
track2Fields	array		The field names for track 2.
track3Fields	array		The field names for track 3.
jisItrack1Fields	array		The field names for JIS I track 1.
jisItrack3Fields	array		The field names for JIS I track 3.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "formName": "string",
    "fieldSeparatorTrack1": "s",
    "fieldSeparatorTrack2": "s",
    "fieldSeparatorTrack3": "s",
    "fieldSeparatorFrontTrack1": "s",
    "fieldSeparatorJISITrack1": "s",
    "fieldSeparatorJISITrack3": "s",
    "action": "read",
    "actionDescription": "string",
    "secure": true,
    "track1Fields": [
      "string"
    ],
    "track2Fields": [
      "string"
    ],
    "track3Fields": [
      "string"
    ],
    "jisItrack1Fields": [
      "string"
    ],
    "jisItrack3Fields": [
      "string"
    ]
  }
}
```

### Event Messages

## CardReader.QueryIFMIdentifier

### Description

This command is used to retrieve the complete list of registration authority Interface Module (IFM) identifiers. The primary registration authority is EMVCo but other organizations are also supported for historical or local country requirements.

New registration authorities may be added in the future so applications should be able to handle the return of new (as yet undefined) IFM identifiers.

### Command Message

#### Message Header

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

#### Message Payload

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
ifmAuthority	string		Specifies the IFM authority that issued the IFM identifier: <b>emv</b>   The Level 1 Type Approval IFM identifier assigned by EMVCo. <b>europay</b>   The Level 1 Type Approval IFM identifier assigned by Europay. <b>visa</b>   The Level 1 Type Approval IFM identifier assigned by VISA. <b>gicb</b>   The IFM identifier assigned by GIE Cartes Bancaires.
ifmIdentifier	string		The IFM Identifier of the chip card reader (or IFM) as assigned by the specified authority.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "ifmAuthority": "emv",
    "ifmIdentifier": "string"
  }
}
```

### Event Messages

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
**CardReader.EMVClassQueryApplications**

---

**Description**

This command is used to retrieve the supported payment system applications available within an intelligent contactless card unit. The payment system application can either be identified by an AID or by the AID in combination with a Kernel Identifier. The Kernel Identifier has been introduced by the EMVCo specifications; see Reference [3].

**Command Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
appData	array		An array of application data objects which specifies a supported identifier (AID) and associated Kernel Identifier.
appData.aid	string		Contains the Base64 encoded payment system application identifier (AID) supported by the intelligent contactless card unit.
appData.kernelIdentifier	string		Contains the Base64 encoded Kernel Identifier associated with the aid. This data may be empty if the reader does not support Kernel Identifiers for example in the case of legacy approved contactless readers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "appData": [
      {
        "aid": "string",
        "kernelIdentifier": "string"
      }
    ]
  }
}
```

### Event Messages

---

## CardReader.ReadTrack

---

### Description

For motor driven card readers, the ID card unit checks whether a card has been inserted. If so, the tracks are read immediately as described in the form specified by the *formName* parameter.

If no card has been inserted, and for all other categories of devices, the ID card unit waits for the application specified timeout for a card to be either inserted or pulled through. Again the next step is reading the tracks specified in the form (see Section XXX, Form Definition, for a more detailed description of the forms mechanism). When the SECURE tag is specified in the associated form, the results of a security check via a security module (i.e. MM, CIM86) are specified and added to the track data.

A [CardReader.InsertCardEvent](#) will be generated when there is no card in the card reader and the device is ready to accept a card. If the security check fails however this should not stop valid data being returned. The error WFS\_ERR\_IDC\_SECURITYFAIL will be returned if the form specifies only security data to be read and the security check could not be executed, in all other cases WFS\_SUCCESS will be returned with the security field of the output parameter set to the relevant value including WFS\_IDC\_SEC\_HWERRO.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <span style="border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="border: 1px solid black; padding: 2px;">(Required)</span>	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.
formName	string		The name of the form that defines the behavior for the reading of tracks.

#### Example Message (generated)

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

### Completion Message

This event notifies the completion of the command and if successful includes the requested data.

#### Message Header

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
trackData	string		The data read successfully from the selected tracks (and value of security module if available).

### Example Message (generated)

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

### Event Messages

- [CardReader.InsertCardEvent](#)
- [CardReader.MediaInsertedEvent](#)
- [CardReader.MediaRemovedEvent](#)
- [CardReader.InvalidTrackDataEvent](#)
- [CardReader.InvalidMediaEvent](#)
- [CardReader.TrackDetectedEvent](#)

## CardReader.WriteTrack

### Description

For motor-driven card readers, the ID card unit checks whether a card has been inserted. If so, the data is written to the track as described in the form specified by the *formName* parameter, and the other parameters.

If no card has been inserted, and for all other categories of devices, the ID card unit waits for the application specified timeout for a card to be either inserted or pulled through. The next step is writing the data defined by the form and the parameters to the respective track (see Section XXX, Form Definition, for a more detailed description of the forms mechanism).

This procedure is followed by data verification.

A [CardReader.InsertCardEvent](#) will be generated when there is no card in the card reader and the device is ready to accept a card.

If power fails during a write the outcome of the operation will be vendor specific, there is no guarantee that the write will have succeeded.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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.
formName	string		The name of the form to be used.
trackData			The track data to be written.
writeMethod	string	auto	<p>Indicates whether a low coercivity or high coercivity magnetic stripe is being written as one of the following:</p> <p><b>loco</b>&lt;br&gt; Low coercivity magnetic stripe is being written.</p> <p><b>hico</b>&lt;br&gt; High coercivity magnetic stripe is being written.</p> <p><b>auto</b>&lt;br&gt; Service Provider will determine whether low or high coercivity stripe is to be written.</p>

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

- [CardReader.InsertCardEvent](#)
- [CardReader.MediaInsertedEvent](#)

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

- [CardReader.MediaRemovedEvent](#)
- [CardReader.InvalidTrackDataEvent](#)
- [CardReader.InvalidMediaEvent](#)

### CardReader.EjectCard

#### Description

This command is only applicable to motor driven card readers and latched dip card readers.

For motorized card readers the default operation is that the card is driven to the exit slot from where the user can remove it. The card remains in position for withdrawal until either it is taken or another command is issued that moves the card.

For latched dip readers, this command causes the card to be unlatched (if not already unlatched), enabling removal.

After successful completion of this command, a [CardReader.MediaRemovedEvent](#) is generated to inform the application when the card is taken.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <small>(Required)</small>	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 <small>(Required)</small>	string		The message type, either command, response, event or completion.
name <small>(Required)</small>	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.
ejectPosition	string	exitPosition	<b>exitPosition</b>   The card will be transferred to the exit slot from where the user can remove it. In the case of a latched dip the card will be unlatched, enabling removal.  <b>transportPosition</b>   The card will be transferred to the transport just behind the exit slot. If a card is already at this position then ok will be returned. Another EjectCard command is required with the wEjectPosition set to exitPosition in order to present the card to the user for removal.

##### Example Message (generated)

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

#### Completion Message

##### Message Header

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

- [CardReader.MediaRemovedEvent](#)
- [CardReader.RetainBinThresholdEvent](#)

## CardReader.RetainCard

**Description**

The card is removed from its present position (card inserted into device, card entering, unknown position) and stored in the retain bin; applicable to motor-driven card readers only. The ID card unit sends a [CardReader.RetainBinThresholdEvent](#) if the storage capacity of the retain bin is reached. If the storage capacity has already been reached, and the command cannot be executed, an error is returned and the card remains in its present position.

**Command Message**

**Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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)**

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
count	integer		Total number of ID cards retained up to and including this operation, since the last ResetCount command was executed.
position	string		Position of card; only relevant if card could not be retained. Possible positions:  <b>unknown</b>   The position of the card cannot be determined with the device in its current state.  <b>present</b>   The card is present in the reader.  <b>entering</b>   The card is in the entering position (shutter).

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "count": 0,
    "position": "unknown"
  }
}
```

### Event Messages

- [CardReader.MediaRemovedEvent](#)
- [CardReader.RetainBinThresholdEvent](#)
- [CardReader.MediaRetainedEvent](#)

---

## CardReader.ResetCount

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Description

This function resets the present value for number of cards retained to zero. The function is possible for motor-driven card readers only.

The number of cards retained is controlled by the service and can be requested before resetting via [CardReader.Status](#).

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

- [CardReader.RetainBinThresholdEvent](#)

## CardReader.SetKey

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Description

This command is used for setting the DES key that is necessary for operating a CIM86 module. The command must be executed before the first read command is issued to the card reader.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
keyValue	string		Contains the Base64 encoded payment containing the CIM86 DES key. This key is supplied by the vendor of the CIM86 module.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

## Event Messages

---

### CardReader.ReadRawData

---

#### Description

For motor driven card readers, the card unit checks whether a card has been inserted. If so, all specified tracks are read immediately. If reading the chip is requested, the chip will be contacted and reset and the ATR (Answer To Reset) data will be read. When this command completes the chip will be in contacted position. This command can also be used for an explicit cold reset of a previously contacted chip.

This command should only be used for user cards and should not be used for permanently connected chips.

If no card has been inserted, and for all other categories of card readers, the card unit waits for the period of time specified in the call for a card to be either inserted or pulled through. The next step is trying to read all tracks specified.

The [CardReader.InsertCardEvent](#) will be generated when there is no card in the card reader and the device is ready to accept a card. In addition to that, a security check via a security module (i.e. MM, CIM86) can be requested. If the security check fails however this should not stop valid data being returned. The response `securityFail` will be returned if the command specifies only security data to be read and the security check could not be executed, in all other cases `ok` will be returned with the data field of the output parameter set to the relevant value including `hardwareError`.

For non-motorized Card Readers which read track data on card exit, the `invalidData` error code is returned when a call to is made to read both track data and chip data.

If the card unit is a latched dip unit then the device will latch the card when the chip card will be read, i.e. `chip` is specified (see below). The card will remain latched until a call to `ejectCard` is made. For contactless chip card readers a collision of two or more card signals may happen. In this case, if the device is not able to pick the strongest signal, `errorCardCollision` will be returned.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The original message name, for example "CardReader.Status"

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
track1	boolean	false	Track 1 of the magnetic stripe will be read.
track2	boolean	false	Track 2 of the magnetic stripe will be read.
track3	boolean	false	Track 3 of the magnetic stripe will be read.
chip	boolean	false	The chip will be read.
security	boolean	false	A security check will be performed.
fluxinactive	boolean	false	If the IDC Flux Sensor is programmable it will be disabled in order to allow chip data to be read on cards which have no magnetic stripes.
watermark	boolean	false	The Swedish Watermark track will be read.
memoryChip	boolean	false	The memory chip will be read.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track1Front	boolean	false	Track 1 data is read from the magnetic stripe located on the front of the card. In some countries this track is known as JIS II track.
frontImage	boolean	false	The front image of the card will be read in Base64 PNG format.
backImage	boolean	false	The back image of the card will be read in Base64 PNG format.
track1JIS	boolean	false	Track 1 of Japanese cash transfer card will be read. In some countries this track is known as JIS I track 1 (8bits/char).
track3JIS	boolean	false	Track 3 of Japanese cash transfer card will be read. In some countries this track is known as JIS I track 1 (8bits/char).
ddi	boolean	false	Dynamic Digital Identification data of the magnetic stripe will be read.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "track1": false,
    "track2": false,
    "track3": false,
    "chip": false,
    "security": false,
    "fluxInactive": false,
    "watermark": false,
    "memoryChip": false,
    "track1Front": false,
    "frontImage": false,
    "backImage": false,
    "track1JIS": false,
    "track3JIS": false,
    "ddi": false
  }
}
```

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
status	string		<p>The status of the command as one of the following:</p> <p><b>ok</b>&lt;br&gt; Successful.</p> <p><b>errorMediaJam</b>&lt;br&gt; The card is jammed. Operator intervention is required.</p> <p><b>errorShutterFail</b>&lt;br&gt; The open of the shutter failed due to manipulation or hardware error. Operator intervention is required.</p> <p><b>errorNoMedia</b>&lt;br&gt; The card was removed before completion of the read action (the <a href="#">CardReader.MediaInsertedEvent</a> has been generated). For motor driven devices, the read is disabled; i.e. another command has be issued to enable the reader for card entry.</p> <p><b>errorInvalidMedia</b>&lt;br&gt; No track or chip found; card may have been inserted or pulled through the wrong way.</p> <p><b>errorCardTooShort</b>&lt;br&gt; The card that was inserted is too short. When this error occurs the card remains at the exit slot.</p> <p><b>errorCardTooLong</b>&lt;br&gt; The card that was inserted is too long. When this error occurs the card remains at the exit slot.</p> <p><b>errorSecurityFail</b>&lt;br&gt; The security module failed reading the cards security sign.</p> <p><b>errorCardCollision</b>&lt;br&gt; There was an unresolved collision of two or more contactless card signals.</p>
data	array		An array of card data structures

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
			Specifies the source of the card data as one of the following:
		<b>track1</b> 	data contains data read from track 1.
		<b>track2</b> 	data contains data read from track 2.
		<b>track3</b> 	data contains data read from track 3.
		<b>chip</b> 	data contains ATR data read from the chip. For contactless chip card readers, multiple identification information can be returned if the card reader detects more than one chip. Each chip identification information is returned as an individual IppCardData array element.
		<b>security</b> 	data contains the value returned by the security module.
		<b>watermark</b> 	data contains data read from the Swedish Watermark track.
<b>data.dataSource</b>	<b>string</b>	<b>memoryChip</b> 	data contains Memory Card Identification data read from the memory chip.
		<b>track1Front</b> 	data contains data read from the front track 1. In some countries this track is known as JIS II track.
		<b>frontImage</b> 	data contains a null-terminated string containing the full path and file name of the BMP image file for the front of the card.
		<b>backImage</b> 	data contains a null-terminated string containing the full path and file name of the BMP image file for the back of the card.
		<b>track1JIS</b> 	data contains data read from JIS I track 1 (8bits/char).
		<b>track3JIS</b> 	data contains data read from JIS I track 3 (8bits/char).
		<b>ddi</b> 	data contains dynamic digital identification data read from magnetic stripe.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
data.status	string		<p>Status of reading the card data. Possible values are:</p> <p><b>ok</b>&lt;br&gt;The data is OK.</p> <p><b>errorDataMissing</b>&lt;br&gt;The track/chip/memory chip is blank.</p> <p><b>errorDataInvalid</b>&lt;br&gt;The data contained on the track/chip/memory chip is invalid. This will typically be returned when lpbData reports WFS_IDC_SEC_BADREADLEVEL or WFS_IDC_SEC_DATAINVAL.</p> <p><b>errorDataTooLong</b>&lt;br&gt;The data contained on the track/chip/memory chip is too long.</p> <p><b>errorDataTooShort</b>&lt;br&gt;The data contained on the track/chip/memory chip is too short.</p> <p><b>errorDataSourceNotSupported</b>&lt;br&gt;The data source to read from is not supported by the Service Provider.</p> <p><b>errorDataSourceMissing</b>&lt;br&gt;The data source to read from is missing on the card, or is unable to be read due to a hardware problem, or the module has not been initialized. For example, this will be returned on a request to read a Memory Card and the customer has entered a magnetic card without associated memory chip. This will also be reported when lpbData reports WFS_IDC_SEC_NODATA, WFS_IDC_SEC_NOINIT or WFS_IDC_SEC_HWERRO. This will also be reported when the image reader could not create a BMP file due to the state of the image reader or due to a failure.</p>
data.data	string		BASE64 encoded representation of the data

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "status": "ok",
    "data": [
      {
        "dataSource": "track1",
        "status": "ok",
        "data": "JUI0ODE1ODgxMDAyODYxODk2X11BVEVTL0VVR0VORSBKT0h0ICAgICAgICAgXjM3ODI5ODIxMDAw MTIzNDU2Nzg5Pw=="
      }
    ]
  }
}
```

**Event Messages**

- [CardReader.InsertCardEvent](#)
- [CardReader.MediaInsertedEvent](#)
- [CardReader.MediaRemovedEvent](#)
- [CardReader.InvalidMediaEvent](#)
- [CardReader.TrackDetectedEvent](#)

---

**CardReader.WriteRawData**

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Description

For motor-driven card readers, the ID card unit checks whether a card has been inserted. If so, the data is written to the tracks.

If no card has been inserted, and for all other categories of devices, the ID card unit waits for the application specified timeout for a card to be either inserted or pulled through. The next step is writing the data to the respective tracks.

The [CardReader.InsertCardEvent](#) event will be generated when there is no card in the card reader and the device is ready to accept a card.

The application must pass the magnetic stripe data in ASCII without any sentinels. The data will be converted by the Service Provider (ref [CardReader.ReadRawData](#)). If the data passed in is too long the invalidError error code will be returned.

This procedure is followed by data verification.

If power fails during a write the outcome of the operation will be vendor specific, there is no guarantee that the write will have succeeded.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <small>(Required)</small>	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 <small>(Required)</small>	string		The message type, either command, response, event or completion.
name <small>(Required)</small>	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	array		An array of card data structures
data.destination	string		Specifies where the card data is to be written to as one of the following:  <b>track1</b>   data is to be written to track 1. <b>track2</b>   data is to be written to track 2. <b>track3</b>   data is to be written to track 3.  <b>track1Front</b>   data is to be written to the front track 1. In some countries this track is known as JIS II track.  <b>track1JIS</b>   data is to be written to JIS I track 1 (8bits/char).  <b>track3JIS</b>   data is to be written to JIS I track 3 (8bits/char).
data.data	string		BASE64 encoded representation of the data
data.writeMethod	string	auto	Indicates whether a low coercivity or high coercivity magnetic stripe is to be written as one of the following:  <b>loco</b>   Write using low coercivity. <b>hico</b>   Write using high coercivity.  <b>auto</b>   Service Provider will determine whether low or high coercivity is to be used.

#### Example Message (generated)

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "data": [
      {
        "destination": "track1",
        "data": "JUI0ODE1ODgxMDAyODYxODk2X11lBVEVTL0VVR0VORSBKT0hOICAgICAgICAgXjM3ODI5ODIxMDAw MTIzNDU2Nzg5Pw==",
        "writeMethod": "auto"
      }
    ]
  }
}
```

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

- [CardReader.InsertCardEvent](#)
- [CardReader.MediaInsertedEvent](#)
- [CardReader.MediaRemovedEvent](#)
- [CardReader.InvalidTrackDataEvent](#)
- [CardReader.InvalidMediaEvent](#)

## CardReader.ChipIO

#### Description

This command is used to communicate with the chip. Transparent data is sent from the application to the chip and the response of the chip is returned transparently to the application.

The identification information e.g. ATR of the chip must be obtained before issuing this command. The identification information for a user card or the Memory Card Identification (when available) must initially be obtained using [CardReader.ReadRawData](#). The identification information for subsequent resets of a user card can be obtained using either CardReader.ReadRawData [CardReader.ChipPower](#). The ATR for permanent connected chips is always obtained through CardReader.ChipPower.

For contactless chip card readers, applications need to specify which chip to contact with, as part of *chipData*, if more than one chip has been detected and multiple identification data has been returned by the CardReader.ReadRawData command.

For contactless chip card readers a collision of two or more card signals may happen. In this case, if the device is not able to pick the strongest signal, the errCardCollision error code will be returned.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

**Command Message**

**Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
chipProtocol	string		Identifies the protocol that is used to communicate with the chip. Possible values are those described in CardReader.Capabilities. This field is ignored in communications with Memory Cards. The Service Provider knows which memory card type is currently inserted and therefore there is no need for the application to manage this.
chipData	string		The Base64 encoded data to be sent to the chip.

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
chipProtocol	string		Identifies the protocol that is used to communicate with the chip. This field contains the same value as the corresponding field in the payload. This field should be ignored in Memory Card dialogs and will contain WFS_IDC_NOTSUPP when returned for any Memory Card dialog.
chipData	string		The Base64 encoded data received from the chip.

**Example Message (generated)**

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

### Event Messages

- [CardReader.MediaRemovedEvent](#)

## CardReader.Reset

### Description

This command is used by the application to perform a hardware reset which will attempt to return the IDC device to a known good state. This command does not over-ride a lock obtained by another application or service handle.

If the device is a user ID card unit, the device will attempt to either retain, eject or will perform no action on any user cards found in the IDC as specified in the input parameter. It may not always be possible to retain or eject the items as specified because of hardware problems. If a user card is found inside the device the [CardReader.MediaInsertedEvent](#) will inform the application where card was actually moved to. If no action is specified the user card will not be moved even if this means that the IDC cannot be recovered.

If the device is a permanent chip card unit, this command will power-off the chip.

For devices with parking station capability there will be one *MediaInsertedEvent* for each card found.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
resetIn	string		Specifies the action to be performed on any user card found within the ID card unit as one of the following values:  <b>eject</b>   Eject any card found. <b>retain</b>   Retain any card found. <b>noAction</b>   No action should be performed on any card found.

#### Example Message (generated)

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

- [CardReader.MediaRemovedEvent](#)
- [CardReader.RetainBinThresholdEvent](#)
- [CardReader.MediaDetectedEvent](#)

## CardReader.ChipPower

#### Description

This command handles the power actions that can be done on the chip. For user chips, this command is only used after the chip has been contacted for the first time using the [CardReader.ReadRawData](#) command. For contactless user chips, this command may be used to deactivate the contactless card communication. For permanently connected chip cards, this command is the only way to control the chip power.

#### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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.
chipPower	string		Specifies the action to perform as one of the following: *cold  The chip is powered on and reset. *warm  The chip is reset. off  The chip is powered off.

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
chipData	string		The Base64 encoded data received from the chip.

**Example Message (generated)**

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

**Event Messages**

- [CardReader.MediaRemovedEvent](#)

---

## CardReader.ParseData

---

**Description**

This command takes the form name and output of a successful [CardReader.ReadRawData](#) and returns the parsed string.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
**Command Message**

**Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
formName	string		The name of the form that defines the behavior for the reading of tracks.
data	array		An array of card data structures
items	object		Specifies the source of the card data as one of the following:  <b>track1</b>   data contains data read from track 1. <b>track2</b>   data contains data read from track 2. <b>track3</b>   data contains data read from track 3. <b>chip</b>   data contains ATR data read from the chip. For contactless chip card readers, multiple identification information can be returned if the card reader detects more than one chip. Each chip identification information is returned as an individual lppCardData array element. <b>security</b>   data contains the value returned by the security module. <b>watermark</b>   data contains data read from the Swedish Watermark track. <b>memoryChip</b>   data contains Memory Card Identification data read from the memory chip. <b>track1Front</b>   data contains data read from the front track 1. In some countries this track is known as JIS II track. <b>frontImage</b>   data contains a null-terminated string containing the full path and file name of the BMP image file for the front of the card. <b>backImage</b>   data contains a null-terminated string containing the full path and file name of the BMP image file for the back of the card. <b>track1JIS</b>   data contains data read from JIS I track 1 (8bits/char). <b>track3JIS</b>   data contains data read from JIS I track 3 (8bits/char). <b>ddi</b>   data contains dynamic digital identification data read from magnetic stripe.
items.dataSource	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
items.status	string		<p>Status of reading the card data. Possible values are:</p> <p><b>ok</b>&lt;br&gt; The data is OK.</p> <p><b>errorDataMissing</b>&lt;br&gt; The track/chip/memory chip is blank.</p> <p><b>errorDataInvalid</b>&lt;br&gt; The data contained on the track/chip/memory chip is invalid. This will typically be returned when lpbData reports WFS_IDC_SEC_BADREADLEVEL or WFS_IDC_SEC_DATAINVAL.</p> <p><b>errorDataTooLong</b>&lt;br&gt; The data contained on the track/chip/memory chip is too long.</p> <p><b>errorDataTooShort</b>&lt;br&gt; The data contained on the track/chip/memory chip is too short.</p> <p><b>errorDataSourceNotSupported</b>&lt;br&gt; The data source to read from is not supported by the Service Provider.</p> <p><b>errorDataSourceMissing</b>&lt;br&gt; The data source to read from is missing on the card, or is unable to be read due to a hardware problem, or the module has not been initialized. For example, this will be returned on a request to read a Memory Card and the customer has entered a magnetic card without associated memory chip. This will also be reported when lpbData reports WFS_IDC_SEC_NODATA, WFS_IDC_SEC_NOINIT or WFS_IDC_SEC_HWEERROR. This will also be reported when the image reader could not create a BMP file due to the state of the image reader or due to a failure.</p>
items.data	string		BASE64 encoded representation of the data

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "formName": "string",
    "data": [],
    "items": [
      {
        "dataSource": "track1",
        "status": "ok",
        "data": "JUI0ODE1ODgxMDAyODYxODk2Xl1BVEVTL0VVR0VORSBKT0hOICAgICAQgXjM3ODI5ODIxMDAw MTIzNDU2Nzg5Pw=="
      }
    ]
  }
}
```

**Completion Message**

**Message Header**

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

**Message Payload**

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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"
  }
}
```

### Event Messages

- [CardReader.InvalidTrackDataEvent](#)

## CardReader.ParkCard

### Description

This command is used to move a card that is present in the reader to a parking station. A parking station is defined as an area in the IDC, which can be used to temporarily store the card while the device performs operations on another card. This command is also used to move a card from the parking station to the read/write, chip I/O or transport position. When a card is moved from the parking station to the read/write, chip I/O or transport position (*parkOut*), the read/write, chip I/O or transport position must not be occupied with another card, otherwise the error *cardPresent* will be returned.

After moving a card to a parking station, another card can be inserted and read by calling, e.g., [CardReader.ReadRawData](#) or [CardReader.ReadTrack](#).

Cards in parking stations will not be affected by any IDC commands until they are removed from the parking station using this command, except for the [CardReader.Reset](#) command, which will move the cards in the parking stations as specified in its input as part of the reset action if possible.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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.
direction	string		Specifies which way to move the card as one of the following values:  <b>in</b>   The card is moved to the parking station from the read/write, chip I/O or transport position. <b>out</b>   The card is moved from the parking station to the read/write, chip I/O or transport position. Once the card has been moved any command can be executed e.g. <a href="#">CardReader.EjectCard</a> or <a href="#">CardReader.ReadRawData</a> .

### Example Message (generated)

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Completion Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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"
  }
}
```

### Event Messages

---

## CardReader.EMVClessConfigure

---

#### Description

This command is used to configure an intelligent contactless card reader before performing a contactless transaction. This command sets terminal related data elements, the list of terminal acceptable applications with associated application specific data and any encryption key data required for offline data authentication.

This command should be used prior to [CardReader.EMVClassPerformTransaction](#) if the command. It may be called once on application start up or when any of the configuration parameters require to be changed. The configuration set by this command is persistent.

This command should be called with a complete list of acceptable payment system applications as any previous configurations will be replaced.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

Example Message (generated)

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

### Event Messages

## CardReader.EMVClessPerformTransaction

#### Description

This command is used to enable an intelligent contactless card reader. The transaction will start as soon as the card tap is detected.

Based on the configuration of the contactless chip card and the reader device, this command could return data formatted either as magnetic stripe information or as a set of BER-TLV encoded EMV tags.

This command supports magnetic stripe emulation cards and EMV-like contactless cards but cannot be used on storage contactless cards. The latter must be managed using the [CardReader.ReadRawData](#) and [CardReader.ChipIO](#) commands.

For specific payment system's card profiles an intelligent card reader could return a set of EMV tags along with magnetic stripe formatted data. In this case, two contactless card data structures will be returned, one containing the magnetic stripe like data and one containing BER-TLV encoded tags.

If no card has been tapped, the contactless chip card reader waits for the period of time specified in the command call for a card to be tapped.

For intelligent contactless card readers, any in-built audio/visual feedback such as Beep/LEDs, need to be controlled directly by the reader. These indications should be implemented based on the EMVCo and payment system's specifications.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
type <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"
<b>Message Payload</b>			
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.
			BASE64 encoded representation of the EMV data elements in a BER-TLV format required to perform a transaction. The types of object that could be included are:
			<ul style="list-style-type: none"> <li>• Transaction Type (9C)</li> <li>• Amount Authorized (9F02)</li> <li>• Transaction Date (9A)*</li> <li>• Transaction Time (9F21)*</li> <li>• Transaction Currency Code (5F2A)</li> </ul>
data	string		Individual payment systems could define further data elements.
			Tags are not mandatory with this command and this value can be omitted.
			*Tags 9A and 9F21 could be managed internally by the reader. If tags are not supplied, tag values may be used from the configuration sent previously in the <a href="#">CardReader.EMVClessConfigure</a> command.

**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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track1	object		<p>Contains the chip returned data formatted in as track 1. This value is set after the contactless transaction has been completed with mag-stripe mode.</p> <p>If multiple data sources are returned, this parameter should be the same for each one. Specifies the contactless transaction outcome as one of the following flags:</p> <ul style="list-style-type: none"> <li>• <b>multipleCards</b>&lt;br&gt; Transaction could not be completed as more than one contactless card was tapped.</li> <li>• <b>approve</b>&lt;br&gt; Transaction was approved offline.</li> <li>• <b>decline</b>&lt;br&gt; Transaction was declined offline.</li> <li>• <b>onlineRequest</b>&lt;br&gt; Transaction was requested for online authorization.</li> <li>• <b>onlineRequestCompletionRequired</b>&lt;br&gt; Transaction requested online authorization and will be completed after a re-tap of the card. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassIssuerUpdate</a> command.</li> <li>• <b>tryAgain</b>&lt;br&gt; Transaction could not be completed due to a card read error. The contactless card could be tapped again to re-attempt the transaction.</li> <li>• <b>tryAnotherInterface</b>&lt;br&gt; Transaction could not be completed over the contactless interface. Another interface may be suitable for this transaction (for example contact).</li> <li>• <b>endApplication</b>&lt;br&gt; Transaction cannot be completed on the contactless card due to an irrecoverable error.</li> <li>• <b>confirmationRequired</b>&lt;br&gt; Transaction was not completed as a result of a requirement to allow entry of confirmation code on a mobile device. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassPerformTransaction</a> after a card removal and a re-tap of the card.</li> </ul> <p>NOTE: The values for outcome have been mapped against the EMV Entry Point Outcome structure values defined in the EMVCo Specifications for Contactless Payment Systems (Book A and B).</p> <p>Specifies the cardholder action as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>none</b>&lt;br&gt; Transaction was completed. No further action is required.</li> <li>• <b>retap</b>&lt;br&gt; The contactless card should be re-tapped to complete the transaction. This value can be returned when txOutcome is onlineRequestCompletionRequired or confirmationRequired.</li> <li>• <b>holdCard</b>&lt;br&gt; The contactless card should not be removed from the field until the transaction is completed.</li> </ul>
track1.txOutcome	string		
track1.cardholderAction	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track1.dataRead	string		The BASE64 encoded representation of the data read from the chip after a contactless transaction has been completed successfully. If the data source is chip, the BER-TLV formatted data contains cryptogram tag (9F26) after a contactless chip transaction has been completed successfully. If the data source is track1, track2 or track3 this contains the data read from the chip, i.e the value returned by the card reader device and no cryptogram tag (9F26). This value is terminated with a single null character and cannot contain UNICODE characters.
track1.clessOutcome	object		The Entry Point Outcome specified in EMVCo Specifications for Contactless Payment Systems (Book A and B). This can be omitted for contactless chip card readers that do not follow EMVCo Entry Point Specifications.
track1.clessOutcome.cvm	string		<p>Specifies the cardholder verification method (CVM) to be performed as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>onlinePIN</b>&lt;br&gt; Online PIN should be entered by the cardholder.</li> <li>• <b>confirmationCodeVerified</b>&lt;br&gt; A confirmation code entry has been successfully done on a mobile device.</li> <li>• <b>sign</b>&lt;br&gt; Application should obtain cardholder signature.</li> <li>• <b>noCVM</b>&lt;br&gt; No CVM is required for this transaction.</li> <li>• <b>noCVMPreference</b>&lt;br&gt; There is no CVM preference, but application can follow the payment system's rules to process the transaction.</li> </ul>
track1.clessOutcome.alternateInterface	string		If txOutcome is not tryAnotherInterface, this should be ignored. If txOutcome is tryAnotherInterface, this specifies the alternative interface to be used to complete a transaction as one of the following: <ul style="list-style-type: none"> <li>• <b>contact</b>&lt;br&gt; Contact chip interface should be used to complete a transaction.</li> <li>• <b>magneticStripe</b>&lt;br&gt; Magnetic stripe interface should be used to complete a transaction.</li> </ul>
track1.clessOutcome.receipt	boolean		Specifies whether a receipt should be printed. True indicates that a receipt is required.
track1.clessOutcome.uiOutcome	object		The user interface details required to be displayed to the cardholder after processing the outcome of a contactless transaction. If no user interface details are required, this will be omitted. Please refer to EMVCo Contactless Specifications for Payment Systems Book A, Section 6.2 for details of the data within this object.
track1.clessOutcome.uiOutcome.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track1.clessOutcome.uiOutcome.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
track1.clessOutcome.uiOutcome.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
track1.clessOutcome.uiOutcome.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
track1.clessOutcome.uiOutcome.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
track1.clessOutcome.uiOutcome.value.display	string		The value to be displayed.
track1.clessOutcome.uiOutcome.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
track1.clessOutcome.uiOutcome.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
track1.clessOutcome.uiRestart	object		The user interface details required to be displayed to the cardholder when a transaction needs to be completed with a re-tap. If no user interface details are required, this will be omitted.
track1.clessOutcome.uiRestart.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track1.clessOutcome.uiRestart.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
track1.clessOutcome.uiRestart.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
track1.clessOutcome.uiRestart.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
track1.clessOutcome.uiRestart.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
track1.clessOutcome.uiRestart.value.display	string		The value to be displayed.
track1.clessOutcome.uiRestart.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
track1.clessOutcome.uiRestart.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
track1.clessOutcome.fieldOffHoldTime	integer		The application should wait for this specific hold time in units of 100 milliseconds, before re-enabling the contactless card reader by issuing either the <a href="#">CardReader.EMVClessPerformTransaction</a> command or the <a href="#">CardReader.EMVClessIssuerUpdate</a> command depending on the value of txOutcome. For intelligent contactless card readers, the completion of this command ensures that the contactless chip card reader field is automatically turned off, so there is no need for the application to disable the field.
track1.clessOutcome.cardRemovalTimeout	integer		Specifies a timeout value in units of 100 milliseconds for prompting the user to remove the card.
track1.clessOutcome.discretionaryData	string		Base64 encoded representation of the payment system's specific discretionary data read from the chip, in a BER-TLV format, after a contactless transaction has been completed. If discretionary data is not present, this will be omitted.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track2	object		<p>Contains the chip returned data formatted in as track 2. This value is set after the contactless transaction has been completed with mag-stripe mode.</p> <p>If multiple data sources are returned, this parameter should be the same for each one. Specifies the contactless transaction outcome as one of the following flags:</p> <ul style="list-style-type: none"> <li>• <b>multipleCards</b>&lt;br&gt; Transaction could not be completed as more than one contactless card was tapped.</li> <li>• <b>approve</b>&lt;br&gt; Transaction was approved offline.</li> <li>• <b>decline</b>&lt;br&gt; Transaction was declined offline.</li> <li>• <b>onlineRequest</b>&lt;br&gt; Transaction was requested for online authorization.</li> <li>• <b>onlineRequestCompletionRequired</b>&lt;br&gt; Transaction requested online authorization and will be completed after a re-tap of the card. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassIssuerUpdate</a> command.</li> <li>• <b>tryAgain</b>&lt;br&gt; Transaction could not be completed due to a card read error. The contactless card could be tapped again to re-attempt the transaction.</li> <li>• <b>tryAnotherInterface</b>&lt;br&gt; Transaction could not be completed over the contactless interface. Another interface may be suitable for this transaction (for example contact).</li> <li>• <b>endApplication</b>&lt;br&gt; Transaction cannot be completed on the contactless card due to an irrecoverable error.</li> <li>• <b>confirmationRequired</b>&lt;br&gt; Transaction was not completed as a result of a requirement to allow entry of confirmation code on a mobile device. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassPerformTransaction</a> after a card removal and a re-tap of the card.</li> </ul> <p>NOTE: The values for outcome have been mapped against the EMV Entry Point Outcome structure values defined in the EMVCo Specifications for Contactless Payment Systems (Book A and B).</p> <p>Specifies the cardholder action as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>none</b>&lt;br&gt; Transaction was completed. No further action is required.</li> <li>• <b>retap</b>&lt;br&gt; The contactless card should be re-tapped to complete the transaction. This value can be returned when txOutcome is onlineRequestCompletionRequired or confirmationRequired.</li> <li>• <b>holdCard</b>&lt;br&gt; The contactless card should not be removed from the field until the transaction is completed.</li> </ul>
track2.txOutcome	string		
track2.cardholderAction	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track2.dataRead	string		The BASE64 encoded representation of the data read from the chip after a contactless transaction has been completed successfully. If the data source is chip, the BER-TLV formatted data contains cryptogram tag (9F26) after a contactless chip transaction has been completed successfully. If the data source is track1, track2 or track3 this contains the data read from the chip, i.e the value returned by the card reader device and no cryptogram tag (9F26). This value is terminated with a single null character and cannot contain UNICODE characters.
track2.clessOutcome	object		The Entry Point Outcome specified in EMVCo Specifications for Contactless Payment Systems (Book A and B). This can be omitted for contactless chip card readers that do not follow EMVCo Entry Point Specifications.
track2.clessOutcome.cvm	string		Specifies the cardholder verification method (CVM) to be performed as one of the following: <ul style="list-style-type: none"> <li>• <b>onlinePIN</b>&lt;br&gt; Online PIN should be entered by the cardholder.</li> <li>• <b>confirmationCodeVerified</b>&lt;br&gt; A confirmation code entry has been successfully done on a mobile device.</li> <li>• <b>sign</b>&lt;br&gt; Application should obtain cardholder signature.</li> <li>• <b>noCVM</b>&lt;br&gt; No CVM is required for this transaction.</li> <li>• <b>noCVMPreference</b>&lt;br&gt; There is no CVM preference, but application can follow the payment system's rules to process the transaction.</li> </ul>
track2.clessOutcome.alternateInterface	string		If txOutcome is not tryAnotherInterface, this should be ignored. If txOutcome is tryAnotherInterface, this specifies the alternative interface to be used to complete a transaction as one of the following: <ul style="list-style-type: none"> <li>• <b>contact</b>&lt;br&gt; Contact chip interface should be used to complete a transaction.</li> <li>• <b>magneticStripe</b>&lt;br&gt; Magnetic stripe interface should be used to complete a transaction.</li> </ul>
track2.clessOutcome.receipt	boolean		Specifies whether a receipt should be printed. True indicates that a receipt is required.
track2.clessOutcome.uiOutcome	object		The user interface details required to be displayed to the cardholder after processing the outcome of a contactless transaction. If no user interface details are required, this will be omitted. Please refer to EMVCo Contactless Specifications for Payment Systems Book A, Section 6.2 for details of the data within this object.
track2.clessOutcome.uiOutcome.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track2.clessOutcome.uiOutcome.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
track2.clessOutcome.uiOutcome.holdTime	integer		<p>The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.</p>
track2.clessOutcome.uiOutcome.value	object		<p>The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.</p>
track2.clessOutcome.uiOutcome.value.qualifier	string		<p>The value qualifier as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
track2.clessOutcome.uiOutcome.value.display	string		<p>The value to be displayed.</p>
track2.clessOutcome.uiOutcome.currencyCode	string		<p>The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.</p>
track2.clessOutcome.uiOutcome.languagePreferenceData	string		<p>The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.</p>
track2.clessOutcome.uiRestart	object		<p>The user interface details required to be displayed to the cardholder when a transaction needs to be completed with a re-tap. If no user interface details are required, this will be omitted.</p>
track2.clessOutcome.uiRestart.messageID	integer		<p>A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track2.clessOutcome.uiRestart.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
track2.clessOutcome.uiRestart.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
track2.clessOutcome.uiRestart.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
track2.clessOutcome.uiRestart.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
track2.clessOutcome.uiRestart.value.display	string		The value to be displayed.
track2.clessOutcome.uiRestart.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
track2.clessOutcome.uiRestart.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
track2.clessOutcome.fieldOffHoldTime	integer		The application should wait for this specific hold time in units of 100 milliseconds, before re-enabling the contactless card reader by issuing either the <a href="#">CardReader.EMVClessPerformTransaction</a> command or the <a href="#">CardReader.EMVClessIssuerUpdate</a> command depending on the value of txOutcome. For intelligent contactless card readers, the completion of this command ensures that the contactless chip card reader field is automatically turned off, so there is no need for the application to disable the field.
track2.clessOutcome.cardRemovalTimeout	integer		Specifies a timeout value in units of 100 milliseconds for prompting the user to remove the card.
track2.clessOutcome.discretionaryData	string		Base64 encoded representation of the payment system's specific discretionary data read from the chip, in a BER-TLV format, after a contactless transaction has been completed. If discretionary data is not present, this will be omitted.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track3	object		<p>Contains the chip returned data formatted in as track 3. This value is set after the contactless transaction has been completed with mag-stripe mode.</p> <p>If multiple data sources are returned, this parameter should be the same for each one. Specifies the contactless transaction outcome as one of the following flags:</p> <ul style="list-style-type: none"> <li>• <b>multipleCards</b>&lt;br&gt; Transaction could not be completed as more than one contactless card was tapped.</li> <li>• <b>approve</b>&lt;br&gt; Transaction was approved offline.</li> <li>• <b>decline</b>&lt;br&gt; Transaction was declined offline.</li> <li>• <b>onlineRequest</b>&lt;br&gt; Transaction was requested for online authorization.</li> <li>• <b>onlineRequestCompletionRequired</b>&lt;br&gt; Transaction requested online authorization and will be completed after a re-tap of the card. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassIssuerUpdate</a> command.</li> <li>• <b>tryAgain</b>&lt;br&gt; Transaction could not be completed due to a card read error. The contactless card could be tapped again to re-attempt the transaction.</li> <li>• <b>tryAnotherInterface</b>&lt;br&gt; Transaction could not be completed over the contactless interface. Another interface may be suitable for this transaction (for example contact).</li> <li>• <b>endApplication</b>&lt;br&gt; Transaction cannot be completed on the contactless card due to an irrecoverable error.</li> <li>• <b>confirmationRequired</b>&lt;br&gt; Transaction was not completed as a result of a requirement to allow entry of confirmation code on a mobile device. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassPerformTransaction</a> after a card removal and a re-tap of the card.</li> </ul> <p>NOTE: The values for outcome have been mapped against the EMV Entry Point Outcome structure values defined in the EMVCo Specifications for Contactless Payment Systems (Book A and B).</p> <p>Specifies the cardholder action as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>none</b>&lt;br&gt; Transaction was completed. No further action is required.</li> <li>• <b>retap</b>&lt;br&gt; The contactless card should be re-tapped to complete the transaction. This value can be returned when txOutcome is onlineRequestCompletionRequired or confirmationRequired.</li> <li>• <b>holdCard</b>&lt;br&gt; The contactless card should not be removed from the field until the transaction is completed.</li> </ul>
track3.txOutcome	string		
track3.cardholderAction	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track3.dataRead	string		The BASE64 encoded representation of the data read from the chip after a contactless transaction has been completed successfully. If the data source is chip, the BER-TLV formatted data contains cryptogram tag (9F26) after a contactless chip transaction has been completed successfully. If the data source is track1, track2 or track3 this contains the data read from the chip, i.e the value returned by the card reader device and no cryptogram tag (9F26). This value is terminated with a single null character and cannot contain UNICODE characters.
track3.clessOutcome	object		The Entry Point Outcome specified in EMVCo Specifications for Contactless Payment Systems (Book A and B). This can be omitted for contactless chip card readers that do not follow EMVCo Entry Point Specifications.
track3.clessOutcome.cvm	string		Specifies the cardholder verification method (CVM) to be performed as one of the following: <ul style="list-style-type: none"> <li>• <b>onlinePIN</b>&lt;br&gt; Online PIN should be entered by the cardholder.</li> <li>• <b>confirmationCodeVerified</b>&lt;br&gt; A confirmation code entry has been successfully done on a mobile device.</li> <li>• <b>sign</b>&lt;br&gt; Application should obtain cardholder signature.</li> <li>• <b>noCVM</b>&lt;br&gt; No CVM is required for this transaction.</li> <li>• <b>noCVMPreference</b>&lt;br&gt; There is no CVM preference, but application can follow the payment system's rules to process the transaction.</li> </ul>
track3.clessOutcome.alternateInterface	string		If txOutcome is not tryAnotherInterface, this should be ignored. If txOutcome is tryAnotherInterface, this specifies the alternative interface to be used to complete a transaction as one of the following: <ul style="list-style-type: none"> <li>• <b>contact</b>&lt;br&gt; Contact chip interface should be used to complete a transaction.</li> <li>• <b>magneticStripe</b>&lt;br&gt; Magnetic stripe interface should be used to complete a transaction.</li> </ul>
track3.clessOutcome.receipt	boolean		Specifies whether a receipt should be printed. True indicates that a receipt is required.
track3.clessOutcome.uiOutcome	object		The user interface details required to be displayed to the cardholder after processing the outcome of a contactless transaction. If no user interface details are required, this will be omitted. Please refer to EMVCo Contactless Specifications for Payment Systems Book A, Section 6.2 for details of the data within this object.
track3.clessOutcome.uiOutcome.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track3.clessOutcome.uiOutcome.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
track3.clessOutcome.uiOutcome.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
track3.clessOutcome.uiOutcome.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
track3.clessOutcome.uiOutcome.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
track3.clessOutcome.uiOutcome.value.display	string		The value to be displayed.
track3.clessOutcome.uiOutcome.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
track3.clessOutcome.uiOutcome.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
track3.clessOutcome.uiRestart	object		The user interface details required to be displayed to the cardholder when a transaction needs to be completed with a re-tap. If no user interface details are required, this will be omitted.
track3.clessOutcome.uiRestart.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
track3.clessOutcome.uiRestart.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
track3.clessOutcome.uiRestart.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
track3.clessOutcome.uiRestart.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
track3.clessOutcome.uiRestart.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
track3.clessOutcome.uiRestart.value.display	string		The value to be displayed.
track3.clessOutcome.uiRestart.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
track3.clessOutcome.uiRestart.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
track3.clessOutcome.fieldOffHoldTime	integer		The application should wait for this specific hold time in units of 100 milliseconds, before re-enabling the contactless card reader by issuing either the <a href="#">CardReader.EMVClessPerformTransaction</a> command or the <a href="#">CardReader.EMVClessIssuerUpdate</a> command depending on the value of txOutcome. For intelligent contactless card readers, the completion of this command ensures that the contactless chip card reader field is automatically turned off, so there is no need for the application to disable the field.
track3.clessOutcome.cardRemovalTimeout	integer		Specifies a timeout value in units of 100 milliseconds for prompting the user to remove the card.
track3.clessOutcome.discretionaryData	string		Base64 encoded representation of the payment system's specific discretionary data read from the chip, in a BER-TLV format, after a contactless transaction has been completed. If discretionary data is not present, this will be omitted.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
chip	object		<p>Contains the BER-TLV formatted data read from the chip. This value is set after the contactless transaction has been completed with EMV mode or mag-stripe mode.</p>
chip.txOutcome	string		<p>If multiple data sources are returned, this parameter should be the same for each one.</p> <p>Specifies the contactless transaction outcome as one of the following flags:</p> <ul style="list-style-type: none"> <li>• <b>multipleCards</b>&lt;br&gt; Transaction could not be completed as more than one contactless card was tapped.</li> <li>• <b>approve</b>&lt;br&gt; Transaction was approved offline.</li> <li>• <b>decline</b>&lt;br&gt; Transaction was declined offline.</li> <li>• <b>onlineRequest</b>&lt;br&gt; Transaction was requested for online authorization.</li> <li>• <b>onlineRequestCompletionRequired</b>&lt;br&gt; Transaction requested online authorization and will be completed after a re-tap of the card. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassIssuerUpdate</a> command.</li> <li>• <b>tryAgain</b>&lt;br&gt; Transaction could not be completed due to a card read error. The contactless card could be tapped again to re-attempt the transaction.</li> <li>• <b>tryAnotherInterface</b>&lt;br&gt; Transaction could not be completed over the contactless interface. Another interface may be suitable for this transaction (for example contact).</li> <li>• <b>endApplication</b>&lt;br&gt; Transaction cannot be completed on the contactless card due to an irrecoverable error.</li> <li>• <b>confirmationRequired</b>&lt;br&gt; Transaction was not completed as a result of a requirement to allow entry of confirmation code on a mobile device. Transaction should be completed by issuing the <a href="#">CardReader.EMVClassPerformTransaction</a> after a card removal and a re-tap of the card.</li> </ul> <p>NOTE: The values for outcome have been mapped against the EMV Entry Point Outcome structure values defined in the EMVCo Specifications for Contactless Payment Systems (Book A and B).</p>
chip.cardholderAction	string		<p>Specifies the cardholder action as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>none</b>&lt;br&gt; Transaction was completed. No further action is required.</li> <li>• <b>retap</b>&lt;br&gt; The contactless card should be re-tapped to complete the transaction. This value can be returned when txOutcome is onlineRequestCompletionRequired or confirmationRequired.</li> <li>• <b>holdCard</b>&lt;br&gt; The contactless card should not be removed from the field until the transaction is completed.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
chip.dataRead	string		The BASE64 encoded representation of the data read from the chip after a contactless transaction has been completed successfully. If the data source is chip, the BER-TLV formatted data contains cryptogram tag (9F26) after a contactless chip transaction has been completed successfully. If the data source is track1, track2 or track3 this contains the data read from the chip, i.e the value returned by the card reader device and no cryptogram tag (9F26). This value is terminated with a single null character and cannot contain UNICODE characters.
chip.clessOutcome	object		The Entry Point Outcome specified in EMVCo Specifications for Contactless Payment Systems (Book A and B). This can be omitted for contactless chip card readers that do not follow EMVCo Entry Point Specifications.
chip.clessOutcome.cvm	string		<p>Specifies the cardholder verification method (CVM) to be performed as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>onlinePIN</b>&lt;br&gt; Online PIN should be entered by the cardholder.</li> <li>• <b>confirmationCodeVerified</b>&lt;br&gt; A confirmation code entry has been successfully done on a mobile device.</li> <li>• <b>sign</b>&lt;br&gt; Application should obtain cardholder signature.</li> <li>• <b>noCVM</b>&lt;br&gt; No CVM is required for this transaction.</li> <li>• <b>noCVMPreference</b>&lt;br&gt; There is no CVM preference, but application can follow the payment system's rules to process the transaction.</li> </ul>
chip.clessOutcome.alternateInterface	string		If txOutcome is not tryAnotherInterface, this should be ignored. If txOutcome is tryAnotherInterface, this specifies the alternative interface to be used to complete a transaction as one of the following: <ul style="list-style-type: none"> <li>• <b>contact</b>&lt;br&gt; Contact chip interface should be used to complete a transaction.</li> <li>• <b>magneticStripe</b>&lt;br&gt; Magnetic stripe interface should be used to complete a transaction.</li> </ul>
chip.clessOutcome.receipt	boolean		Specifies whether a receipt should be printed. True indicates that a receipt is required.
chip.clessOutcome.uiOutcome	object		The user interface details required to be displayed to the cardholder after processing the outcome of a contactless transaction. If no user interface details are required, this will be omitted. Please refer to EMVCo Contactless Specifications for Payment Systems Book A, Section 6.2 for details of the data within this object.
chip.clessOutcome.uiOutcome.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
chip.clessOutcome.uiOutcome.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
chip.clessOutcome.uiOutcome.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
chip.clessOutcome.uiOutcome.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
chip.clessOutcome.uiOutcome.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
chip.clessOutcome.uiOutcome.value.display	string		The value to be displayed.
chip.clessOutcome.uiOutcome.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
chip.clessOutcome.uiOutcome.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
chip.clessOutcome.uiRestart	object		The user interface details required to be displayed to the cardholder when a transaction needs to be completed with a re-tap. If no user interface details are required, this will be omitted.
chip.clessOutcome.uiRestart.messageID	integer		A value which represents the EMVCo defined message identifier that indicates the text string to be displayed e.g. 27 is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
chip.clessOutcome.uiRestart.status	string		<p>The EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <ul style="list-style-type: none"> <li>• <b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</li> <li>• <b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</li> <li>• <b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</li> <li>• <b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</li> <li>• <b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</li> <li>• <b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</li> </ul>
chip.clessOutcome.uiRestart.holdTime	integer		The hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
chip.clessOutcome.uiRestart.value	object		The value of the amount or balance to be displayed. If no value is to be displayed this will be omitted.
chip.clessOutcome.uiRestart.value.qualifier	string		The value qualifier as one of the following: <ul style="list-style-type: none"> <li>• <b>amount</b>&lt;br&gt; The value is an Amount.</li> <li>• <b>balance</b>&lt;br&gt; The value is a Balance.</li> </ul>
chip.clessOutcome.uiRestart.value.display	string		The value to be displayed.
chip.clessOutcome.uiRestart.currencyCode	string		The numeric value of currency code as per ISO 4217. This will be omitted if the Currency Code is not available.
chip.clessOutcome.uiRestart.languagePreferenceData	string		The language preference (EMV Tag '5F2D') if returned by the card. If not returned, this will be omitted. The application should use this data to display all messages in the specified language until the transaction concludes.
chip.clessOutcome.fieldOffHoldTime	integer		The application should wait for this specific hold time in units of 100 milliseconds, before re-enabling the contactless card reader by issuing either the <a href="#">CardReader.EMVClessPerformTransaction</a> command or the <a href="#">CardReader.EMVClessIssuerUpdate</a> command depending on the value of txOutcome. For intelligent contactless card readers, the completion of this command ensures that the contactless chip card reader field is automatically turned off, so there is no need for the application to disable the field.
chip.clessOutcome.cardRemovalTimeout	integer		Specifies a timeout value in units of 100 milliseconds for prompting the user to remove the card.
chip.clessOutcome.discretionaryData	string		Base64 encoded representation of the payment system's specific discretionary data read from the chip, in a BER-TLV format, after a contactless transaction has been completed. If discretionary data is not present, this will be omitted.

Example Message (generated)

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
    "headers": {
      "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
      "type": "command",
      "name": "string"
    },
    "payload": {
      "completionCode": "success",
      "errorDescription": "string",
      "track1": {
        "txOutcome": "multipleCards",
        "cardholderAction": "none",
        "dataRead": "string",
        "clessOutcome": {
          "cvm": "onlinePIN",
          "alternateInterface": "contact",
          "receipt": true,
          "uiOutcome": {
            "messageID": 0,
            "status": "notReady",
            "holdTime": 0,
            "value": {
              "qualifier": "amount",
              "display": "string"
            }
          },
          "currencyCode": "string",
          "languagePreferenceData": "string"
        },
        "uiRestart": {
          "messageID": 0,
          "status": "notReady",
          "holdTime": 0,
          "value": {
            "qualifier": "amount",
            "display": "string"
          }
        },
        "currencyCode": "string",
        "languagePreferenceData": "string"
      },
      "fieldOffHoldTime": 0,
      "cardRemovalTimeout": 0,
      "discretionaryData": "string"
    }
  },
  "track2": {
    "txOutcome": "multipleCards",
    "cardholderAction": "none",
    "dataRead": "string",
    "clessOutcome": {
      "cvm": "onlinePIN",
      "alternateInterface": "contact",
      "receipt": true,
      "uiOutcome": {
        "messageID": 0,
        "status": "notReady",
        "holdTime": 0,
        "value": {
          "qualifier": "amount",
          "display": "string"
        }
      },
      "currencyCode": "string",
      "languagePreferenceData": "string"
    },
    "uiRestart": {
      "messageID": 0,
      "status": "notReady",
      "holdTime": 0,
      "value": {
        "qualifier": "amount",
        "display": "string"
      }
    },
    "currencyCode": "string",
    "languagePreferenceData": "string"
  },
  "fieldOffHoldTime": 0,
  "cardRemovalTimeout": 0,
  "discretionaryData": "string"
}
},
"track3": {
  "txOutcome": "multipleCards",
  "cardholderAction": "none",
  "dataRead": "string",
  "clessOutcome": {
    "cvm": "onlinePIN",
    "alternateInterface": "contact",
    "receipt": true,
    "uiOutcome": {
      "messageID": 0,
      "status": "notReady",
      "holdTime": 0,
      "value": {
        "qualifier": "amount",
        "display": "string"
      }
    },
    "currencyCode": "string",
    "languagePreferenceData": "string"
  },
  "fieldOffHoldTime": 0,
  "cardRemovalTimeout": 0,
  "discretionaryData": "string"
}
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
"receipt": true,
"uiOutcome": {
  "messageID": 0,
  "status": "notReady",
  "holdTime": 0,
  "value": {
    "qualifier": "amount",
    "display": "string"
  },
  "currencyCode": "string",
  "languagePreferenceData": "string"
},
"uiRestart": {
  "messageID": 0,
  "status": "notReady",
  "holdTime": 0,
  "value": {
    "qualifier": "amount",
    "display": "string"
  },
  "currencyCode": "string",
  "languagePreferenceData": "string"
},
"fieldOffHoldTime": 0,
"cardRemovalTimeout": 0,
"discretionaryData": "string"
),
),
"chip": {
  "txOutcome": "multipleCards",
  "cardholderAction": "none",
  "dataRead": "string",
  "clessOutcome": {
    "cvm": "onlinePIN",
    "alternateInterface": "contact",
    "receipt": true,
    "uiOutcome": {
      "messageID": 0,
      "status": "notReady",
      "holdTime": 0,
      "value": {
        "qualifier": "amount",
        "display": "string"
      },
      "currencyCode": "string",
      "languagePreferenceData": "string"
    },
    "uiRestart": {
      "messageID": 0,
      "status": "notReady",
      "holdTime": 0,
      "value": {
        "qualifier": "amount",
        "display": "string"
      },
      "currencyCode": "string",
      "languagePreferenceData": "string"
    },
    "fieldOffHoldTime": 0,
    "cardRemovalTimeout": 0,
    "discretionaryData": "string"
  }
}
}
```

#### Event Messages

- [CardReader.EMVClessReadStatusEvent](#)
- [CardReader.MediaRemovedEvent](#)

---

### CardReader.EMVClessIssuerUpdate

---

#### Description

This command performs the post authorization processing on payment systems contactless cards. Before an online authorized transaction is considered complete, further chip processing may be requested by the issuer. This is only required when the authorization response includes issuer update data; either issuer scripts or issuer authentication data. The command enables the contactless card reader and waits for the customer to re-tap their card. The contactless chip card reader waits for the period of time specified in the WFSEExecute call for a card to be tapped.

---

#### Command Message

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

- [CardReader.EMVClassReadStatusEvent](#)
- [CardReader.MediaRemovedEvent](#)

## Unsolicited Events

### CardReader.MediaRemovedEvent

#### Description

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This event specifies that the inserted card was manually removed by the user during the processing of a read command, during the processing of a chip\_io/power command, during or after a retain/reset operation, after an eject operation or after the card is removed by the user in a latched dip card unit.

### Message Header

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

### Example Message (generated)

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

---

## CardReader.CardActionEvent

---

### Description

This service event specifies that a card has been retained or ejected by either the automatic power on or power off action of the device.

### Message Header

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

### Message Payload

Name	Type	Default	Description
action	string		Specifies which action has been performed with the card. Possible values are: <b>retained</b>   The card has been retained. <b>ejected</b>   The card has been ejected. <b>readPosition</b>   The card has been moved to the read position.
position	string		Position of card before being retained or ejected. Possible values are: <b>unknown</b>   The position of the card cannot be determined. <b>present</b>   The card was present in the reader. <b>entering</b>   The card was entering the reader.

### Example Message (generated)

```
[{"headers": {  
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
    "type": "command",  
    "name": "string"  
},  
  "payload": {  
    "action": "retained",  
    "position": "entering"  
  }  
}
```

---

## CardReader.RetainBinThresholdEvent

---

### Description

This specifies that the retain bin holding the retained cards has reached a threshold condition or the threshold condition is removed.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
state	string		Specifies the state of the ID card unit retain bin as one of the following values:  <b>ok</b>   The retain bin of the ID card unit was emptied. <b>full</b>   The retain bin of the ID card unit is full. <b>high</b>   The retain bin of the ID card unit is nearly full."

### Example Message (generated)

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

## Events

### CardReader.InsertCardEvent

#### Description

This event notifies the application when the device is ready for the user to insert a card.

#### Message Header

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

### Example Message (generated)

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

### CardReader.MediaInsertedEvent

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Description

This event notifies the application when the device is ready for the user to insert a card.

#### Message Header

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

#### Example Message (generated)

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

---

## CardReader.InvalidTrackDataEvent

---

### Description

This execute event specifies that a track contained invalid or no data.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
			Status of reading the track as one of the following:
			<b>missing</b>   The track is blank.
status	string		<b>invalid</b>   The data contained on the track is invalid.
			<b>tooLong</b>   The data contained on the track is too long.
			<b>tooShort</b>   The data contained on the track is too short.
track	string		The keyword of the track on which the error occurred.
data	string		Any data from the track that could be read.

#### Example Message (generated)

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

## CardReader.InvalidMediaEvent

### Description

This event specifies that the media the user is attempting to insert is not a valid card or it is a card but it is in the wrong orientation.

### Message Header

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

### Example Message (generated)

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

## CardReader.TrackDetectedEvent

### Description

This execute event notifies the application what track data the inserted card has, before the reading of the data has completed. This event will be posted once when tracks are detected during card insertion.

### Message Header

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

### Message Payload

Name	Type	Default	Description
track1	boolean	false	The card reader has track 1.
track2	boolean	false	The card reader has track 2.
track3	boolean	false	The card reader has track 3.
watermark	boolean	false	The card reader has the Swedish watermark track.
frontTrack1	boolean	false	The card reader has front track 1.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Example Message (generated) Type Default Description

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "track1": false,
    "track2": false,
    "track3": false,
    "watermark": false,
    "frontTrack1": false
  }
}
```

### CardReader.MediaRetainedEvent

#### Description

This specifies that the card was retained.

#### Message Header

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

Example Message (generated)

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

### CardReader.MediaDetectedEvent

#### Description

This is generated if media is detected during a [CardReader.Reset](#). The parameter on the event informs the application of the position of the card on the completion of the reset. For devices with parking station capability there will be one event for each card found.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
			Specifies the action that was performed on any card found within the IDC as one of the following:  <b>ejected</b>   The card was ejected.  <b>retained</b>   The card was retained.  <b>readPosition</b>   The card is in read position.  <b>jammed</b>   The card is jammed in the device.
resetOut	string		

**Example Message (generated)**

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

## CardReader.EMVClessReadStatusEvent

### Description

This notifies that the communication (i.e. the commands exchanged linked to the tap) between the card and the intelligent contactless card reader are complete. The application can use this event to display intermediate messages, progress of card read, audio signals or anything else that might be required. The intelligent contactless card reader will continue the processing and the result of the processing will be returned in the output of the [CardReader.EMVClessPerformTransaction](#) command.

### Message Header

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

### Message Payload

Name	Type	Default	Description
messageId	integer		Represents the EMVCo defined message identifier that indicates the text string to be displayed, e.g., 0x1B is the "Authorising Please Wait" message (see EMVCo Contactless Specifications for Payment Systems Book A, Section 9.4).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
status	string		<p>Represents the EMVCo defined transaction status value to be indicated through the Beep/LEDs as one of the following:</p> <p><b>notReady</b>&lt;br&gt; Contactless card reader is not able to communicate with a card. This status occurs towards the end of a contactless transaction or if the reader is not powered on.</p> <p><b>idle</b>&lt;br&gt; Contactless card reader is powered on, but the reader field is not yet active for communication with a card.</p> <p><b>readyToRead</b>&lt;br&gt; Contactless card reader is powered on and attempting to initiate communication with a card.</p> <p><b>processing</b>&lt;br&gt; Contactless card reader is in the process of reading the card.</p> <p><b>cardReadOk</b>&lt;br&gt; Contactless card reader was able to read a card successfully.</p> <p><b>processingError</b>&lt;br&gt; Contactless card reader was not able to process the card successfully.</p>
holdTime	integer		Represents the hold time in units of 100 milliseconds for which the application should display the message before processing the next user interface data.
valueQualifier	string	notApplicable	<p>Qualifies <i>value</i>. This data is defined by EMVCo as one of the following:</p> <p><b>amount</b>&lt;br&gt; <i>value</i> is an Amount.</p> <p><b>balance</b>&lt;br&gt; <i>value</i> is a Balance.</p> <p><b>notApplicable</b>&lt;br&gt; <i>value</i> is neither of the above.</p>
value	string		Represents the value of the amount or balance (as specified by <i>valueQualifier</i> ) to be displayed where appropriate
currencyCode	string		Represents the numeric value of currency code as per ISO 4217.
languagePreferenceData	string		Represents the language preference (EMV Tag '5F2D') if returned by the card. The application should use this data to display all messages in the specified language until the transaction concludes.

**Example Message (generated)**

XFS4IoT  
All rights

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "messageId": 0,
    "status": "notReady",
    "holdTime": 0,
    "valueQualifier": "amount",
    "value": "123.45",
    "currencyCode": "GBP",
    "languagePreferenceData": ""
  }
}
```

# XFS4IOT Sample Messaging for CashAcceptor

## Draft 0.0.3

This specification describes the functionality of an XFS4IoT compliant Cash Acceptor interface. It defines the service-specific commands that can be issued to the service using the WebSocket endpoint.

Persistent values are maintained through power failures, open sessions, close session and system resets.

This specification covers the acceptance of items. An "item" is defined as any media that can be accepted and includes coupons, documents, bills and coins. However, if coins and bills are both to be accepted separate Services must be implemented for each.

## Commands

### CashAcceptor.BanknoteTypes

---

#### Description

This command is used to obtain information about the banknote types that can be detected by the banknote reader.

#### Command Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
noteTypes	array		List of banknote types the banknote reader supports.
noteTypes.noteID	integer		Identification of note type.
noteTypes.currencyID	string		Currency ID in ISO 4217 format [Ref. 2].
noteTypes.values	number		The value of a single item expressed as floating point value.
noteTypes.release	integer		The release of the banknote type. The higher this number is, the newer the release. Zero means that there is only one release of that banknote type. This value has not been standardized and therefore a release number of the same banknote will not necessarily have the same value in different systems.
noteTypes.configured	boolean		If TRUE the banknote reader will accept this note type during a cash-in operation, if FALSE the banknote reader will refuse this note type unless it must be retained by note classification rules.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "noteTypes": [
      {
        "noteID": 0,
        "currencyID": "string",
        "values": 0,
        "release": 0,
        "configured": true
      }
    ]
  }
}
```

## Event Messages

---

## CashAcceptor.CashInStatus

---

### Description

This command is used to get information about the status of the currently active cash-in transaction or in the case where no cash-in transaction is active the status of the most recently ended cash-in transaction. This value is persistent and is valid until the next command CashAcceptor.CashInStart.

### Command Message

#### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
status	string		<p>Status of the currently active or most recently ended cash-in transaction. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash-in transaction is complete and has ended with a CashAcceptor.CashInEnd command call.</li> <li>"rollback": The cash-in transaction was has ended with a CashAcceptor.CashInRollback command call.</li> <li>"active": There is a cash-in transaction active. See the CashAcceptor.CashInStart command description for a definition of an active cash-in transaction.</li> <li>"retract": The cash-in transaction ended with a Retract command call.</li> <li>"unknown": The state of the cash-in transaction is unknown. This status is also set if the noteNumberList details are not known or are not reliable.</li> <li>"reset": The cash-in transaction ended with a Reset command call.</li> </ul>
numOfRefused	integer		Specifies the number of items refused during the currently active or most recently ended cash-in transaction period.

Name	Type	Default	Description
noteNumberList	object		<p>List of banknote types that were inserted, identified and accepted during the currently active or most recently ended cash-in transaction period. If items have been rolled back (status is "rollback") they will be included in this list. If status is "retract" or "reset" then identified and accepted items moved to Cash-In or Recycle cash units are included in this list, but items moved to the Retract or Reject cash units are not included.</p> <p>noteNumberList includes any level 2 or level 3 notes, and all level 4 fit and unfit notes.</p>
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.Banknote Types</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
extra	array		TODO

Name	Type	Default	Description
unfitNoteNumberList	object		<p>List of level 4 unfit banknote types that were inserted, identified and accepted during the currently active or most recently ended cash-in transaction period.</p> <p>If items have been rolled back (status is "rollback") they will be included in this list. If status is "retract" or "reset" then identified and accepted items moved to Cash-In units are included in this list, but items moved to the Retract or Reject cash units are not included.</p>
unfitNoteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
unfitNoteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
unfitNoteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "numOfRefused": 0,
    "noteNumberList": {
      "noteNumber": [
        {
          "noteID": 0,
          "count": 0
        }
      ]
    },
    "extra": [
      "string"
    ],
    "unfitNoteNumberList": {
      "noteNumber": [
        {
          "noteID": 0,
          "count": 0
        }
      ]
    }
  }
}
```

## Event Messages

---

## CashAcceptor.PositionCapabilities

---

### Description

This command allows the application to get additional information about the use assigned to each position available in the device.

### Command Message

#### Message Header

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

Message Payload	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload**

Name	Type	Default	Description
posCapabilities	array		Array of position capabilities for all positions configured in this service.

Name	Type	Default	Description
posCapabilities.position	string		<p>Specifies one of the input or output positions as one of the following values:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>
posCapabilities.usage	object		Indicates if an output position is used to reject or rollback.
posCapabilities.usage.in	boolean		It is an input position.
posCapabilities.usage.refuse	boolean		It is an output position used to refuse items.
posCapabilities.usage.rollback	boolean		It is an output position used to rollback items.

Name	Type	Default	Description
posCapabilities.shutterControl	boolean		If set to TRUE the shutter is controlled implicitly by the Service. If set to FALSE the shutter must be controlled explicitly by the application using the OpenShutter and the CloseShutter commands. In either case the CashAcceptor.PresentMedia command may be used if the presentControl field is reported as FALSE. The shutterControl field is always set to TRUE if the described position has no shutter.
posCapabilities.itemsTakenSensor	boolean		Specifies whether or not the described position can detect when items at the exit position are taken by the user. If set to TRUE the Service generates an accompanying CashAcceptor.ItemsTaken event. If set to FALSE this event is not generated. This field relates to output and refused positions.
posCapabilities.itemsInsertedSensor	boolean		Specifies whether the described position has the ability to detect when items have been inserted by the user. If set to TRUE the Service Provider generates an accompanying CashAcceptor.ItemsInserted event. If set to FALSE this event is not generated. This field relates to all input positions.
posCapabilities.retractAreas	object		Specifies the areas to which items may be retracted from this position. If the device does not have a retract capability all values will be FALSE.
posCapabilities.retractAreas.retract	boolean		Items may be retracted to a retract cash unit.
posCapabilities.retractAreas.reject	boolean		Items may be retracted to a reject cash unit.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
posCapabilities.retractAreas.transport	boolean		Items may be retracted to the transport.
posCapabilities.retractAreas.stacker	boolean		Items may be retracted to the intermediate stacker.
posCapabilities.retractAreas.billCassettes	boolean		Items may be retracted to item cassettes, i.e. cash-in and recycle cash units.
posCapabilities.retractAreas.cashIn	boolean		Items may be retracted to a cash-in cash unit.
posCapabilities.presentControl	boolean		Specifies how the presenting of media items is controlled. If presentControl is TRUE then the CashAcceptor.PresentMedia command is not supported and items are moved to the output position for removal as part of the relevant command, e.g. CashAcceptor.CashIn or CashAcceptor.CashInRollback where there is implicit shutter control. If presentControl is FALSE then items returned or rejected can be moved to the output position using the CashAcceptor.PresentMedia command, this includes items returned or rejected as part of a CashAcceptor.CashIn or CashAcceptor.CashInRollback operation. The CashAcceptor.PresentMedia command will open and close the shutter implicitly.

Name	Type	Default	Description
posCapabilities.preparePresent	boolean		<p>Specifies how the presenting of items is controlled. If preparePresent is FALSE then items to be removed are moved to the output position as part of the relevant command e.g. CashAcceptor.OpenShutter or CashAcceptor.PresentMedia or CashAcceptor.CashInRollback. If preparePresent is TRUE then items are moved to the output position using the CashAcceptor.PreparePresent command.</p>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "posCapabilities": [
      {
        "position": "inLeft",
        "usage": {
          "in": true,
          "refuse": true,
          "rollback": true
        },
        "shutterControl": true,
        "itemsTakenSensor": true,
        "itemsInsertedSensor": true,
        "retractAreas": {
          "retract": true,
          "reject": true,
          "transport": true,
          "stacker": true,
          "billCassettes": true,
          "cashIn": true
        },
        "presentControl": true,
        "preparePresent": true
      }
    ]
  }
}
```

#### Event Messages

## CashAcceptor.ReplenishTarget

---

### Description

This command is used to determine which cash units can be specified as target cash units for a given source cash unit with the CashAcceptor.Replenish command. For example it can be used to determine which targets can be used for replenishment from a replenishment container or from a recycle cash unit.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
numberSource	integer		Index number of the logical cash unit which would be used as the source of the replenishment operation. This is the index number identifier defined in the "number" field of the output data of the CashManagement.CashUnitInfo command.

#### Example Message (generated)

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

### Completion Message

#### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
replenishTargets	array		Array of all suitable replenish targets. Empty if no suitable target was found.
replenishTargets.numberTarget	integer		Index number of the logical cash unit that can be used as a target. This is the index number identifier defined in the "number" field of the output data of the CashManagement.CashUnitInfo command.

### Example Message (generated)

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

## Event Messages

---

### CashAcceptor.DevicelockStatus

---

#### Description

This command is used to retrieve the lock/unlock statuses of the CashAcceptor device and each of its cash units. If the physical lock/unlock of both the CashAcceptor device and the cash units are not supported then an error will be returned.

#### Command Message

##### Message Header

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

##### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
deviceLockStatus	string		<p>Specifies the physical lock/unlock status of the CIM device. Following values are possible:</p> <ul style="list-style-type: none"> <li>"lock": The device is physically locked.</li> <li>"unlock": The device is physically unlocked.</li> <li>"lockUnknown": Due to a hardware error or other condition, the physical lock/unlock status of the device cannot be determined.</li> <li>"lockNotSupported": The Service does not support physical lock/unlock control of the device.</li> </ul>
cashUnitLock	array		<p>Array specifying the physical lock/unlock status of cash units. Cash units that do not support the physical lock/unlock control are not contained in the array. If there are no cash units that support physical lock/unlock control this will be empty.</p>
cashUnitLock.physicalPositionName	string		<p>A name identifying the physical location of the cash unit within the CashAcceptor. This name is the same as the "physicalPositionName" in the CashManagement.CashUnitInfo command.</p>
cashUnitLock.cashUnitLockStatus	string		<p>Specifies the physical lock/unlock status of cash units supported. Following values are possible:</p> <ul style="list-style-type: none"> <li>"lock": The cash unit is physically locked.</li> <li>"unlock": The cash unit is physically unlocked.</li> <li>"lockUnknown": Due to a hardware error or other condition, the physical lock/unlock status of the cash unit cannot be determined.</li> </ul>

Name	Type	Default	Description

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "deviceLockStatus": "lock",
    "cashUnitLock": [
      {
        "physicalPositionName": "string",
        "cashUnitLockStatus": "lock"
      }
    ]
  }
}
```

### Event Messages

---

## CashAcceptor.CashUnitCapabilities

---

#### Description

This command is used to retrieve information on cash unit capabilities. It does not provide information on status or counters of cash units. This command can be seen as an extension to the WFS\_INF\_CIM\_CASH\_UNIT\_INFO command as it will always result in the same contents with regard to usNumber and the physical cash unit information.

#### Command Message

##### Message Header

Name	Type	Default	Description

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

### Message Payload

Name	Type	Default	Description
timeoutinteger0	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
cashUnitCaps	array		TODO
cashUnitCaps.number	integer		TODO
cashUnitCaps.physical	array		TODO

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
cashUnitCaps.physical.physicalPositionName	string	TODO	
cashUnitCaps.physical.maximum	integer	TODO	
cashUnitCaps.physical.hardwareSensors	boolean	TODO	
cashUnitCaps.physical.extra	array	TODO	
cashUnitCaps.retractNoteCountThresholds	boolean	TODO	
cashUnitCaps.extra	array	TODO	

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "cashUnitCaps": [
      {
        "number": 0,
        "physical": [
          {
            "physicalPositionName": "string",
            "maximum": 0,
            "hardwareSensors": true,
            "extra": [
              "string"
            ]
          }
        ],
        "retractNoteCountThresholds": true,
        "extra": [
          "string"
        ]
      }
    ]
  }
}
```

#### Event Messages

## CashAcceptor.DepleteSource

#### Description

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This command is used to determine which cash units can be specified as source cash units for a given target cash unit with the CashAcceptor.Deplete command. For example it can be used to determine which sources can be used for depletion to a replenishment container or to a cash-in cash unit.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
numberTarget	integer		Index number of the logical cash unit which would be used as the target of the depletion operation. This is the index number identifier defined in the "number" field of the output data of the CashManagement.CashUnitInfo command.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
depleteSources	array		Array of all suitable deplete sources. Empty if no suitable source was found.
depleteSources.numberSource	integer		Index number of the logical cash unit that can be used as a source. This is the index number identifier defined in the "number" field of the output data of the CashManagement.CashUnitInfo command.

#### Example Message (generated)

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

### Event Messages

---

## CashAcceptor.GetCashUnitCountStatus

---

### Description

During normal processing it is possible that the *count* of a cash unit can become inaccurate due to a jam, mis-pick or other error situation. In this case the GetCashUnitCountStatus command could be used to report which cash units are known to have an inaccurate *count*. The application can then issue a CashAcceptor.CashUnitCount command for only those cash units if supported. Or alternatively the notes could be manually counted as part of a replenishment operation. This command returns the cash unit count status of all cash units.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

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

#### Message Payload

Name	Type	Default	Description
number	integer		Index number of the logical cash unit.

Name	Type	Default	Description
accuracy	string		<p>Describes the accuracy of <i>count</i>. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The hardware is not capable to determine the accuracy of <i>count</i>.</li> <li>"accurate": The <i>count</i> is expected to be accurate. The notes were previously counted or replenished and there have since been no events that might have introduced inaccuracy. This value will be reported as a result of the following commands: Replenish and CashUnitCount.</li> <li>"accurateSet": The <i>count</i> is expected to be accurate. The notes were previously set and there have since been no events that might have introduced inaccuracy.</li> <li>"inaccurate": The <i>count</i> is likely to be inaccurate. A jam, picking fault, or some other event may have resulted in a counting inaccuracy.</li> <li>"unknown": The accuracy of <i>count</i> cannot be determined. This may be due to cash unit insertion or some other hardware event.</li> </ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
physicalCashUnitStatus	array		Array of cash count status objects for physical cash units of this logical cash unit.
physicalCashUnitStatus.physicalPositionName	string		A name identifying the physical location of the cash unit within the CashAcceptor. This field can be used to identify shared cash units/media bins.

Name	Type	Default	Description
physicalCashUnitStatus.accuracy	string		<p>Describes the accuracy of <i>count</i> of a physical cash unit. Following values are possible:</p> <p>"notSupported": The hardware is not capable to determine the accuracy of <i>count</i>.</p> <p>"accurate": The <i>count</i> is expected to be accurate. The notes were previously counted or replenished and there have since been no events that might have introduced inaccuracy. This value will be reported as a result of the following commands: Replenish and CashUnitCount.</p> <p>"accurateSet": The <i>count</i> is expected to be accurate. The notes were previously set and there have since been no events that might have introduced inaccuracy.</p> <p>"inaccurate": The <i>count</i> is likely to be inaccurate. A jam, picking fault, or some other event may have resulted in a counting inaccuracy.</p> <p>"unknown": The accuracy of <i>count</i> cannot be determined. This may be due to cash unit insertion or some other hardware event.</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
physicalCashUnitStatus.extra	array		TODO
extra	array		TODO

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": [
    {
      "number": 0,
      "accuracy": "notSupported",
      "physicalCashUnitStatus": [
        {
          "physicalPositionName": "string",
          "accuracy": "notSupported",
          "extra": [
            "string"
          ]
        }
      ],
      "extra": [
        "string"
      ]
    }
  ]
}
```

### Event Messages

## CashAcceptor.GetPresentStatus

### Description

This command is used to obtain the status of the most recent attempt to present or return items to the customer. This information includes the number of items previously moved to the output position and the number of items which have yet to be returned as a result of the following commands: CashIn, CashInRollback, PreparePresent, PresentMedia, OpenShutter (In the case of returning multiple bunches)

### Command Message

#### Message Header

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
position	string		<p>Specifies the output position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"left": Left output position.</li> <li>"right": Right output position.</li> <li>"center": Center output position.</li> <li>"top": Top output position.</li> <li>"bottom": Bottom output position.</li> <li>"front": Front output position.</li> <li>"rear": Rear output position.</li> </ul>
presentState	string		<p>Supplies the status of the items that were to be presented by the most recent attempt to present or return items to the customer. Following values are possible:</p> <ul style="list-style-type: none"> <li>"presented": The items were presented. This status is set as soon as the customer has access to the items.</li> <li>"notPresented": The customer has not had access to the items.</li> <li>"unknown": It is not known if the customer had access to the items.</li> </ul>
additionalBunches	string		<p>Specifies whether or not additional bunches of items are remaining to be presented as a result of the most recent operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"none": No additional bunches remain.</li> <li>"oneMore": At least one additional bunch remains.</li> <li>"unknown": It is unknown whether additional bunches remain.</li> </ul>

Name	Type	Default	Description
bunchesRemaining	integer		If <i>additionalBunches</i> is "oneMore", specifies the number of additional bunches of items remaining to be presented as a result of the current operation. If the number of additional bunches is at least one, but the precise number is unknown, <i>bunchesRemaining</i> will be 255 (TODO: Check if there is a better way to represent this state). For any other value of <i>additionalBunches</i> , <i>bunchesRemaining</i> will be zero.
returnedItems	object		Array holding a list of banknote numbers which have been moved to the output position as a result of the most recent operation.
returnedItems.noteNumber	array		Array of banknote numbers the cash unit contains.
returnedItems.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.Banknote Types</i> command. If this value is zero then the note type is unknown.
returnedItems.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
totalReturnedItems	object		Array of cumulative banknote numbers which have been moved to the output position. This value will be reset when the CashInStart, CashIn, CashInEnd, Retract, Reset or CashInRollback command is executed.

Name	Type	Default	Description
totalReturnedItems.noteNumber	array		Array of banknote numbers the cash unit contains.
totalReturnedItems.noteNumber.noteIDinteger			Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
totalReturnedItems.noteNumber.count integer			Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
remainingItems	object		Array of banknote numbers on the intermediate stacker or transport which have not been yet moved to the output position.
remainingItems.noteNumber	array		Array of banknote numbers the cash unit contains.
remainingItems.noteNumber.noteID integer			Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
remainingItems.noteNumber.count integer			Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
extra	array		TODO

**Example Message (generated)**

```
{  
    "headers": {  
        "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
        "type": "command",  
        "name": "string"  
    },  
    "payload": {  
        "position": "left",  
        "presentState": "presented",  
        "additionalBunches": "none",  
        "bunchesRemaining": 0,  
        "returnedItems": {  
            "noteNumber": [  
                {  
                    "noteID": 0,  
                    "count": 0  
                }  
            ]  
        },  
        "totalReturnedItems": {  
            "noteNumber": [  
                {  
                    "noteID": 0,  
                    "count": 0  
                }  
            ]  
        },  
        "remainingItems": {  
            "noteNumber": [  
                {  
                    "noteID": 0,  
                    "count": 0  
                }  
            ]  
        },  
        "extra": [  
            "string"  
        ]  
    }  
}
```

## Event Messages

---

### CashAcceptor.CashInStart

---

#### Description

Before initiating a cash-in operation, an application must issue the CashAcceptor.CashInStart command to begin a cash-in transaction. During a cash-in transaction any number of CashAcceptor.CashIn commands may be issued. The transaction is ended when either a CashAcceptor.CashInRollback, CashAcceptor.CashInEnd, CashAcceptor.Retract or CashAcceptor.Reset command is sent. Where Capability *shutterControl* == FALSE this command precedes any explicit operation of the shutters.

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

CashAcceptor.Retract will terminate a transaction. In this case CashAcceptor.CashInEnd, CashAcceptor.CashInRollback and CashAcceptor.CashIn will report *noCashInActive*. If an application wishes to determine where the notes went during a transaction it can execute a CashUnitInfo before and after the transaction and then derive the difference.

A hardware failure during the cash-in transaction does not reset the note number list information; instead the note number list information will include items that could be accepted and identified up to the point of the hardware failure.

**Exchange:** This command can be used during an Exchange (*exchangeType == depositInfo*) to deposit items accepted from the input position.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
tellerID	integer		Identification of teller. This field is not applicable to Self-Service devices and should be omitted.
useRecycleUnits	boolean		Specifies whether or not the recycle cash units should be used when items are cashed in on a successful CashAcceptor.CashInEnd command. This parameter will be ignored if there are no recycle cash units or the hardware does not support this.

Name	Type	Default	Description
outputPosition	string		<p>The output position where the items will be presented to the customer in the case of a rollback. Following values are possible:</p> <p>"null": The items will be presented to the default configuration.</p> <p>"left": The items will be presented to the left output position.</p> <p>"right": The items will be presented to the right output position.</p> <p>"center": The items will be presented to the center output position.</p> <p>"top": The items will be presented to the top output position.</p> <p>"bottom": The items will be presented to the bottom output position.</p> <p>"front": The items will be presented to the front output position.</p> <p>"rear": The items will be presented to the rear output position.</p>

Name	Type	Default	Description
inputPosition	string		<p>Specifies from which position the cash should be inserted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": The cash is inserted from the default configuration.</li> <li>"left": The cash is inserted from the left input position.</li> <li>"right": The cash is inserted from the right input position.</li> <li>"center": The cash is inserted from the center input position.</li> <li>"top": The cash is inserted from the top input position.</li> <li>"bottom": The cash is inserted from the bottom input position.</li> <li>"front": The cash is inserted from the front input position.</li> <li>"rear": The cash is inserted from the rear input position.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "tellerID": 0,
    "useRecycleUnits": true,
    "outputPosition": null,
    "inputPosition": null
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
------	------	---------	-------------

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

---

## CashAcceptor.CashIn

---

### Description

This command moves items into the cash device from an input position.

On devices with implicit shutter control, the CashAcceptor.InsertItems event will be generated when the device is ready to start accepting media.

The items may pass through the banknote reader for identification. Failure to identify items does not mean that the command has failed - even if some or all of the items are rejected by the banknote reader, the command may return *success*. In this case one or more CashAcceptor.InputRefuse events will be sent to report the rejection. See also paragraph below regarding returning refused items.

If the device does not have a banknote reader then the completion message will be empty.

If the device has a cash-in stacker then this command will cause inserted level 4 items to be moved there after validation. Level 2 and level 3 items may also be moved to the cash-in stacker, but some

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
devices may immediately move them to a designated cash unit. Items on the stacker will remain there until the current cash-in transaction is either cancelled by the CashAcceptor.CashInRollback command or confirmed by the CashAcceptor.CashInEnd command. These commands will cause any level 2 or level 3 items on the cash-in stacker to be moved to the appropriate cash unit. If there is no cash-in stacker then this command will move items directly to the cash units and the CashAcceptor.CashInRollback command will not be supported. Cash unit information will be updated accordingly whenever notes are moved to a cash unit during this command.

Note that the *acceptor* status field may change value during a cash-in transaction. If media has been retained to cash units during a cash-in transaction, it may mean that *acceptor* is set to *stop*, which means subsequent cash-in operations may not be possible. In this case, the subsequent command fails with error code *CashUnitError*.

The *shutterControl* field of the capabilities structure returned from the Common.Capabilities query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly open and close the shutter using the OpenShutter and CloseShutter commands, or the CashAcceptor.PresentMedia command. If *shutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *shutterControl* is TRUE this command opens the shutter at the start of the command and closes it once bills are inserted.

The *presentControl* field of the positionCapabilities structure returned from the CashAcceptor.PositionCapabilities query will determine whether or not it is necessary to call the CashAcceptor.PresentMedia command in order to move items to the output position. If *presentControl* is TRUE then all items are moved immediately to the correct output position for removal (a OpenShutter command will be needed in the case of explicit shutter control). If *presentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the CashAcceptor.PresentMedia command.

It is possible that a device may divide bill or coin accepting into a series of sub-operations under hardware control. In this case a CashAcceptor.SubCashIn event may be sent after each sub-operation, if the hardware capabilities allow it.

#### **Returning items (single bunch):**

If *shutterControl* is TRUE, and a single bunch of items is returned then this command will complete once the notes have been returned. A CashAcceptor.ItemsPresented event will be generated.

If *shutterControl* is FALSE, and a single bunch of items is returned then this command will complete without generating a CashAcceptor.ItemsPresented event, instead the CashAcceptor.ItemsPresented event will be generated by the subsequent OpenShutter or CashAcceptor.PresentMedia command.

#### **Returning items (multiple bunches):**

It is possible that a device will in certain situations return refused items in multiple bunches. In this case, this command will not complete until the final bunch has been presented and after the last CashAcceptor.ItemsPresented event has been generated. For these devices *shutterControl* and *presentControl* fields of the Capabilities / positionCapabilities structure returned from the Common.Capabilities / CashAcceptor.PositionCapabilities query must both be TRUE otherwise it will not be possible to return multiple bunches. Additionally it may be possible to request the completion of this command with a Cancel before the final bunch is presented so that after the completion of this command the CashAcceptor.Retract or CashAcceptor.Reset command can be used to move the remaining bunches, although the ability to do this will be hardware dependent.

**Mixed Media Mode:** If the device is operating in Mixed Media mode (Status *mixedMode* == *mixedMedia*) the Service Provider will not perform any operation unless the ItemProcessor.MediaIn command is called or has already been called on the ItemProcessor interface.

**Exchange:** This command can be used during an Exchange (*exchangeType* == *depositInto*) to accept items from the input position.

## **Command Message**

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

#### Message Payload

Name	Type	Default	Description
noteNumber	array		Array of banknote numbers the cash unit contains.

Name	Type	Default	Description
noteNumber.noteIDinteger			Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumber.count integer			Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.

**Example Message (generated)**

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

**Event Messages**

- [CashManagement.CashUnitErrorEvent](#)
- [CashAcceptor.InputRefuseEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashAcceptor.SubCashInEvent](#)
- [CashAcceptor.ItemsInsertedEvent](#)
- [CashAcceptor.ItemsTakenEvent](#)
- [CashAcceptor.ItemsPresentedEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashAcceptor.InsertItemsEvent](#)
- [CashManagement.CashUnitThresholdEvent](#)
- [CashAcceptor.ShutterStatusChangedEvent](#)

**CashAcceptor.CashInEnd****Description**

This command ends a cash-in transaction. If cash items are on the stacker as a result of a *CashAcceptor.CashIn* command these items are moved to the appropriate cash units.

The cash-in transaction is ended even if this command does not complete successfully.

**Mixed Media Mode:**

### XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

If the device is operating in Mixed Media mode (Status *mixedMode* == mixedMedia) non-cash items, e.g. checks may be moved to an output position or media bin specified by the ItemProcessor interface. Additionally, the Service will not perform any operation unless the ItemProcessor.MedialnEnd command is called or has already been called on the ItemProcessor. Alternatively, if Capabilities *mixedDepositAndRollback* is TRUE, then the ItemProcessor.MedialnRollback command could be used instead of the ItemProcessor.MedialnEnd command in order to deposit the bills and return the checks.

Where ItemProcessor items may be presented the *presentControl* field of the positionCapabilities structure returned from the CashAcceptor.PositionCapabilities query will determine whether or not it is necessary to call the CashAcceptor.PresentMedia command in order to move items to the output position. If *presentControl* is TRUE then all items are moved immediately to the correct output position for removal. If *presentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the CashAcceptor.PresentMedia command.

**Exchange:** This command can be used during an Exchange (*exchangeType* == *depositInto*) to deposit items accepted from the input position.

## Command Message

### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
list	array		Array of cash unit objects.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "list": [
      {
        "number": 0,
        "type": "notApplicable",
        "unitID": "string",
        "currencyID": "string",
        "values": 0,
        "count": 0,
        "maximum": 0,
        "status": "ok",
        "appLock": true,
        "physical": [
          {
            "physicalPositionName": "string",
            "unitID": "string",
            "count": 0,
            "maximum": 0,
            "pStatus": "ok",
            "hardwareSensor": true,
            "initialCount": 0,
            "dispensedCount": 0,
            "presentedCount": 0,
            "retractedCount": 0,
            "rejectCount": 0,
            "cashInCount": 0,
            "extra": [
              "string"
            ]
          }
        ],
        "cashUnitName": "string",
        "cashUnitType": "string"
      }
    ]
  }
}
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "minimum": 0,
        "itemType": {
            "all": true,
            "unfit": true,
            "individual": true,
            "level1": true,
            "level2": true,
            "level3": true,
            "itemProcessor": true,
            "unfitIndividual": true
        },
        "cashInCount": 0,
        "noteNumberList": {
            "noteNumber": [
                {
                    "noteID": 0,
                    "count": 0
                }
            ]
        },
        "noteIDs": [
            0
        ],
        "extra": [
            "string"
        ]
    }
}
```

## Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitInfoChangedEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashAcceptor.ItemsTakenEvent](#)
- [CashAcceptor.ItemsPresentedEvent](#)
- [CashManagement.CountsChangedEvent](#)
- [CashAcceptor.ShutterStatusChangedEvent](#)

---

## CashAcceptor.CashInRollback

---

### Description

This command is used to roll back a cash-in transaction. It causes all the cash items cashed in since the last CashAcceptor.CashInStart command to be returned to the customer.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This command ends the current cash-in transaction. The cash-in transaction is ended even if this command does not complete successfully.

The *shutterControl* field of the Capabilities structure returned from the Common.Capabilities query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the OpenShutter and CloseShutter commands, or CashAcceptor.PresentMedia command. If *shutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *shutterControl* is TRUE then this command opens the shutter and it is closed when all items are removed.

The *presentControl* field of the positionCapabilities structure returned from the CashAcceptor.PositionCapabilities query will determine whether or not it is necessary to call the CashAcceptor.PresentMedia command in order to move items to the output position. If *presentControl* is TRUE then all items are moved immediately to the correct output position for removal (a OpenShutter command will be needed in the case of explicit shutter control). If *presentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the CashAcceptor.PresentMedia command.

Items are returned in a single bunch or multiple bunches in the same way as described for the CashAcceptor.CashIn command.

**Mixed Media Mode:** If the device is operating in Mixed Media mode (Status *mixedMode* == mixedMedia) the Service will not perform any operation unless the ItemProcesspr.MediaInRollback command is called or has already been called on the ItemProcesspr interface. Alternatively, if the Capabilities *mixedDepositAndRollback* is TRUE, then the ItemProcessor.MediaInEnd command could be used instead of the ItemProcessor.MediaInRollback command in order to deposit the checks and return the items.

**Exchange:** This command can be used during an Exchange (*exchangeType* == depositInto) to return items accepted from the input position. Note that *ExchangeActive* would not be generated in this case.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
list	array		Array of cash unit objects.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "list": [
      {
        "number": 0,
        "type": "notApplicable",
        "unitID": "string",
        "currencyID": "string",
        "values": 0,
        "count": 0,
        "maximum": 0,
        "status": "ok",
        "appLock": true,
        "physical": [
          {
            "physicalPositionName": "string",
            "unitID": "string",
            "count": 0,
            "maximum": 0
          }
        ]
      }
    ]
  }
}
```

```
        "maximum": 0,
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    },
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
]
}
```

## Event Messages

- [CashManagement.CashUnitErrorEvent](#)
- [CashAcceptor.ItemsTakenEvent](#)
- [CashAcceptor.ItemsPresentedEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashManagement.CashUnitThresholdEvent](#)

- [CashManagement.CountsChangedEvent](#)
  - [CashAcceptor.ShutterStatusChangedEvent](#)
- 

## CashAcceptor.Retract

---

### Description

This command retracts items from an output position or internal areas within the device. Retracted items will be moved to either a retract bin, a reject bin, cash-in/recycle cash units, the transport or an intermediate stacker area. If items from internal areas within the device are preventing items at an output position from being retracted then the items from the internal areas will be retracted first. When the items are retracted from an output position the shutter is closed automatically, even if the *shutterControl* capability is set to FALSE.

This command terminates a running cash-in transaction. The cash-in transaction is terminated even if this command does not complete successfully.

#### Mixed Media Mode:

If the device is operating in Mixed Media mode (Status *mixedMode == mixedMedia*) this command will not perform any operation unless the ItemProcessor.RetractMedia command is called or has already been called on the ItemProcessor interface. Where the parameters for this command and the corresponding ItemProcessor.RetractMedia command conflict, for example the device is physically unable to satisfy both commands, the CashAcceptor.Retract input parameters will be used for all items.

**Exchange:** This command can be used during an Exchange (*exchangeType == depositInto*) to retract items. Note that an ExchangeActive error would not be generated in this case.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
outputPosition	string		<p>Specifies the output position from which to retract the bills. Following values are possible:</p> <ul style="list-style-type: none"><li>"null": The default configuration information should be used. This value is also used to retract items from internal device locations.</li><li>"left": Retract items from the left output position.</li><li>"right": Retract items from the right output position.</li><li>"center": Retract items from the center output position.</li><li>"top": Retract items from the top output position.</li><li>"bottom": Retract items from the bottom output position.</li><li>"front": Retract items from the front output position.</li><li>"rear": Retract items from the rear output position.</li></ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"billCassettes": Retract the items to item cassettes, i.e. cash-in and recycle cash units.</li> <li>"cashIn": Retract the items to a cash-in cash unit. The <i>itemType</i> of the cash-in cash unit defined in <i>cashInfo</i> must include ("all"   "unfit").</li> </ul>

Name	Type	Default	Description
index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. The maximum value of <i>index</i> is the sum of the <i>maximum</i> of each retract cash unit.</p> <p>If <i>retractArea</i> is set to "cashIn" this field defines the physical cash unit under the "cashIn" cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) and would be incremented from the first physical cash unit to the last physical cash unit defined in <i>cashInfo</i>.</p> <p>If <i>retractArea</i> is not set to "retract" or "cashIn" then the value of this field is ignored.</p>

#### Example Message (generated)

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

#### Completion Message

##### Message Header

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
list	array		Array of cash unit objects.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "list": [
      {
        "number": 0,
        "type": "notApplicable",
        "unitID": "string",
        "currencyID": "string",
        "values": 0,
        "count": 0,
        "maximum": 0,
        "status": "ok",
        "appLock": true,
        "physical": [
          {
            "physicalPositionName": "string",
            "unitID": "string",
            "count": 0,
            "maximum": 0,
            "pStatus": "ok",
            "hardwareSensor": true,
            "initialCount": 0,
            "dispensedCount": 0,
            "presentedCount": 0,
            "retractedCount": 0,
            "rejectCount": 0,
            "cashInCount": 0,
            "extra": [
              "string"
            ]
          }
        ],
        "cashUnitName": "string",
        "cashUnitType": "string"
      }
    ]
  }
}
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "minimum": 0,
        "itemType": {
            "all": true,
            "unfit": true,
            "individual": true,
            "level1": true,
            "level2": true,
            "level3": true,
            "itemProcessor": true,
            "unfitIndividual": true
        },
        "cashInCount": 0,
        "noteNumberList": {
            "noteNumber": [
                {
                    "noteID": 0,
                    "count": 0
                }
            ]
        },
        "noteIDs": [
            0
        ],
        "extra": [
            "string"
        ]
    }
}
```

## Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashAcceptor.ItemsTakenEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashManagement.CashUnitInfoChangedEvent](#)
- [CashAcceptor.ShutterStatusChangedEvent](#)

---

## CashAcceptor.OpenShutter

---

### Description

This command opens the shutter. In cases where multiple bunches are to be returned under explicit shutter control and the first bunch has already been presented and taken and the output position is empty, this command moves the next bunch to the output position before opening the shutter -- see sections 8.6 and 8.7. This does not apply if the output position is not empty, for example if items had been re-inserted or dropped back into the output position as the shutter closed.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
position	string		<p>Position where the shutter is to be opened. If the application does not need to specify the shutter, this field can be omitted or set to "null". Otherwise this field should be set to one of the following values:</p> <ul style="list-style-type: none"> <li>"null": The default configuration information should be used.</li> <li>"inLeft": Open the shutter of the left input position.</li> <li>"inRight": Open the shutter of the right input position.</li> <li>"inCenter": Open the shutter of the center input position.</li> <li>"inTop": Open the shutter of the top input position.</li> <li>"inBottom": Open the shutter of the bottom input position.</li> <li>"inFront": Open the shutter of the front input position.</li> <li>"inRear": Open the shutter of the rear input position.</li> </ul>

Name	Type	Default	Description
		"outLeft":	shutter of the left output position.
		"outRight":	Open the shutter of the right output position.
		"outCenter":	Open the shutter of the center output position.
		"outTop":	Open the shutter of the top output position.
		"outBottom":	Open the shutter of the bottom output position.
		"outFront":	Open the shutter of the front output position.
		"outRear":	Open the shutter of the rear output position.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

#### Event Messages

- [CashAcceptor.ItemsTakenEvent](#)
- [CashAcceptor.ItemsInsertedEvent](#)
- [CashAcceptor.ShutterStatusChangedEvent](#)

## CashAcceptor.CloseShutter

#### Description

This command closes the shutter.

#### Command Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
			Position where the

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
position	string		<p>shutter <del>destroyed</del>. If the application does not need to specify the shutter, this field can be omitted or set to "null". Otherwise this field should be set to one of the following values:</p> <ul style="list-style-type: none"> <li>"null": The default configuration information should be used.</li> <li>"inLeft": Close the shutter of the left input position.</li> <li>"inRight": Close the shutter of the right input position.</li> <li>"inCenter": Close the shutter of the center input position.</li> <li>"inTop": Close the shutter of the top input position.</li> <li>"inBottom": Close the shutter of the bottom input position.</li> <li>"inFront": Close the shutter of the front input position.</li> <li>"inRear": Close the shutter of the rear input position.</li> <li>"outLeft": Close the shutter of the left output position.</li> <li>"outRight": Close the shutter of the right output position.</li> <li>"outCenter": Close the shutter of the center output position.</li> <li>"outTop": Close the shutter of the top output position.</li> <li>"outBottom": Close the shutter of the bottom output position.</li> <li>"outFront": Close the shutter of the front</li> </ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
			"outRear": Close the shutter of the rear output position.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

## Event Messages

- [CashAcceptor.ShutterStatusChangedEvent](#)
- 

## CashAcceptor.Reset

---

### Description

This command is used by the application to perform a hardware reset which will attempt to return the device to a known good state. This command does not over-ride a lock obtained on another application or service handle.

If a cash-in transaction is active, this command will end it (even if this command does not complete successfully). If an exchange state is active then this command will end the exchange state (even if this command does not complete successfully).

Persistent values, such as counts and configuration information are not cleared by this command.

The device will attempt to move any items found anywhere within the device to the position specified within the *resetIn* parameter. This may not always be possible because of hardware problems.

If items are found inside the device one or more CashAcceptor.MediaDetected events will be generated to inform the application where the items have actually been moved to.

The *shutterControl* field of the Capabilities structure returned from the Common.Capabilities query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the OpenShutter and CloseShutter commands, or the CashAcceptor.PresentMedia command. If *shutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *shutterControl* is TRUE then this command operates the shutter as necessary so that the shutter is closed after the command completes successfully and any items returned to the customer have been removed.

The *presentControl* field of the *positionCapabilities* structure returned from the CashAcceptor.PositionCapabilities query will determine whether or not it is necessary to call the CashAcceptor.PresentMedia command in order to move items to the output position. If *presentControl* is TRUE then all items are moved immediately to the correct output position for removal (a OpenShutter command will be needed in the case of explicit shutter control). If *presentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the CashAcceptor.PresentMedia command.

If requested, items are returned in a single bunch or multiple bunches in the same way as described for the CashAcceptor.CashIn command.

### Mixed Media Mode:

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

The value of Status *mixedMode* is not changed by this command. Where the items are to be moved to a cash unit, the cash unit must support an *itemType* of "itemProcessor".

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found. If items are to be moved to an output position, this value must be zero, <i>retractArea</i> must be omitted and <i>outputPosition</i> specifies where items are to be moved to. If this value is zero and items are to be moved to internal areas of the device, <i>retractArea</i> specifies where items are to be moved to or stored.
retractArea	object		This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker or the transport. The field is only relevant if <i>number</i> is zero.

Name	Type	Default	Description
retractArea.outputPosition	string		<p>Specifies the output position from which to retract the bills. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": The default configuration information should be used. This value is also used to retract items from internal device locations.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul> <p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"billCassettes": Retract the items to item cassettes, i.e. cash-in and recycle cash units.</li> <li>"cashIn": Retract the items to a cash-in cash unit. The <i>itemType</i> of the cash-in cash unit defined in <i>cashInfo</i> must include ("all"   "unfit").</li> </ul>
retractArea.retractArea	string		

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
retractArea.index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. The maximum value of <i>index</i> is the sum of the <i>maximum</i> of each retract cash unit.</p> <p>If <i>retractArea</i> is set to "cashIn" this field defines the physical cash unit under the "cashIn" cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) and would be incremented from the first physical cash unit to the last physical cash unit defined in <i>cashInfo</i>.</p> <p>If <i>retractArea</i> is not set to "retract" or "cashIn" then the value of this field is ignored.</p>
outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>nretractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": Take the default configuration.</li> <li>"left": Move items to the left output position.</li> <li>"right": Move items to the right output position.</li> <li>"center": Move items to the center output position.</li> <li>"top": Move items to the top output position.</li> <li>"bottom": Move items to the bottom output position.</li> <li>"front": Move items to the front output position.</li> <li>"rear": Move items to the rear output position.</li> </ul>

---

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "retractArea": {
      "outputPosition": null,
      "retractArea": "retract",
      "index": 0
    },
    "outputPosition": null
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

- [CashManagement.CashUnitThresholdEvent](#)
  - [CashAcCashManagementceptor.CashUnitErrorEvent](#)
  - [CashAcceptor.MediaDetectedEvent](#)
  - [CashAcceptor.ItemsTakenEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
  - [CashAcceptor.ShutterStatusChangedEvent](#)
- 

## CashAcceptor.ConfigureNotetypes

---

### Description

This command is used to configure the note types the banknote reader should accept during cash-in. All note types the banknote reader should accept must be given in the input structure. If an unknown note type is given the error code "unsupportedData" will be returned. The values set by this command are persistent.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
noteIDs	array		Array of unsigned integers which contains the note IDs of the banknotes the banknote reader can accept.

#### Example Message (generated)

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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

### CashAcceptor.CreateP6Signature

---

#### Description

This command is used to create a reference signature (normally a level 3 note) that was checked and regarded as a forgery. The reference can be compared with the available signatures of the cash-in transactions to track back the customer.

When this command is executed, the device waits for a note to be inserted at the input position, transports the note to the recognition module, creates the signature and then returns the note to the output position.

The *shutterControl* field of the *capabilities* structure returned from the Common.Capabilities query will

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
 determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the OpenShutter and CloseShutter commands, or CashAcceptor.PresentMedia command. If *shutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *shutterControl* is TRUE then this command opens and closes the shutter at various times during the command execution and the shutter is finally closed when all items are removed.

The *presentControl* field of the *positionCapabilities* structure returned from the CashAcceptor.PositionCapabilities query will determine whether or not it is necessary to call the CashAcceptor.PresentMedia command in order to move items to the output position. If *presentControl* is TRUE then all items are moved immediately to the correct output position for removal (a OpenShutter command will be needed in the case of explicit shutter control). If *presentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the CashAcceptor.PresentMedia command.

On devices with implicit shutter control, the InsertItems event will be generated when the device is ready to start accepting media.

The application may have to execute this command repeatedly to make sure that all possible signatures are captured.

If a single note is entered and returned to the customer but cannot be processed fully (e.g. no recognition software in the recognition module, the note is not recognized, etc.) then a InputRefuse event will be sent and the command will complete. In this case, the output parameters will be set as follows, *noteId* = zero, *length* = zero, *orientation* = "unknown" and *signature* = NULL.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
noteld	integer		Identification of note type.
orientation	object		Orientation of the entered banknote.
orientation.frontTop	boolean		If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the left edge was inserted first.
orientation.frontBottom	boolean		If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the right edge was inserted first.
orientation.backTop	boolean		If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the left edge was inserted first.

Name	Type	Default	Description
orientation.backBottom	boolean		If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the right edge was inserted first.
orientation.unknown	boolean		The orientation for the inserted note can not be determined.
orientation.notSupported	boolean		The hardware is not capable to determine the orientation.
signature	string		Base64 encoded signature data.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "noteId": 0,
    "orientation": {
      "frontTop": true,
      "frontBottom": true,
      "backTop": true,
      "backBottom": true,
      "unknown": true,
      "notSupported": true
    },
    "signature": "string"
  }
}
```

**Event Messages**

- [CashAcceptor.InputRefuseEvent](#)
- [CashAcceptor.ItemsInsertedEvent](#)
- [CashAcceptor.ItemsTakenEvent](#)
- [CashAcceptor.ItemsPresentedEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashAcceptor.InsertItemsEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashAcceptor.ShutterStatusChangedEvent](#)

**CashAcceptor.ConfigureNoteReader****Description**

This command is used to configure the currency description configuration data into the banknote

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
 reader module. The format and location of the configuration data is vendor and/or hardware dependent.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
loadAlways	boolean		If set to TRUE, the Service loads the currency description data into the note reader, even if it is already loaded.

### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
rebootNecessary	boolean		If set to TRUE, the machine needs a reboot before the note reader can be accessed again.

#### Example Message (generated)

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

#### Event Messages

---

## CashAcceptor.CompareP6Signature

---

#### Description

This command is used to compare the signatures of a reference banknote with the available signatures of the cash-in transactions.

The reference signatures are created by the CashAcceptor.CreateP6Signature command.

The transaction signatures are obtained through the CashAcceptor.GetP6Signature command.

The signatures (1 to 4) of the reference banknote are typically the signatures of the 4 orientations of the banknote.

The CashAcceptor.CompareP6Signature command may return a single indication or a list of indications to the matching signatures, each one associated to a confidence level factor. If the Service Provider does not support the confidence level factor, it returns a single indication to the best matching signature with the confidence level factor set to zero.

If the comparison completed with no matching signatures found then the command returns "ok" with p6SignaturesIndex empty.

This command must be used outside of the cash-in transactions and outside of exchange states.

#### Command Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
p6ReferenceSignatures	array		Array of P6Signature structures. Each structure represents the signature corresponding to one orientation of a single reference banknote. At least one orientation must be provided. If no orientations are provided (this array is missing or empty) the command returns an invalidData error.
p6ReferenceSignatures.noteId	integer		Identification of note type.
p6ReferenceSignatures.orientation	object		Orientation of the entered banknote.
p6ReferenceSignatures.orientation.frontTop	boolean		If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the left edge was inserted first.
p6ReferenceSignatures.orientation.frontBottom	boolean		If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the right edge was inserted first.
p6ReferenceSignatures.orientation.backTop	boolean		If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the left edge was inserted first.

Name	Type	Default	Description
p6ReferenceSignatures.orientation.backBottom	boolean		If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the right edge was inserted first.
p6ReferenceSignatures.orientation.unknown	boolean		The orientation for the inserted note can not be determined.
p6ReferenceSignatures.orientation.notSupported	boolean		The hardware is not capable to determine the orientation.
p6ReferenceSignatures.signature	string		Base64 encoded signature data.
p6Signatures	array		Array of P6Signature structures. Each structure represents a level 2/3 signature, from the cash-in transactions, to be compared with the reference signatures in <i>p6ReferenceSignature</i> . At least one signature must be provided. If there are no signatures provided (this array is missing or empty) the command returns an invalidData error.
p6Signatures.notId	integer		Identification of note type.
p6Signatures.orientation	object		Orientation of the entered banknote.
p6Signatures.orientation.frontTop	boolean		If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the left edge was inserted first.

Name	Type	Default	Description
p6Signatures.orientation.frontBottom	boolean		If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the right edge was inserted first.
p6Signatures.orientation.backTop	boolean		If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the left edge was inserted first.
p6Signatures.orientation.backBottom	boolean		If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the right edge was inserted first.
p6Signatures.orientation.unknown	boolean		The orientation for the inserted note can not be determined.
p6Signatures.orientation.notSupported	boolean		The hardware is not capable to determine the orientation.
p6Signatures.signature	string		Base64 encoded signature data.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "p6ReferenceSignatures": [
      {
        "noteId": 0,
        "orientation": {
          "frontTop": true,
          "frontBottom": true,
          "backTop": true,
          "backBottom": true,
          "unknown": true,
          "notSupported": true
        },
        "signature": "string"
      }
    ],
    "p6Signatures": [
      {
        "noteId": 0,
        "orientation": {
          "frontTop": true,
          "frontBottom": true,
          "backTop": true,
          "backBottom": true,
          "unknown": true,
          "notSupported": true
        },
        "signature": "string"
      }
    ]
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
p6SignaturesIndex	array		Array of compare results. This array is empty when the compare operation completes with no match found. If there are matches found, <i>p6SignaturesIndex</i> contains the indexes of the matching signatures from the input parameter <i>p6Signatures</i> . If there is a match found but the Service does not support the confidence level factor, <i>p6SignaturesIndex</i> contains a single index with confidenceLevel set to zero.
p6SignaturesIndex.index	integer		Specifies the index (zero to # <i>p6Signatures</i> - 1) of the matching signature from the input parameter <i>p6Signatures</i> .
p6SignaturesIndex.confidenceLevel	integer		Specifies the level of confidence for the match found. This value is in a scale 1 - 100, where 100 is the maximum confidence level. This value is zero if the Service does not support the confidence level factor.
p6SignaturesIndex.comparisonData	string		Vendor dependent comparison result data. This data may be used as justification for the signature match or confidence level. This field is omitted if no additional comparison data is returned.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "p6SignaturesIndex": [
      {
        "index": 0,
        "confidenceLevel": 0,
        "comparisonData": "string"
      }
    ]
  }
}
```

#### Event Messages

---

## CashAcceptor.Replenish

---

#### Description

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This command replenishes items from a single cash unit to multiple cash units. Applications can use this command to ensure that there is the optimum number of items in the cassettes by moving items from a source cash unit to a target cash unit. This is especially applicable if a replenishment cash unit is used for the replenishment and can help to minimize manual replenishment operations.

The CashAcceptor.ReplenishTarget command can be used to determine what cash units can be specified as target cash units for a given source cash unit. Any items which are removed from the source cash unit that are not of the correct currency ID and value for the target cash unit during execution of this command will be returned to the source cash unit.

The *count*, *cashInCount*, *dispensedCount* and *rejectCount* returned with the CashManagement.CashUnitInfo command will be updated as part of the execution of this command.

If the command fails after some items have been moved, the command will complete with an appropriate error code, and a CashAcceptor.IncompleteReplenishEvent will be sent.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
numberSource	integer		Index number of the cash unit from which items are to be removed. This is the index number identifier defined in the <i>number</i> field of the output data of the CashManagement.CashUnitInfo command.
replenishTargets	array		Array of replenish Target elements. There must be at least one array element.
replenishTargets.numberTarget	integer		Index number of the cash unit to which items are to be moved. This is the index number identifier defined in the <i>number</i> field of the output data of the CashManagement.CashUnitInfo command.

Name	Type	Default	Description
replenishTargets.numberOfItemsToMove	integer		The number of items to be moved to the target cash unit. Any items which are removed from the source cash unit that are not of the correct currency ID and value for the target cash unit during execution of this command will be returned to the source cash unit. This field will be ignored if the <i>removeAll</i> parameter is set to TRUE.
replenishTargets.removeAll	boolean		Specifies if all items are to be moved to the target cash unit. Any items which are removed from the source cash unit that are not of the correct currency ID and value for the target cash unit during execution of this command will be returned to the source cash unit. If TRUE all items in the source will be moved, regardless of the <i>numberOfItemsToMove</i> field value. If FALSE the number of items specified with <i>numberOfItemsToMove</i> will be moved.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "numberSource": 0,
    "replenishTargets": [
      {
        "numberTarget": 0,
        "numberOfItemsToMove": 0,
        "removeAll": true
      }
    ]
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	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
numberOfItemsRemoved	integer		Total number of items removed from the source cash unit including rejected items during execution of this command.
numberOfItemsRejected	integer		Total number of items rejected during execution of this command.
replenishTargetResults	array		Array of replenishTargetResult structures. In the case where one note type has several releases and these are moved, or where items are moved from a multi denomination cash unit to a multi denomination cash unit, each target can receive several <i>noteID</i> note types. For example: If one single target was specified with the <i>replenishTargets</i> input structure, and this target received two different <i>noteID</i> note types, then the <i>replenishTargetResults</i> array will have two elements. Or if two targets were specified and the first target received two different <i>noteID</i> note types and the second target received three different <i>noteID</i> note types, then the <i>replenishTargetResults</i> array will have five elements.
replenishTargetResults.numberTarget	integer		Index number of the cash unit to which items have been moved. This is the index number identifier defined in the <i>number</i> field of the output data of the CashManagement.CashUnitInfo command.
replenishTargetResults.noteID	integer		Identification of note type. The note ID represents the note identifiers reported by the CashAcceptor.BanknoteTypes command.

Name	Type	Default	Description
replenishTargetResults.numberOfItemsReceived	integer		Total number of items received in this target cash unit of the <i>noteID</i> note type. A zero value will be returned if this target cash unit did not receive any items of this note type, for example due to a cash unit or transport jam.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "numberOfItemsRemoved": 0,
    "numberOfItemsRejected": 0,
    "replenishTargetResults": [
      {
        "noteTarget": 0,
        "noteID": 0,
        "numberOfItemsReceived": 0
      }
    ]
  }
}
```

#### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashAcceptor.IncompleteReplenishEvent](#)

## CashAcceptor.SetCashInLimit

#### Description

This command specifies the amount/number of items limitation for the current cash-in transaction. This command can only be called after the CashAcceptor.CashInStart command and before the first CashAcceptor.CashIn command, otherwise it will fail with the SequenceError error. Any command that completes the cash-in transaction (i.e. CashAcceptor.CashInEnd, CashAcceptor.CashInRollback, CashAcceptor.Retract and CashAcceptor.Reset commands) will clear the limit.

This limit is active until the end of the current cash-in transaction. The use of this command is optional, however it needs to be called for each cash-in transaction that needs a limitation.

This command does not disable/enable the recognition of individual note types. The CashAcceptor.ConfigureNotetypes command must be used to refuse a certain note type during cash-

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
in transactions.

If "limitMultiple" is specified in the *cashInLimit* capability, the command may be called multiple times to add to or override amount limits placed on the current cash-in transaction; the input parameter descriptions below define whether limits are added or overridden. If "limitMultiple" is not specified, this command can only be called once per cash-in transaction otherwise it will fail with the SequenceError error.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
totalItemsLimit	integer		If set to a non-zero value, specifies a limit on the total number of items to be accepted during the cash-in transaction. If set to a zero value, this limitation will not be performed. This limitation can only be used if "limitByTotalItems" is specified in the <i>cashInLimit</i> field of the Common.Capabilities command. If <i>totalItemsLimit</i> is non-zero but not supported the UnsupportedData error will be returned and no limit will be set. This parameter overrides any previously set limit on the total number of items.
amountLimit	object		Array of amountLimit structures. This limitation can only be used if "limitByAmount" is reported in the <i>cashInLimit</i> field of the Common.Capabilities command. If <i>amountLimit</i> is not empty but not supported the UnsupportedData error will be returned and no limit will be set. If <i>amountLimit</i> is empty or omitted, this has no impact. If <i>amountLimit</i> is not empty, this specifies the maximum amount of the currency specified by <i>currencyID</i> which can be accepted in the current cash-in transaction. If the currency has already been specified for the current cash-in transaction, the maximum amount is overridden for that currency. If the currency has not already been specified, it is added to a set of currency specific limits to apply to the cash-in transaction. If any currency limits are specified for the current cash-in transaction, the handling of other currencies is dependent on whether the "refuseOther" flag is reported in the <i>cashInLimit</i> field of the Common.Capabilities command.
amountLimit.currencyID	string		Currency identifier in ISO 4217 format [Ref. 2]. This must not be three ASCII 0x20 characters.

Name	Type	Default	Description
amountLimit.amount	number		If set to a non-zero value, specifies a limit on the total amount of the cash-in transaction for the specified cCurrencyID. If set to a zero value, no amount limit will apply to the specified currency.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

## Event Messages

---

### CashAcceptor.CashUnitCount

---

#### Description

This command counts the items in the cash unit(s). If it is necessary to move items internally to count them, the items should be returned to the cash unit from which they originated before completion of the command. If items could not be moved back to the cash unit they originated from and did not get rejected, the command will complete with an appropriate error.

During the execution of this command one CashManagement.CashUnitInfoChangedEvent will be generated for each cash unit that has been counted successfully, or if the counts have changed, even if the overall command fails.

After completion of this command the number of items rejected can be determined by calling the CashManagement.CashUnitInfo command and checking the value of the *rejectCount* field within the output structure. The *rejectCount* value is incremented by one for each item rejected during execution of this command.

This command is designed to be used on devices where the *count* cannot be guaranteed to be accurate and therefore may need to be automatically counted periodically. Upon successful completion, for those cash units that have been counted, the *count* field is accurately reported with the CashManagement.CashUnitInfo command.

#### Command Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
------	------	---------	-------------

Name	Type	Default	Description
cUNumList array			Array containing the numbers of the individual cash units to be counted. If an invalid number is contained in this list, the command will fail with a CashUnitError error.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

## Event Messages

- [CashManagement.CashUnitInfoChangedEvent](#)
  - [CashManagement.CashUnitThresholdEvent](#)
  - [CashManagement.CashUnitErrorEvent](#)
  - [CashManagement.NoteErrorEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
- 

## CashAcceptor.DeviceLockControl

---

### Description

This command can be used to lock or unlock a CashAcceptor device, it can also be used to lock or unlock one or more cash units.

During normal device operation the device and cash units will be locked and removal will not be possible. If supported the device or cash units can be unlocked, ready for removal. In this situation the device will still remain online and cash-in or dispense operations will be possible, as long as the device or cash units are not physically removed from their normal operating position.

If the lock action is specified and the device or cash units are already locked, or if the unlock action is specified and the device or cash units are already unlocked then the action will complete successfully.

Once a cash unit has been removed and reinserted it will then have a "manualInsertion" status. This status can only be cleared by issuing a  
CashManagement.StartExchange/CashManagement.EndExchange command sequence.

The device and all cash units will also be locked implicitly as part of the execution of the  
CashManagement.EndExchange or the CashAcceptor.Reset command.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

Name	Type	Default	Description
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
deviceAction	string		<p>Specifies to lock or unlock the device in its normal operating position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"lock": Locks the device so that it cannot be removed from its normal operating position.</li> <li>"unlock": Unlocks the device so that it can be removed from its normal operating position.</li> <li>"noLockAction": No lock/unlock action will be performed on the device.</li> </ul>
cashUnitAction	integer		<p>Specifies the type of lock/unlock action on physical cash units. Following values are possible:</p> <ul style="list-style-type: none"> <li>"lockAll": Locks all physical cash units supported.</li> <li>"unlockAll": Unlocks all physical cash units supported.</li> <li>"lockIndividual": Locks/unlocks physical cash units individually as specified in the <i>unitLockControl</i> parameter.</li> <li>"noLockAction":</li> </ul>
unitLockControl	array		<p>Array of UnitLockControl structures; only valid in the case where "lockIndividual" is specified in the <i>cashUnitAction</i> field. Otherwise this field will be ignored. Each element specifies one cash unit to be locked/unlocked.</p>

Name	Type	Default	Description
unitLockControl.physicalPositionName	string		Specifies which physical cash unit is to be locked/unlocked. This name is the same as the <i>physicalPositionName</i> in the CashUnitInfo structure. Only physical cash units reported by the CashAcceptor.DeviceLockStatus command can be specified.
unitLockControl.unitAction	string		<p>Specifies whether to lock or unlock the physical cash unit indicated in the <i>physicalPositionName</i> parameter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"lock": Locks the specified cash unit so that it cannot be removed from the device.</li> <li>"unlock": Unlocks the specified cash unit so that it can be removed from the device.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "deviceAction": "lock",
    "cashUnitAction": 0,
    "unitLockControl": [
      {
        "physicalPositionName": "string",
        "unitAction": "lock"
      }
    ]
  }
}
```

#### Completion Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitErrorEvent](#)

## CashAcceptor.SetMode

### Description

This command is used to set the deposit mode for the device and is only applicable for Mixed Media processing. The deposit mode determines how the device will process non cash items that are inserted. The deposit mode applies to all subsequent transactions. The deposit mode is persistent and is unaffected by a device reset by CashDispenser.Reset or reset on another interface. The command will fail with a InvalidData error where an attempt is made to set a mode that is not supported.

### Command Message

#### Message Header

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
mixedMode	string	mixedMediaNotActive	<p>Specifies the Mixed Media mode of the device. Following values are possible:</p> <p>"mixedMediaNotActive": Mixed Media transactions are deactivated. This is the default mode.</p> <p>"mixedMedia": Mixed Media transactions are activated in combination with the <code>ItemProcessor</code> interface as defined by the capability <code>mixedMode</code>.</p>

### Example Message (generated)

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

### Completion Message

#### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

---

## CashAcceptor.PresentMedia

---

### Description

This command opens the shutter and presents items to be taken by the customer. The shutter is automatically closed after the media is taken. The command can be called after a CashAcceptor.CashIn, CashAcceptor.CashInRollback, CashAcceptor.Reset or CashAcceptor.CreateP6Signature command and can be used with explicit and implicit shutter control. The command is only valid on positions where *usage* reported by the CashAcceptor.PositionCapabilities command is "rollback" or "refuse" and where *presentControl* reported by the CashAcceptor.PositionCapabilities command is FALSE.

This command cannot be used to present items stacked through the CashDispenser interface. Where this is attempted the command fails with a SequenceError error.

### Mixed Media Mode:

If the device is operating in Mixed Media mode (Status *mixedMode* == "mixedMedia") this command will not perform any operation unless the ItemProcessor.PresentMedia command is called or has already been called on the ItemProcessor interface. Shutter control on devices that support Mixed Media processing is always implicit.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Describes the position where the media is to be presented. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": The default configuration information should be used.</li> <li>"inLeft": Present items to the left input position.</li> <li>"inRight": Present items to the right input position.</li> <li>"inCenter": Present items to of the center input position.</li> <li>"inTop": Present items to the top input position.</li> <li>"inBottom": Present items to the bottom input position.</li> <li>"inFront": Present items to the front input position.</li> <li>"inRear": Present items to the rear input position.</li> <li>"outLeft": Present items to the left output position.</li> <li>"outRight": Present items to the right output position.</li> <li>"outCenter": Present items to the center output position.</li> <li>"outTop": Present items to the top output position.</li> <li>"outBottom": Present items to the bottom output position.</li> <li>"outFront": Present items to the front output position.</li> <li>"outRear": Present items to of the rear output position.</li> </ul>

Name	Type	Default	Description
<b>Example Message (generated)</b>			

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [CashAcceptor.ItemsTakenEvent](#)
- [CashAcceptor.ItemsPresentedEvent](#)

- [CashAcceptor.ShutterStatusChangedEvent](#)

## CashAcceptor.Deplete

### Description

This command removes items from multiple cash units to a single cash unit. Applications can use this command to ensure that there is the optimum number of items in the cassettes by moving items from source cash units to a target cash unit. This is especially applicable if surplus items are removed from multiple recycle cash units to a replenishment cash unit and can help to minimize manual replenishment operations.

The CashAcceptor.DepleteSource command can be used to determine what cash units can be specified as source cash units for a given target cash unit.

The *count*, *cashInCount*, *dispensedCount* and *rejectCount* returned with the CashManagement.CashUnitInfo command will be updated as part of the execution of this command.

If the command fails after some items have been moved, the command will complete with an appropriate error code, and a CashAcceptor.IncompleteDepleteEvent event will be sent.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
depleteSources	array		Array of DepleteSource structures. There must be at least one element in this array.
depleteSources.numberSource	integer		Index number of the cash unit from which items are to be removed. This is the index number identifier defined in the <i>number</i> field of the CashUnit structure of the output data of the CashManagement.CashUnitInfo command.

Name	Type	Default	Description
depleteSources.numberOfItemsToMove	integer		The number of items to be moved from the source cash unit. This must be equal to or less than the count of items reported for the cash unit specified by <i>numberSource</i> . This field will be ignored if the <i>removeAll</i> parameter is set to TRUE.
depleteSources.removeAll	boolean		Specifies if all items are to be moved from the source cash unit. If TRUE all items in the source will be moved, regardless of the <i>numberOfItemsToMove</i> field value. If FALSE the number of items specified with <i>numberOfItemsToMove</i> will be moved.
numberTarget	integer		Index number of the cash unit to which items are to be moved. This is the index number identifier defined in the <i>number</i> field of the CashUnit structure of the output data of the CashManagement.CashUnitInfo command.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "depleteSources": [
      {
        "numberSource": 0,
        "numberOfItemsToMove": 0,
        "removeAll": true
      }
    ],
    "numberTarget": 0
  }
}
```

#### Completion Message

##### Message Header

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

Message Payload Default	Description		
Name	Type	Default	Description
numberOfItemsReceived	integer		Total number of items received in the target cash unit during execution of this command.
numberOfItemsRejected	integer		Total number of items rejected during execution of this command.
depleteSourceResults	array		<p>Array of DepleteSpourceResult structures. In the case where one item type has several releases and these are moved, or where items are moved from a multi denomination cash unit to a multi denomination cash unit, each source can move several <i>noteID</i> item types.</p> <p>For example: If one single source was specified with the <i>depleteSources</i> input structure, and this source moved two different <i>noteID</i> item types, then the <i>depleteSourceResults</i> array will have two elements. Or if two sources were specified and the first source moved two different <i>noteID</i> item types and the second source moved three different <i>noteID</i> item types, then the <i>depleteSourceResults</i> array will have five elements.</p>
depleteSourceResults.numberSource	integer		<p>Index number of the logical cash unit from which items have been removed. This is the index number identifier defined in the <i>number</i> field of the output structure of the output data of the CashManagement.CashUnitInfo command.</p>
depleteSourceResults.noteID	integer		Identification of item type. The note ID represents the item identifiers reported by the CashAcceptor.BanknoteTypes command.

Name	Type	Default	Description
depleteSourceResults.numberOfItemsRemoved	integer		Total number of items removed from this source cash unit of the <i>noteID</i> item type. A zero value will be returned if this source cash unit did not move any items of this item type, for example due to a cash unit or transport jam.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "numberOfItemsReceived": 0,
    "numberOfItemsRejected": 0,
    "depleteSourceResults": [
      {
        "numberSource": 0,
        "noteID": 0,
        "numberOfItemsRemoved": 0
      }
    ]
  }
}
```

#### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashAcceptor.IncompleteDepleteEvent](#)

## CashAcceptor.PreparePresent

#### Description

In cases where multiple bunches are to be returned under explicit shutter control, this command is used for the purpose of moving a remaining bunch to the output position explicitly before using the following commands:

OpenShutter

CashAcceptor.PresentMedia

The application can tell whether the additional items were left by using CashAcceptor.PresentStatus command. This command does not affect the status of the current cash-in transaction.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
position	string		<p>Describes the position where the items are to be moved. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": The default configuration information should be used.</li> <li>"outLeft": Move items to the left output position.</li> <li>"outRight": Move items to the right output position.</li> <li>"outCenter": Move items to the center output position.</li> <li>"outTop": Move items to the top output position.</li> <li>"outBottom": Move items to the bottom output position.</li> <li>"outFront": Move items to the front output position.</li> <li>"outRear": Move items to the rear output position.</li> </ul>

### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)

## Unsolicited Events

### CashAcceptor.ItemsTakenEvent

---

#### Description

This service specifies that items presented to the user have been taken. This event may be generated at any time.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Specifies the position where the items have been inserted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"inLeft": Items taken from the left input position.</li> <li>"inRight": Items taken from the right input position.</li> <li>"inCenter": Items taken from the center input position.</li> <li>"inTop": Items taken from the top input position.</li> <li>"inBottom": Items taken from the bottom input position.</li> <li>"inFront": Items taken from the front input position.</li> <li>"inRear": Items taken from the rear input position.</li> <li>"outLeft": Items taken from the left output position.</li> <li>"outRight": Items taken from the right output position.</li> <li>"outCenter": Items taken from the center output position.</li> <li>"outTop": Items taken from the top output position.</li> <li>"outBottom": Items taken from the bottom output position.</li> <li>"outFront": Items taken from the front output position.</li> <li>"outRear": Items taken from the rear output position.</li> </ul>

---

#### Example Message (generated)

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

## CashAcceptor.ItemsPresentedEvent

### Description

This service event specifies that items have been presented to the output position, and the shutter has been opened to allow the user to take the items.

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Specifies the position where the items have been inserted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"inLeft": Items presented at the left input position.</li> <li>"inRight": Items presented at the right input position.</li> <li>"inCenter": Items presented at the center input position.</li> <li>"inTop": Items presented at the top input position.</li> <li>"inBottom": Items presented at the bottom input position.</li> <li>"inFront": Items presented at the front input position.</li> <li>"inRear": Items presented at the rear input position.</li> <li>"outLeft": Items presented at the left output position.</li> <li>"outRight": Items presented at the right output position.</li> <li>"outCenter": Items presented at the center output position.</li> <li>"outTop": Items presented at the top output position.</li> <li>"outBottom": Items presented at the bottom output position.</li> <li>"outFront": Items presented at the front output position.</li> <li>"outRear": Items presented at the rear output position.</li> </ul>

Name	Type	Default	Description
additionalBunches	string		<p>Specifies whether or not additional bunches of items are remaining to be presented as a result of the current operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"none": No additional bunches remain.</li> <li>"oneMore": At least one additional bunch remains.</li> <li>"unknown": It is unknown whether additional bunches remain.</li> </ul>
bunchesRemaining	integer		<p>If <i>additionalBunches</i> is "oneMore", specifies the number of additional bunches of items remaining to be presented as a result of the current operation. If the number of additional bunches is at least one, but the precise number is unknown, <i>bunchesRemaining</i> will be 255 (TODO: Check if there is a better way to represent this state). For any other value of <i>additionalBunches</i>, <i>bunchesRemaining</i> will be zero.</p>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "position": "inLeft",
    "additionalBunches": "none",
    "bunchesRemaining": 0
  }
}
```

## CashAcceptor.ItemsInsertedEvent

---

### Description

This service event specifies that items have been inserted into the cash-in position by the user. This event may be generated at any time.

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Specifies the position where the items have been inserted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"inLeft": Items detected in the left input position.</li> <li>"inRight": Items detected in the right input position.</li> <li>"inCenter": Items detected in the center input position.</li> <li>"inTop": Items detected in the top input position.</li> <li>"inBottom": Items detected in the bottom input position.</li> <li>"inFront": Items detected in the front input position.</li> <li>"inRear": Items detected in the rear input position.</li> <li>"outLeft": Items detected in the left output position.</li> <li>"outRight": Items detected in the right output position.</li> <li>"outCenter": Items detected in the center output position.</li> <li>"outTop": Items detected in the top output position.</li> <li>"outBottom": Items detected in the bottom output position.</li> <li>"outFront": Items detected in the front output position.</li> <li>"outRear": Items detected in the rear output position.</li> </ul>

#### Example Message (generated)

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

## CashAcceptor.MediaDetectedEvent

### Description

This service event is generated if media is detected during a CashAcceptor.Reset command. The parameter on the event specifies the position of the media on completion of the reset. If the device has been unable to successfully move the items found then this parameter will be omitted.

### Message Header

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

### Message Payload

Name	Type	Default	Description
number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found. If items are to be moved to an output position, this value must be zero, <i>retractArea</i> must be omitted and <i>outputPosition</i> specifies where items are to be moved to. If this value is zero and items are to be moved to internal areas of the device, <i>retractArea</i> specifies where items are to be moved to or stored.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
retractArea	object		<p>This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker or the transport. The field is only relevant if <i>number</i> is zero.</p>
retractArea.outputPosition	string		<p>Specifies the output position from which to retract the bills. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": The default configuration information should be used. This value is also used to retract items from internal device locations.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>

Name	Type	Default	Description
retractArea.retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"billCassettes": Retract the items to item cassettes, i.e. cash-in and recycle cash units.</li> <li>"cashIn": Retract the items to a cash-in cash unit. The <i>itemType</i> of the cash-in cash unit defined in <i>cashInfo</i> must include ("all"   "unfit").</li> </ul> <p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. The maximum value of <i>index</i> is the sum of the <i>maximum</i> of each retract cash unit.</p>
retractArea.index	integer		<p>If <i>retractArea</i> is set to "cashIn" this field defines the physical cash unit under the "cashIn" cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) and would be incremented from the first physical cash unit to the last physical cash unit defined in <i>cashInfo</i>.</p> <p>If <i>retractArea</i> is not set to "retract" or "cashIn" then the value of this field is ignored.</p>

Name	Type	Default	Description
outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>retractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"null": Take the default configuration.</li> <li>"left": Move items to the left output position.</li> <li>"right": Move items to the right output position.</li> <li>"center": Move items to the center output position.</li> <li>"top": Move items to the top output position.</li> <li>"bottom": Move items to the bottom output position.</li> <li>"front": Move items to the front output position.</li> <li>"rear": Move items to the rear output position.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "retractArea": {
      "outputPosition": null,
      "retractArea": "retract",
      "index": 0
    },
    "outputPosition": null
  }
}
```

---

## CashAcceptor.ShutterStatusChangedEvent

---

### Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Within the limitations of the hardware sensors this service event is generated whenever the status of a shutter changes. The shutter status can change because of an explicit, implicit or manual operation depending on how the shutter is operated.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Specifies one of the input or output positions whose shutter status has changed. Following values are possible:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>

Name	Type	Default	Description
shutter	string		<p>Specifies the new state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is closed.</li> <li>"open": The shutter is opened.</li> <li>"jammed": The shutter is jammed.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> </ul>

**Example Message (generated)**

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

**CashAcceptor.CountAccuracyChangedEvent****Description**

This event is generated when information about the accuracy of *count* contained in the logical or physical cash unit is changed.

**Message Header**

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

**Message Payload**

	Name	Type	Default	Description
	deplete	array		
	deplete.number	integer		Index number of the logical cash unit.

Name	Type	Default	Description
deplete.accuracy	string		<p>Describes the accuracy of <i>count</i>. Following values are possible:</p> <p>"notSupported": The hardware is not capable to determine the accuracy of <i>count</i>.</p> <p>"accurate": The <i>count</i> is expected to be accurate. The notes were previously counted or replenished and there have since been no events that might have introduced inaccuracy. This value will be reported as a result of the following commands: Replenish and CashUnitCount.</p> <p>"accurateSet": The <i>count</i> is expected to be accurate. The notes were previously set and there have since been no events that might have introduced inaccuracy.</p> <p>"inaccurate": The <i>count</i> is likely to be inaccurate. A jam, picking fault, or some other event may have resulted in a counting inaccuracy.</p> <p>"unknown": The accuracy of <i>count</i> cannot be determined. This may be due to cash unit insertion or some other hardware event.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
deplete.physicalCashUnitStatus	array	Array of cash count status objects for physical cash units of this logical cash unit.	
deplete.physicalCashUnitStatus.physicalPositionName	string	A name identifying the physical location of the cash unit within the CashAcceptor. This field can be used to identify shared cash units/media bins.	
deplete.physicalCashUnitStatus.accuracy	string	<p>Describes the accuracy of <i>count</i> of a physical cash unit. Following values are possible:</p> <p>"notSupported": The hardware is not capable to determine the accuracy of <i>count</i>.</p> <p>"accurate": The <i>count</i> is expected to be accurate. The notes were previously counted or replenished and there have since been no events that might have introduced inaccuracy. This value will be reported as a result of the following commands: Replenish and CashUnitCount.</p> <p>"accurateSet": The <i>count</i> is expected to be accurate. The notes were previously set and there have since been no events that might have introduced</p>	

Name	Type	Default	Description
		"inaccurate"	"inaccurate": The <i>count</i> is likely to be inaccurate. A jam, picking fault, or some other event may have resulted in a counting inaccuracy.
		"unknown"	"unknown": The accuracy of <i>count</i> cannot be determined. This may be due to cash unit insertion or some other hardware event.
deplete.physicalCashUnitStatus.extra	array	TODO	
deplete.extra	array	TODO	

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "deplete": [
      {
        "number": 0,
        "accuracy": "notSupported",
        "physicalCashUnitStatus": [
          {
            "physicalPositionName": "string",
            "accuracy": "notSupported",
            "extra": [
              "string"
            ]
          }
        ],
        "extra": [
          "string"
        ]
      }
    ]
  }
}
```

## Events

### CashManagement.CashUnitErrorEvent

---

#### Description

This event is generated if there is a problem with a cash unit during the execution of a command.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
failure	string		<p>Specifies the kind of failure that occurred in the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": Specified cash unit is empty.</li> <li>"error": Specified cash unit has malfunctioned.</li> <li>"full": Specified cash unit is full.</li> <li>"locked": Specified cash unit is locked.</li> <li>"invalid": Specified cash unit is invalid.</li> <li>"config": An attempt has been made to change the settings of a self-configuring cash unit.</li> <li>"notConfigured": Specified cash unit is not configured.</li> </ul>
cashUnit.	object		Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.
cashUnit.number	integer		

Name	Type	Default	Description
cashUnit.type	string		<p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> <li>"retractCassette": Retract cash unit.</li> <li>"coupon": Cash unit containing coupons or advertising material.</li> <li>"document": Cash unit containing documents.</li> <li>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</li> <li>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</li> <li>"cashIn": Cash-in cash unit.</li> </ul>
cashUnit.unitID	string		The Cash Unit Identifier.
cashUnit.currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
cashUnit.values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
cashUnit.count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
cashUnit.maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>ulCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent (high)</i> will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
cashUnit.status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"maneuellnsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
cashUnit.appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
cashUnit.physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnit.cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
cashUnit.initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
cashUnit.dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
cashUnit.rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
cashUnit.minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
cashUnit.	object		
cashUnit.itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
cashUnit.itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
cashUnit.itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
cashUnit.itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
cashUnit.itemType.level1	boolean		Level 1 note types are stored in this cash unit.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
cashUnit.itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
cashUnit.itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
cashUnit.itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
cashUnit.itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashUnit.cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
cashUnit.noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
cashUnit.noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
cashUnit.noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
cashUnit.noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
cashUnit.noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
cashUnit.extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "failure": "empty",
    "cashUnit": {
      "number": 0,
      "type": "notApplicable",
      "unitID": "string",
      "currencyID": "string",
      "values": 0,
      "count": 0,
      "maximum": 0,
      "status": "ok",
      "appLock": true,
      "physical": [
        {
          "physicalPositionName": "string",
          "unitID": "string",
          "noteID": 0
        }
      ]
    }
  }
}
```

```
        "count": 0,
        "maximum": 0,
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    },
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
}
```

---

## CashAcceptor.InputRefuseEvent

---

### Description

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

This event specifies that the device has refused either a portion or the entire amount of the cash-in order.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

Name	Type	Default	Description
reason	string		<p>Reason for refusing a part of the amount. Following values are possible:</p> <ul style="list-style-type: none"> <li>"cashInUnitFull": Cash unit is full.</li> <li>"invalidBill": Recognition of the items took place, but one or more of the items are invalid.</li> <li>"noBillsToDeposit": There are no items in the input area.</li> <li>"depositFailure": A deposit has failed for a reason not covered by the other reasons and the failure is not a fatal hardware problem, for example failing to pick an item from the input area.</li> <li>"commonInputComponentFailure": Failure of a common input component which is shared by all cash units.</li> <li>"stackerFull": The intermediate stacker is full.</li> <li>"foreignItemsDetected": Foreign items have been detected in the input position.</li> <li>"invalidBunch": Recognition of the items did not take place. The bunch of notes inserted is invalid, e.g. it is too large or was inserted incorrectly.</li> <li>"counterfeit": One or more counterfeit items have been detected and refused. This is only applicable where notes are not classified as level 2 and the device is capable of differentiating between invalid and counterfeit items.</li> <li>"limitOverTotalItems": Number of items count exceeded the limitation set with the CashAcceptor.SetCashInLimit command.</li> <li>"limitOverAmount": Amount exceeded the limitation set with the CashAcceptor.SetCashInLimit command.</li> </ul>

---

**Example Message (generated)**

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

## CashManagement.NoteErrorEvent

### Description

This event specifies the reason for a note detection error during the execution of a command.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
reason	string		<p>The reason for the notes detection error.</p> <p>Following values are possible:</p> <ul style="list-style-type: none"> <li>"doubleNote": Double notes have been detected.</li> <li>"longNote": A long note has been detected.</li> <li>"skewedNote": A skewed note has been detected.</li> <li>"incorrectCount": An item counting error has occurred.</li> <li>"notesTooClose": Notes have been detected as being too close.</li> <li>"otherNoteError": An item error not covered by the other values has been detected.</li> <li>"shortNote": Short notes have been detected.</li> </ul>

#### Example Message (generated)

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

## CashAcceptor.SubCashInEvent

### Description

This event is generated when one of the sub cash-in operations into which the cash-in operation was divided has finished successfully.

### Message Header

Name	Type	Default	Description
------	------	---------	-------------

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.

### Example Message (generated)

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

## CashManagement.InfoAvailableEvent

### Description

This execute event is generated when information is available for items detected during the cash processing operation.

### Message Header

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
itemInfoSummary	array		Array of itemInfoSummary objects, one object for every level.
itemInfoSummary.level	string		Defines the note level. Following values are possible: "level1": Information for level 1 notes. "level2": Information for level 2 notes. "level3": Information for level 3 notes. "level4": Information for level 4 notes.
itemInfoSummary.numOfItems	integer		Number of items classified as <i>level</i> which have information available.

#### Example Message (generated)

```
{  
  "headers": {  
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
    "type": "command",  
    "name": "string"  
  },  
  "payload": {  
    "itemInfoSummary": [  
      {  
        "level": "level1",  
        "numOfItems": 0  
      }  
    ]  
  }  
}
```

## CashAcceptor.InsertItemsEvent

### Description

This event notifies the application when the device is ready for the user to insert items.

### Message Header

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

### Example Message (generated)

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

## CashManagement.CashUnitThresholdEvent

### Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

This user event is generated when a threshold condition has occurred in one of the logical cash units. This event can be triggered either by hardware sensors in the device or by the logical *count* reaching the *minimum* or *maximum* value as specified in the CashUnitInfo structure. The application can check if the device has hardware sensors by querying the *hardwareSensor* field of the physical cash unit structure. If any of the physical cash units associated with the logical cash unit have this capability then threshold events based on hardware sensors will be triggered if the *maximum* or *minimum* values are not used and are set to zero. In the situation where the cash unit is associated with multiple physical cash units the CashManagement.CashUnitInfoChangedEvent will be generated when any of the physical cash units reaches the threshold. When the final physical cash unit reaches the threshold, the CashManagement.CashUnitThresholdEvent as well as the CashManagement.CashUnitInfoChangedEvent event will be generated.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
number	integer	object	Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

Name	Type	Default	Description
type	string		<p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> <li>"retractCassette": Retract cash unit.</li> <li>"coupon": Cash unit containing coupons or advertising material.</li> <li>"document": Cash unit containing documents.</li> <li>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</li> <li>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</li> <li>"cashIn": Cash-in cash unit.</li> </ul>
unitID	string		The Cash Unit Identifier.
currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>uiCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> (<i>high</i>) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuelInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
itemType.level1	boolean		Level 1 note types are stored in this cash unit.

Name	Type	Default	Description
itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0
      }
    ]
  }
}
```

```
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    },
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
```

---

## CashManagement.CashUnitInfoChangedEvent

---

### Description

This service event is generated under the following circumstances:

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

- It is generated whenever *status* and/or *pStatus* changes. For instance, a physical cash unit has been removed or inserted, or a physical/logical cash unit has become empty or full.
- This event will also be generated for every cash unit changed in any way (including changes to counts, e.g. *count*, *rejectCount*, *initialCount*, *dispensedCount* and *presentedCount*) as a result of the following commands:

CashManagement.SetCashUnitInfo > CashManagement.EndExchange

- This event will also be fired when any change is made to a cash unit by the following commands, except for changes to counts (e.g. *count*, *rejectCount*, *initialCount*, *dispensedCount* and *presentedCount*):

Dispenser.CalibrateCashUnit > Dispenser.TestCashUnit

- In addition this event will be generated when a cash unit has been counted during the CashAcceptor.CashUnitCount command execution. When a physical cash unit is removed, the status of the physical cash unit becomes "missing". If there are no physical cash units of the same logical type remaining the status of the logical type becomes "missing". When a physical cash unit is inserted and this physical cash unit is of an existing logical type, both the logical and the physical cash unit structures will be updated. If a physical cash unit of a new logical type is inserted the cash unit structure reported by the last CashManagement.CashUnitInfo command is no longer valid. In that case an application should issue a CashManagement.CashUnitInfo command after receiving this event to obtain updated cash unit information.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
number	integer		Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

Name	Type	Default	Description
			<p>Type of cash unit. Following values are possible:</p> <p>"notApplicable": Not applicable. Typically means cash unit is missing.</p> <p>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</p> <p>"billCassette": Cash unit containing bills.</p> <p>"coinCylinder": Coin cylinder.</p> <p>"coinDispenser": Coin dispenser as a whole unit.</p>
type	string		<p>"retractCassette": Retract cash unit.</p> <p>"coupon": Cash unit containing coupons or advertising material.</p> <p>"document": Cash unit containing documents.</p> <p>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</p> <p>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</p> <p>"cashIn": Cash-in cash unit.</p>
unitID	string		The Cash Unit Identifier.
currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>uiCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> (<i>high</i>) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuelInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
itemType.level1	boolean		Level 1 note types are stored in this cash unit.

Name	Type	Default	Description
itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0,
        "physicalType": "string"
      }
    ]
  }
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    ],
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
```

---

## CashManagement.CountsChangedEvent

---

### Description

Deprecated

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
cUNumList	array		Array of the number values of the cash units whose counts have changed.

### Example Message (generated)

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

## CashAcCashManagementceptor.CashUnitErrorEvent

### Description

This event is generated if there is a problem with a cash unit during the execution of a command.

### Message Header

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

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
**Message Payload**

Name	Type	Default	Description
failure	string		<p>Specifies the kind of failure that occurred in the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": Specified cash unit is empty.</li> <li>"error": Specified cash unit has malfunctioned.</li> <li>"full": Specified cash unit is full.</li> <li>"locked": Specified cash unit is locked.</li> <li>"invalid": Specified cash unit is invalid.</li> <li>"config": An attempt has been made to change the settings of a self-configuring cash unit.</li> <li>"notConfigured": Specified cash unit is not configured.</li> </ul>
cashUnit.	object		
cashUnit.number	integer		<p>Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.</p>

Name	Type	Default	Description
cashUnit.type	string		<p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> <li>"retractCassette": Retract cash unit.</li> <li>"coupon": Cash unit containing coupons or advertising material.</li> <li>"document": Cash unit containing documents.</li> <li>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</li> <li>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</li> <li>"cashIn": Cash-in cash unit.</li> </ul>
cashUnit.unitID	string		The Cash Unit Identifier.
cashUnit.currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
cashUnit.values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
cashUnit.count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
cashUnit.maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>ulCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent (high)</i> will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
cashUnit.status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuellInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
cashUnit.appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
cashUnit.physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnit.cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
cashUnit.initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
cashUnit.dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
cashUnit.rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
cashUnit.minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event CashManagement.CashUnitThresholdEvent ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
cashUnit.	object		
cashUnit.itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
cashUnit.itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
cashUnit.itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
cashUnit.itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
cashUnit.itemType.level1	boolean		Level 1 note types are stored in this cash unit.

Name	Type	Default	Description
cashUnit.itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
cashUnit.itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
cashUnit.itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
cashUnit.itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashUnit.cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
cashUnit.noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
cashUnit.noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
cashUnit.noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
cashUnit.noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
cashUnit.noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
cashUnit.extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "failure": "empty",
    "cashUnit": {
      "number": 0,
      "type": "notApplicable",
      "unitID": "string",
      "currencyID": "string",
      "values": 0,
      "count": 0,
      "maximum": 0,
      "status": "ok",
      "appLock": true,
      "physical": [
        {
          "physicalPositionName": "string",
          "unitID": "string"
        }
      ]
    }
  }
}
```

```
"unitID": "string",
    "count": 0,
    "maximum": 0,
    "pStatus": "ok",
    "hardwareSensor": true,
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "cashInCount": 0,
    "extra": [
        "string"
    ]
},
],
"cashUnitName": "string",
"initialCount": 0,
"dispensedCount": 0,
"presentedCount": 0,
"retractedCount": 0,
"rejectCount": 0,
"minimum": 0,
"itemType": {
    "all": true,
    "unfit": true,
    "individual": true,
    "level1": true,
    "level2": true,
    "level3": true,
    "itemProcessor": true,
    "unfitIndividual": true
},
"cashInCount": 0,
"noteNumberList": {
    "noteNumber": [
        {
            "noteID": 0,
            "count": 0
        }
    ]
},
"noteIDs": [
    0
],
"extra": [
    "string"
]
}
}
```

---

## CashAcceptor.IncompleteReplenishEvent

---

## Description

This event is generated when some items had been moved before the CashAcceptor.Replenish command failed with an error code (not "success"), but some items were moved then the details will be reported with this event. This event can only occur once per command.

## Message Header

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

## Message Payload

Name	Type	Default	Description
replenish	object		Note that in this case the values in this structure report the amount and number of each denomination that have actually been moved during the replenishment command.
replenish.numberItemsRemoved	integer		Total number of items removed from the source cash unit including rejected items during execution of this command.
replenish.numberItemsRejected	integer		Total number of items rejected during execution of this command.

Name	Type	Default	Description
replenish.replenishTargetResults	array		Array of replenishTargetResult structures. In the case where one note type has several releases and these are moved, or where items are moved from a multi denomination cash unit to a multi denomination cash unit, each target can receive several <i>noteID</i> note types. For example: If one single target was specified with the <i>replenishTargets</i> input structure, and this target received two different <i>noteID</i> note types, then the <i>replenishTargetResults</i> array will have two elements. Or if two targets were specified and the first target received two different <i>noteID</i> note types and the second target received three different <i>noteID</i> note types, then the <i>replenishTargetResults</i> array will have five elements.
replenish.replenishTargetResults.numberTarget	integer		Index number of the cash unit to which items have been moved. This is the index number identifier defined in the <i>number</i> field of the output data of the CashManagement.CashUnitInfo command.
replenish.replenishTargetResults.noteID	integer		Identification of note type. The note ID represents the note identifiers reported by the CashAcceptor.BanknoteTypes command.
replenish.replenishTargetResults.numberOfItemsReceived	integer		Total number of items received in this target cash unit of the <i>noteID</i> note type. A zero value will be returned if this target cash unit did not receive any items of this note type, for example due to a cash unit or transport jam.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "replenish": {
      "numberOfItemsRemoved": 0,
      "numberOfItemsRejected": 0,
      "replenishTargetResults": [
        {
          "numberTarget": 0,
          "noteID": 0,
          "numberOfItemsReceived": 0
        }
      ]
    }
  }
}
```

## CashAcceptor.IncompleteDeleteEvent

### Description

This execute event is generated when some items had been moved before the CashAcceptor.Deplete command failed with an error code (not "success"), but some items were moved. In this case the details will be reported with this event. This event can only occur once per command.

### Message Header

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

### Message Payload

Name	Type	Default	Description
deplete	object		Note that in this case the values in this structure report the amount and number of each denomination that have actually been moved during the depletion command.

Name	Type	Default	Description
deplete.numberItemsReceived	integer		Total number of items received in the target cash unit during execution of this command.
deplete.numberItemsRejected	integer		Total number of items rejected during execution of this command.
deplete.depleteSourceResults	array		<p>Array of DepleteSpourceResult structures. In the case where one item type has several releases and these are moved, or where items are moved from a multi denomination cash unit to a multi denomination cash unit, each source can move several <i>noteID</i> item types.</p> <p>For example: If one single source was specified with the <i>depleteSources</i> input structure, and this source moved two different <i>noteID</i> item types, then the <i>depleteSourceResults</i> array will have two elements. Or if two sources were specified and the first source moved two different <i>noteID</i> item types and the second source moved three different <i>noteID</i> item types, then the <i>depleteSourceResults</i> array will have five elements.</p>
deplete.depleteSourceResults.numberSource	integer		<p>Index number of the logical cash unit from which items have been removed. This is the index number identifier defined in the <i>number</i> field of the output structure of the output data of the CashManagement.CashUnitInfo command.</p>
deplete.depleteSourceResults.noteID	integer		Identification of item type. The note ID represents the item identifiers reported by the CashAcceptor.BanknoteTypes command.

Name	Type	Default	Description
deplete.depleteSourceResults.numberOfItemsRemoved	integer		Total number of items removed from this source cash unit of the <i>noteID</i> item type. A zero value will be returned if this source cash unit did not move any items of this item type, for example due to a cash unit or transport jam.

**Example Message (generated)**

XFS4  
risk  
All rig

```
{  
  "headers": {  
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
    "type": "command",  
    "name": "string"  
  },  
  "payload": {  
    "deplete": {  
      "numberOfItemsReceived": 0,  
      "numberOfItemsRejected": 0,  
      "depleteSourceResults": [  
        {  
          "numberSource": 0,  
          "noteID": 0,  
          "numberOfItemsRemoved": 0  
        }  
      ]  
    }  
  }  
}
```

# XFS4IoT Preview Messaging for Dispenser Draft

## 0.0.3

This specification describes the functionality of an XFS4IoT compliant Cash Dispenser interface. It defines the service-specific commands that can be issued to the service using the WebSocket endpoint.

Persistent values are maintained through power failures, open sessions, close session and system resets.

This specification covers the dispensing of items. An "item" is defined as any media that can be dispensed and includes coupons, documents, bills and coins.

## Commands

### Dispenser.MixTypes

---

#### Description

This command is used to obtain a list of supported mix algorithms and available house mix tables.

#### Command Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
mixTypes	array		Array of mix type objects.
mixTypes.mixNumber	integer		Number identifying the mix algorithm or the house mix table. This number can be passed to the Dispenser.MixTable, Dispenser.Dispense and Dispenser.Denominate commands.
mixTypes.mixType	string		Specifies whether the mix type is an algorithm or a house mix table. Possible values are "mixAlgorithm" and "mixTable".
mixTypes.subType	integer		Contains a vendor-defined number that identifies the type of algorithm. Individual vendor-defined mix algorithms are defined above hexadecimal 7FFF. Mix algorithms which are provided by the Service are in the range hexadecimal 8000 - 8FFF. Application defined mix algorithms start at hexadecimal 9000. All numbers below 8000 hexadecimal are reserved. If <i>mixType</i> is "mixTable", this value will be zero.
mixTypes.name	string		Name of the table/algorith used.

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "mixTypes": [
      {
        "mixNumber": 0,
        "mixType": "mixAlgorithm",
        "subType": 0,
        "name": "string"
      }
    ]
  }
}
```

## Event Messages

### Dispenser.MixTable

#### Description

This command is used to obtain the house mix table specified by the supplied mix number.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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.
mixNumber	integer		Number of the requested house mix table.

Name	Type	Default	Description
------	------	---------	-------------

**Example Message (generated)**

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

**Completion Message****Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
	object		
mixNumber	integer		Number identifying the house mix table.
name	string		Name of the house mix table.
mixHeader	array		Array of floating point numbers; each element defines the value of the item corresponding to its respective column.
mixRows	array		Array of rows of the mix table.
mixRows.amount	number		Amount denominated by this mix row.

Name	Type	Default	Description
mixRows.mixture array			A mix row, an array of integers; each element defines the quantity of each item denomination in the mix used in the denomination of <i>amount</i> . The value of each array element is defined by the <i>mixHeader</i> .

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "mixNumber": 0,
    "name": "string",
    "mixHeader": [
      0
    ],
    "mixRows": [
      {
        "amount": 0,
        "mixture": [
          0
        ]
      }
    ]
  }
}
```

## Event Messages

---

### Dispenser.PresentStatus

---

#### Description

This command is used to obtain the status of the most recent attempt to dispense and/or present items to the customer from a specified output position. The items may have been dispensed and/or presented as a result of the Dispenser.Present or Dispenser.Dispense command. This status is not updated as a result of any other command that can dispense/present items.

This value is persistent and is valid until the next time an attempt is made to present or dispense items to the customer.

The denominations reported by this command may not accurately reflect the operation if the cash units have been re-configured (e.g. if the values associated with a cash unit are changed, or new cash units are configured).

#### Command Message

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
position	string		<p>Required output position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
denomination	object		Denomination structure which contains the amount dispensed from the specified output position and the number of items dispensed from each cash unit. Where the capability <i>moveItems</i> reports <i>toStacker</i> this value is cumulative across a series of Dispenser.Dispense calls that add additional items to the stacker. Where mixed currencies were dispensed the <i>amount</i> field in the returned denomination structure will be zero and the <i>currencyID</i> field will be omitted.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
denomination.currencies	array		Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.
denomination.currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
denomination.currencies.amount	number		The amount to be denominated or dispensed.
denomination.values	array		This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands. The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.
denomination.cashBox	integer		Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.
presentState	string		<p>Supplies the status of the last dispense or present operation. Following values are possible:</p> <p>"presented": The items were presented. This status is set as soon as the customer has access to the items.</p> <p>"notPresented": The customer has not had access to the items.</p> <p>"unknown": It is not known if the customer had access to the items.</p>

Name	Type	Default	Description
extra	array		Pointer to 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 extensible by Service Providers.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "denomination": {
      "currencies": [
        {
          "currencyID": "string",
          "amount": 0
        }
      ],
      "values": [
        0
      ],
      "cashBox": 0
    },
    "presentState": "presented",
    "extra": [
      "string"
    ]
  }
}
```

#### Event Messages

---

## Dispenser.Denominate

---

#### Description

This command provides a denomination. A denomination specifies the number of items which are required from each cash unit in order to satisfy a given amount. The denomination depends upon the currencies, the mix algorithm and any partial denomination supplied by the application.

This command can also be used to validate that any denomination supplied by the application can be dispensed.

If items of differing currencies are to be included in the same denomination then the currencies array

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
 has one entry per currency. Alternatively the currency information can be omitted and the mix number must be 0 ("individual").

If the *cashBox* field returned by the Dispenser.Capabilities command is TRUE then, if the entire denomination cannot be satisfied, a partial denomination will be returned with the remaining amount to be supplied from the teller's cash box.

This command can be used in four different ways:

1. In order to check that it is possible to dispense a given denomination. The input parameters to the command are currency and denomination, with a mix number of 0 ("individual") and an amount of zero. If items of differing currencies are to be dispensed then the currencies array needs one item per currency.
2. In order to validate that a given amount matches a given denomination and that it is possible to dispense the denomination. The input parameters to the command should be amount, currency and denomination, with a mix number of 0 ("individual").
3. In order to obtain a denomination of a given amount. The input parameters supplied should be amount, currency and mix number.
4. In order to complete a partial denomination of a given amount. In this case the input parameters to the command should be currency, amount, mix number and either a partially specified denomination or a minimum amount from the cash box. A completed denomination is returned. *cashBox* of the denomination structure may be updated as a result of this command.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
tellerID	integer		Identification of teller. This field is ignored if the device is a Self-Service Dispenser.
mixNumber	integer		Mix algorithm or house mix table to be used.
denomination	object		Denomination object describing the contents of the denomination operation.

Name	Type	Default	Description
denomination.currencies	array		Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.
denomination.currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
denomination.currencies.amount	number		The amount to be denominated or dispensed.
denomination.values	array		This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands. The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.
denomination.cashBox	integer		Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "tellerID": 0,
    "mixNumber": 0,
    "denomination": {
      "currencies": [
        {
          "currencyID": "string",
          "amount": 0
        }
      ],
      "values": [
        0
      ],
      "cashBox": 0
    }
  }
}
```

#### Completion Message

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
currencies	array	object	Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.
currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
currencies.amount	number		The amount to be denominated or dispensed.
values	array		This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands. The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.
cashBox	integer		Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.

### Example Message (generated)

```
{  
    "headers": {  
        "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
        "type": "command",  
        "name": "string"  
    },  
    "payload": {  
        "completionCode": "success",  
        "errorDescription": "string",  
        "currencies": [  
            {  
                "currencyID": "string",  
                "amount": 0  
            }  
        ],  
        "values": [  
            0  
        ],  
        "cashBox": 0  
    }  
}
```

## Event Messages

- [CashManagement.CashUnitErrorEvent](#)
- 

## Dispenser.Dispense

---

### Description

This command performs the dispensing of items to the customer. The command provides the same functionality as the Dispenser.Denominate command plus the additional functionality of dispensing the items. If items of differing currencies are to be dispensed then the currencies array has one entry per currency. Alternatively the currency information can be omitted and the mix number must be 0 ("individual"). However, these restrictions do not apply if a single currency is dispensed with non-currency items, such as coupons.

The Dispenser.Dispense command can be used in the following ways:

1. The input parameters to the command are amounts, currencies and denomination. The mix number is 0 ("individual"). In this case, the denomination is checked for validity and, if valid, is dispensed.
  2. The input parameters are amounts, currencies and mix number. In this case the amount is denominaded and, if this succeeds, the items are dispensed.
  3. If the amounts are zero, but the currencies and the denomination are supplied with a mix number of 0 ("individual") the denomination is checked for validity and, if valid, is dispensed.
  4. The command will calculate a partial denomination of a given amount and dispense the complete denomination. In this case the input parameters to the command should be currencies, amounts, mix number and either a partially specified denomination or a minimum amount from the cash box. The cash box amount may be updated as a result of this command.
-

#### XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

When more than one physical cash unit exists for a logical cash unit number, the device selects the actual physical cash unit to use in the dispense operation.

If the *cashBox* field returned by the Dispenser.Capabilities command is TRUE then, if the entire denomination cannot be satisfied, a partial denomination will be returned with the remaining amount to be supplied from the teller's cash box.

If the device is a Teller Dispenser, the input field *position* can be set to "default". If this is the case the *tellerID* is used to perform the dispense operation to the assigned teller position.

It will be necessary to use the Dispenser.Present command to present the items to the user. If the Dispenser does not have an intermediate stacker the Dispenser.Present command does not need to be called and does not serve any purpose.

Note that a level 4 note can be dispensed, but is not necessarily presented to the customer. e.g. a note can be skewed, or can be unfit for dispensing.

The values in the completion message report the amount dispensed and the number of items dispensed from each cash unit.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		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.
tellerID	object		Identifies the teller. This field is ignored if the device is a Self-Service Dispenser.
mixNumber	integer		Mix algorithm or house mix table to be used to create a denomination of the supplied amount. If the value is 0 ("individual"), the denomination supplied in the <i>denomination</i> field is validated prior to the dispense operation. If it is found to be invalid no alternative denomination will be calculated.

Name	Type	Default	Description
position	string		<p>Required output position. Following values are possible:</p> <p>"default": The default configuration information is used. This can be either position dependent or teller dependent.</p> <p>"left": Present items to left side of device.</p> <p>"right": Present items to right side of device.</p> <p>"center": Present items to center output position.</p> <p>"top": Present items to the top output position.</p> <p>"bottom": Present items to the bottom output position.</p> <p>"front": Present items to the front output position.</p> <p>"rear": Present items to the rear output position.</p>
denomination	object		Denomination object describing the denominations used for the dispense operation.
denomination.currencies	array		Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.
denomination.currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
denomination.currencies.amount	number		The amount to be denominated or dispensed.
denomination.values	array		<p>This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands.</p> <p>The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.</p>
denomination.cashBox	integer		Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.

Name	Type	Default	Description
			<p>The dispense token that authorises the dispense operation, as created by the authorising host. See the section on end to end security for more information.</p>
			<p>The dispense token will follow the standard token format, and will contain the following key:</p>
			<p>"DISPENSE1": The maximum value to be dispensed. This will be a number string that may contain a fractional part. The decimal character will be ". ". The value, including the fractional part, will be defined by the ISO currency. The number will be followed by the ISO currency code. The currency code will be upper case.</p>
token	string		<p>For example, "123.45EUR" will be €123 and 45 cents.</p>
			<p>The "DISPENSE" key may appear multiple times with a number suffix. For example, DISPENSE1, DISPENSE2, DISPENSE3. The number will start at 1 and increment. Each key can only be given once. Each key must have a value in a different currency. For example, DISPENSE1=100.00EUR,DISPENSE2=200.00USD</p>
			<p>The actual amount dispensed will be given by the denomination. The value in the token MUST be greater or equal to the amount in the denomination parameter. If the Token has a lower value, or the Token is invalid for any reason, then the command will fail with an invalid data error code.</p>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "tellerID": 0,
    "mixNumber": 0,
    "position": "default",
    "denomination": {
      "currencies": [
        {
          "currencyID": "string",
          "amount": 0
        }
      ],
      "values": [
        0
      ],
      "cashBox": 0
    },
    "token": "string"
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
currencies	array		Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.

Name	Type	Default	Description
currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
currencies.amount	number		The amount to be denominated or dispensed.
values	array		This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands. The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.
cashBox	integer	0	Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "currencies": [
      {
        "currencyID": "string",
        "amount": 0
      }
    ],
    "values": [
      0
    ],
    "cashBox": 0
  }
}
```

#### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [Dispenser.DelayedDispenseEvent](#)
- [Dispenser.StartDispenseEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [Dispenser.ItemsTakenEvent](#)
- [Dispenser.PartialDispenseEvent](#)
- [Dispenser.SubDispenseOkEvent](#)
- [Dispenser.IncompleteDispenseEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)

- [Dispenser.ShutterStatusChangedEvent](#)
- 

## Dispenser.Present

---

### Description

This command will move items to the exit position for removal by the user. If a shutter exists, then it will be implicitly controlled during the present operation, even if the *shutterControl* capability is set to FALSE. The shutter will be closed when the user removes the items or the items are retracted. If *position* is "default" the position set in the Dispenser.Dispense command which caused these items to be dispensed will be used.

When this command successfully completes the items are in customer access.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <small>(Required)</small>	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 <small>(Required)</small>	string		The message type, either command, response, event or completion.
name <small>(Required)</small>	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
position	string		<p>Output position where the amount is to be presented. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

**Example Message (generated)**

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

**Completion Message****Message Header**

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
  - [Dispenser.ItemsTakenEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
  - [Dispenser.ShutterStatusChangedEvent](#)
- 

## Dispenser.Reject

---

### Description

This command will move items from the intermediate stacker and transport them to a reject cash unit (i.e. a cash unit with type "rejectCassette"). The *count* field of the reject cash unit is incremented by the number of items that were thought to be present at the time of the reject or the number counted by the device during the reject. Note that the reject bin count is unreliable.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
  - [CashManagement.CashUnitErrorEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
- 

## Dispenser.Retract

---

### Description

This command will retract items which may have been in customer access from an output position or from internal areas within the Dispenser. Retracted items will be moved to either a retract cash unit, a reject cash unit, item cash units, the transport or the intermediate stacker. After the items are retracted the shutter is closed automatically, even if the *shutterControl* capability is set to FALSE.

If items are moved to a retract cash unit (i.e. a cash unit with *type* "retractCassette"), then the *count* field of the retract cash unit must be incremented by 1 to specify the number of retracts. If items are moved to any other cash unit (e.g. a cash unit with *type* "rejectCassette") then the *count* field of the cash unit must be incremented by the number of items that were thought to be present at the time the Dispenser.Retract command was issued or the number counted by the device during the retract. Note that reject bin counts are unreliable.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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.
object			<p>Output position from which to retract the items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>
outputPosition	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"itemCassette": Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</li> </ul>
retractArea	string		

Name	Type	Default	Description
index	integer		If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type "retractCassette" in command CashManagement.CashUnitInfo), <i>index</i> would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit. The maximum value of <i>index</i> is the sum of <i>maximum</i> of each retract cash unit. If <i>retractArea</i> is not set to "retract" the value of this field is ignored.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
	object		
itemNumber	array		Array of item number objects.
itemNumber.currencyID	string		A three character array storing the ISO format [Ref. 2] Currency ID; if the currency of the item is not known this is omitted.
itemNumber.values	number		The value of a single item expressed as floating point value; or a zero value if the value of the item is not known.
itemNumber.release	integer		The release of the item. The higher this number is, the newer the release. Zero means that there is only one release or the release is not known. This value has not been standardized and therefore a release number of the same item will not necessarily have the same value in different systems.
itemNumber.count	integer		The count of items of the same type moved to the same destination during the execution of this command.
itemNumber.number	integer		The logical number of the cash unit which received items during the execution of this command. This value will be zero if items were moved to the <i>retractArea</i> "transport" or "stacker".

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "itemNumber": [
      {
        "currencyID": "string",
        "values": 0,
        "release": 0,
        "count": 0,
        "number": 0
      }
    ]
  }
}
```

## Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
  - [CashManagement.CashUnitErrorEvent](#)
  - [Dispenser.ItemsTakenEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
  - [Dispenser.IncompleteRetractEvent](#)
  - [Dispenser.ShutterStatusChangedEvent](#)
- 

## Dispenser.OpenShutter

---

### Description

This command opens the shutter.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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
position	string		<p>The output position where the shutter is to be opened. If the application does not need to specify a shutter, this field can be omitted or its contents set to "default". Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Open the shutter at the left output position.</li> <li>"right": Open the shutter at the right output position.</li> <li>"center": Open the shutter at the center output position.</li> <li>"top": Open the shutter at the top output position.</li> <li>"bottom": Open the shutter at the bottom output position.</li> <li>"front": Open the shutter at the front output position.</li> <li>"rear": Open the shutter at the rear output position.</li> </ul>

#### Example Message (generated)

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

#### Completion Message

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

- [Dispenser.ShutterStatusChangedEvent](#)
- 

## Dispenser.CloseShutter

---

### Description

This command closes the shutter.

### Command Message

#### Message Header

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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
position	string		<p>The output position where the shutter is to be closed. If the application does not need to specify a shutter, this field can be omitted or its contents set to "default". Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Close the shutter at the left output position.</li> <li>"right": Close the shutter at the right output position.</li> <li>"center": Close the shutter at the center output position.</li> <li>"top": Close the shutter at the top output position.</li> <li>"bottom": Close the shutter at the bottom output position.</li> <li>"front": Close the shutter at the front output position.</li> <li>"rear": Close the shutter at the rear output position.</li> </ul>

#### Example Message (generated)

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

#### Completion Message

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

- [Dispenser.ShutterStatusChangedEvent](#)
- 

**Dispenser.SetMixTable****Description**

This command is used to set up the mix table specified by the mix number. Mix tables are persistent and are available to all applications in the system. An amount can be specified as different denominations within the mix table. If the amount is specified more than once the Service Provider will attempt to denominate or dispense the first amount in the table. If this does not succeed (e.g. because of a cash unit failure) the Service Provider will attempt to denominate or dispense the next amount in the table. The Service Provider can only dispense amounts which are explicitly mentioned in the mix table.

If a mix number passed in already exists then the information is overwritten with the new information.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
	object		
mixNumber	integer		Number identifying the house mix table.
	string		
name	string		Name of the house mix table.
	array		
mixHeader	array		Array of floating point numbers; each element defines the value of the item corresponding to its respective column.
	array		
mixRows	array		Array of rows of the mix table.
	number		
mixRows.amount	number		Amount denominated by this mix row.
	array		
mixRows.mixture	array		A mix row, an array of integers; each element defines the quantity of each item denomination in the mix used in the denomination of <i>amount</i> . The value of each array element is defined by the <i>mixHeader</i> .

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "mixNumber": 0,
    "name": "string",
    "mixHeader": [
      0
    ],
    "mixRows": [
      {
        "amount": 0,
        "mixture": [
          0
        ]
      }
    ]
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

### Dispenser.Reset

---

#### Description

This command is used by the application to perform a hardware reset which will attempt to return the Dispenser device to a known good state. This command does not over-ride a lock obtained through Common.Lock (TODO) on another application or service handle.

The device will attempt to move any items found anywhere within the device to the position specified within the command payload. This may not always be possible because of hardware problems.

If items are found inside the device the Dispenser.MediaDetectedEvent will be generated and will inform the application where the items were actually moved to.

If an exchange state is active then this command will end the exchange state (even if this command does not complete successfully).

On a recycling device this command is not accepted if a cash-in transaction is active and will return a "deviceNotReady" error.

If items are moved to a retract cash unit (i.e. a cash unit with type "retractCassette"), then the count field of the retract cash unit must be incremented by 1 to specify the number of operations that changed the count. If items are moved to any other cash unit (e.g. a cash unit with type "rejectCassette"), then the count field of the cash unit must be incremented either by the number of items that were present at the time the Dispenser.Reset command was issued or the number counted by the device during the Dispenser.Reset command. Note that reject bin counts are unreliable.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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.
number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found.
retractArea	object		This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker, or the transport.
retractArea.outputPosition	string		<p>Output position from which to retract the items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>

Name	Type	Default	Description
retractArea.retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"itemCassette": Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</li> </ul>
retractArea.index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type "retractCassette" in command CashManagement.CashUnitInfo), <i>index</i> would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit. The maximum value of <i>index</i> is the sum of <i>maximum</i> of each retract cash unit. If <i>retractArea</i> is not set to "retract" the value of this field is ignored.</p>

Name	Type	Default	Description
outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>retractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "number": 0,
    "retractArea": {
      "outputPosition": "default",
      "retractArea": "retract",
      "index": 0
    },
    "outputPosition": "default"
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	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
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [Dispenser.MediaDetectedEvent](#)
- [Dispenser.ItemsTakenEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [Dispenser.IncompleteRetractEvent](#)
- [Dispenser.ShutterStatusChangedEvent](#)

## Dispenser.TestCashUnits

### Description

This command is used to test cash units following replenishment. The command payload specifies where items dispensed as a result of this command should be moved to. All physical cash units which are testable (i.e. that have a *status* of "ok" or "low" and no application lock in the logical cash unit associated with the physical cash unit) are tested. If the hardware is able to do so tests are continued even if an error occurs while testing one of the cash units. The command completes with success completion message if the Service successfully manages to test all of the testable cash units regardless of the outcome of the test. This is the case if all testable cash units could be tested and a dispense was possible from at least one of the cash units.

A CashManagement.CashUnitErrorEvent will be sent for any logical cash unit which has one or more physical cash units which cannot be tested or which fail the test, even if the logical cash unit has other physical cash units which are successfully tested. **If all the cash units could not be tested or no**

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

**cash units are testable then** a "cashUnitError" code will be returned and

CashManagement.CashUnitErrorEvents generated for every logical cash unit that encountered a problem. The operation performed to test the cash units is vendor dependent. Items may be dispensed or transported into a reject bin as a result of this command.

If no cash units are testable then a "cashUnitError" code will be returned and CashManagement.CashUnitErrorEvents will be generated for every cash unit.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found.
retractArea	object		This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker, or the transport.

Name	Type	Default	Description
retractArea.outputPosition	string		<p>Output position from which to retract the items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>
retractArea.retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"itemCassette": Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</li> </ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
retractArea.index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type "retractCassette" in command CashManagement.CashUnitInfo), <i>index</i> would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit. The maximum value of <i>index</i> is the sum of <i>maximum</i> of each retract cash unit. If <i>retractArea</i> is not set to "retract" the value of this field is ignored.</p>
outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>retractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "number": 0,
    "retractArea": {
      "outputPosition": "default",
      "retractArea": "retract",
      "index": 0
    },
    "outputPosition": "default"
  }
}
```

## Completion Message

### Message Header

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

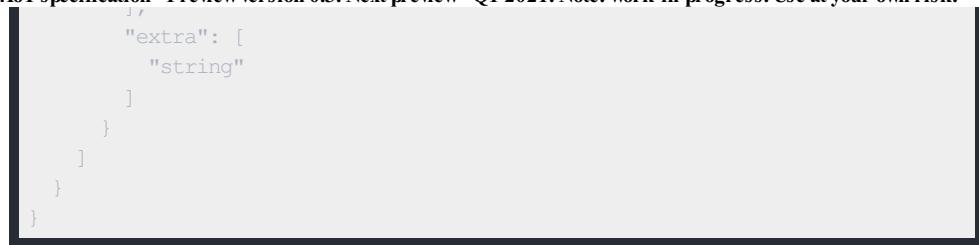
### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
list	array		Array of cash unit objects.

### Example Message (generated)

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

```
"list": [
  {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0,
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
          "string"
        ]
      }
    ],
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
      "all": true,
      "unfit": true,
      "individual": true,
      "level1": true,
      "level2": true,
      "level3": true,
      "itemProcessor": true,
      "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
      "noteNumber": [
        {
          "noteID": 0,
          "count": 0
        }
      ]
    },
    "noteIDs": [
      0
    ]
  }
]
```



## Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
  - [CashManagement.CashUnitErrorEvent](#)
  - [Dispenser.ItemsTakenEvent](#)
  - [CashManagement.CashUnitInfoChangedEvent](#)
  - [CashManagement.NoteErrorEvent](#)
  - [Dispenser.ShutterStatusChangedEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
- 

## Dispenser.Count

### Description

This command empties the specified physical cash unit(s). All items dispensed from the cash unit are counted and moved to the specified output location.

The number of items counted can be different from the number of items dispensed in cases where the Dispenser has the ability to detect this information. If the Dispenser cannot differentiate between what is dispensed and what is counted then *dispensed* will be the same as *counted*.

Upon successful Dispenser.Count command execution the physical cash unit(s) *count* field is reset.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
emptyAll	boolean object		<p>Specifies whether all physical cash units are to be emptied. If this value is TRUE then <i>physicalPositionName</i> is ignored.</p>
position	string		<p>Specifies the location to which items should be moved. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": Output location is determined by Service.</li> <li>"left": Present items to left side of device.</li> <li>"right": Present items to right side of device.</li> <li>"center": Present items to center output position.</li> <li>"top": Present items to the top output position.</li> <li>"bottom": Present items to the bottom output position.</li> <li>"front": Present items to the front output position.</li> <li>"rear": Present items to the rear output position.</li> <li>"reject": Reject bin is used as output location.</li> </ul>
physicalPositionName	string		Specifies which physical cash unit to empty and count. This name is the same as the <i>physicalPositionName</i> in the CashManagement.CashUnitInfo completion message.

**Example Message (generated)**

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
countedPhysCUs	object		
countedPhysCUs.array	array		Array of counted physical cash unit objects.
countedPhysCUs.string	string		Specifies which physical cash unit was emptied and counted. This name is the same as the <i>physicalPositionName</i> in the CashManagement.CashUnitInfo completion message.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
countedPhysCUs.unitId	string		Cash unit ID. This is the identifier defined in the <i>unitID</i> field in the CashManagement.CashUnitInfo completion message.
countedPhysCUs.dispensed	integer		The number of items that were dispensed during the emptying of the cash unit.
countedPhysCUs.counted	integer		The number of items that were counted during the emptying of the cash unit.

Name	Type	Default	Description
countedPhysCUs.pStatus	string		<p>Supplies the status of the physical cash unit. Following values are possible:</p> <p>"ok": The cash unit is in a good state.</p> <p>"full": The cash unit is full.</p> <p>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</p> <p>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</p> <p>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</p> <p>"inoperative": The cash unit is inoperative.</p> <p>"missing": The cash unit is missing.</p> <p>"noValue": The values of the specified cash unit are not available.</p> <p>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</p> <p>"manuellInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</p>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "countedPhysCUS": [
      {
        "physicalPositionName": "string",
        "unitId": "string",
        "dispensed": 0,
        "counted": 0,
        "pStatus": "ok"
      }
    ]
  }
}
```

## Event Messages

- [CashManagement.CashUnitErrorEvent](#)
  - [Dispenser.ItemsTakenEvent](#)
  - [Dispenser.ItemsPresentedEvent](#)
  - [CashManagement.NoteErrorEvent](#)
  - [CashManagement.InfoAvailableEvent](#)
  - [Dispenser.ShutterStatusChangedEvent](#)
- 

## Dispenser.PrepareDispense

---

### Description

On some hardware it can take a significant amount of time for the dispenser to get ready to dispense media. On this type of hardware the Dispenser.PrepareDispense command can be used to improve transaction performance.

If this command is supported (see the *prepareDispense* capability) then applications can help to improve the time taken to dispense media by issuing this command as soon as the application knows that a dispense is likely to happen. This command either prepares the device for the next dispense operation, or terminates the dispense preparation if the subsequent dispense operation is no longer required.

With the exception of the Dispenser.Denominate and Dispenser.Dispense commands, which will not stop the dispense preparation, any execute command on Dispenser or CashAcceptor will automatically stop the dispense preparation.

If this command is executed and the device is already in the specified *action* state, then this execution will have no effect and will complete with a successful completion message.

### Command Message

#### Message Header

---

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
action	string		<p>A value specifying the type of actions. Following values are possible:</p> <p>"start": Initiates the action to prepare for the next dispense command. This command does not wait until the device is ready to dispense before returning a completion event, it completes as soon as the preparation has been initiated.</p> <p>"stop": Stops the previously activated dispense preparation. For example the motor of the transport will be stopped. This should be used if for some reason the subsequent dispense operation is no longer required.</p>

#### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

# Unsolicited Events

## Dispenser.ItemsTakenEvent

---

### Description

This event is generated when items presented to the user have been taken. This event may be generated at any time.

### Message Header

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

### Message Payload

Name	Type	Default	Description
position	string		<p>The output position from which the items have been removed. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

### Example Message (generated)

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

## Dispenser.ItemsPresentedEvent

### Description

This event specifies that items have been presented to the user during a count operation and need to be taken.

### Message Header

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

### Example Message (generated)

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

## Dispenser.MediaDetectedEvent

### Description

This service event is generated if media is detected during a reset command. The payload on the event informs the application of the position of the media after the reset completes. If the device has been unable to successfully move the items found then this payload will be omitted.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found.
retractArea	object		This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker, or the transport.
retractArea.outputPosition	string		<p>Output position from which to retract the items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
retractArea.retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"itemCassette": Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</li> </ul>
retractArea.index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type "retractCassette" in command CashManagement.CashUnitInfo), <i>index</i> would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit. The maximum value of <i>index</i> is the sum of <i>maximum</i> of each retract cash unit. If <i>retractArea</i> is not set to "retract" the value of this field is ignored.</p>

Name	Type	Default	Description
outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>retractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "retractArea": {
      "outputPosition": "default",
      "retractArea": "retract",
      "index": 0
    },
    "outputPosition": "default"
  }
}
```

---

## Dispenser.ShutterStatusChangedEvent

---

### Description

Within the limitations of the hardware sensors this event is generated whenever the status of a shutter changes. The shutter status can change because of an explicit, implicit or manual operation depending on how the shutter is operated.

---

### Message Header

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
position	string		Specifies one of the Dispenser output positions whose shutter status has changed. Following values are possible:  "left": Left output position.  "right": Right output position.  "center": Center output position.  "top": Top output position.  "bottom": Bottom output position.  "front": Front output position.  "rear": Rear output position.

Name	Type	Default	Description
shutter	string		<p>Specifies the new state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is closed.</li> <li>"open": The shutter is opened.</li> <li>"jammed": The shutter is jammed.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> </ul>

#### Example Message (generated)

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

## Events

### CashManagement.CashUnitErrorEvent

#### Description

This event is generated if there is a problem with a cash unit during the execution of a command.

#### Message Header

Name	Type	Default	Description
requestId	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.

Name	Type	Default	Description
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
failure	string		<p>Specifies the kind of failure that occurred in the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": Specified cash unit is empty.</li> <li>"error": Specified cash unit has malfunctioned.</li> <li>"full": Specified cash unit is full.</li> <li>"locked": Specified cash unit is locked.</li> <li>"invalid": Specified cash unit is invalid.</li> <li>"config": An attempt has been made to change the settings of a self-configuring cash unit.</li> <li>"notConfigured": Specified cash unit is not configured.</li> </ul>
cashUnit.	object		
cashUnit.number	integer		<p>Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.</p>

Name	Type	Default	Description
cashUnit.type	string		<p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> <li>"retractCassette": Retract cash unit.</li> <li>"coupon": Cash unit containing coupons or advertising material.</li> <li>"document": Cash unit containing documents.</li> <li>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</li> <li>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</li> <li>"cashIn": Cash-in cash unit.</li> </ul>
cashUnit.unitID	string		The Cash Unit Identifier.
cashUnit.currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
cashUnit.values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
cashUnit.count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
cashUnit.maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>ulCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> (<i>high</i>) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
cashUnit.status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"maneuellnsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
cashUnit.appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
cashUnit.physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnit.cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
cashUnit.initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
cashUnit.dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
cashUnit.rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
cashUnit.minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
cashUnit.	object		
cashUnit.itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
cashUnit.itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
cashUnit.itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
cashUnit.itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
cashUnit.itemType.level1	boolean		Level 1 note types are stored in this cash unit.

Name	Type	Default	Description
cashUnit.itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
cashUnit.itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
cashUnit.itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
cashUnit.itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashUnit.cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
cashUnit.noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
cashUnit.noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
cashUnit.noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
cashUnit.noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
cashUnit.noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
cashUnit.extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "failure": "empty",
    "cashUnit": {
      "number": 0,
      "type": "notApplicable",
      "unitID": "string",
      "currencyID": "string",
      "values": 0,
      "count": 0,
      "maximum": 0,
      "status": "ok",
      "appLock": true,
      "physical": [
        {
          "physicalPositionName": "string",
          "unitID": "string",
          "noteID": 0
        }
      ]
    }
  }
}
```

```
        "count": 0,
        "maximum": 0,
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    },
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
}
```

---

## CashManagement.CashUnitThresholdEvent

---

### Description

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

This user event is generated when a threshold condition has occurred in one of the logical cash units. This event can be triggered either by hardware sensors in the device or by the logical *count* reaching the *minimum* or *maximum* value as specified in the CashUnitInfo structure. The application can check if the device has hardware sensors by querying the *hardwareSensor* field of the physical cash unit structure. If any of the physical cash units associated with the logical cash unit have this capability then threshold events based on hardware sensors will be triggered if the *maximum* or *minimum* values are not used and are set to zero. In the situation where the cash unit is associated with multiple physical cash units the CashManagement.CashUnitInfoChangedEvent will be generated when any of the physical cash units reaches the threshold. When the final physical cash unit reaches the threshold, the CashManagement.CashUnitThresholdEvent as well as the CashManagement.CashUnitInfoChangedEvent event will be generated.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
number	integer	object	Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

Name	Type	Default	Description
			<p>Type of cash unit. Following values are possible:</p> <p>"notApplicable": Not applicable. Typically means cash unit is missing.</p> <p>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</p> <p>"billCassette": Cash unit containing bills.</p> <p>"coinCylinder": Coin cylinder.</p> <p>"coinDispenser": Coin dispenser as a whole unit.</p> <p>"retractCassette": Retract cash unit.</p> <p>"coupon": Cash unit containing coupons or advertising material.</p> <p>"document": Cash unit containing documents.</p> <p>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</p> <p>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</p> <p>"cashIn": Cash-in cash unit.</p>
type	string		The Cash Unit Identifier.
unitID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
currencyID	string		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
values	number		

Name	Type	Default	Description
count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
maximum	integer		<p>This field is only applicable to retract and reject cash units. When ulCount reaches this value the threshold event CashManagement.CashUnitThresholdEvent (<i>high</i>) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuelInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event CashManagement.CashUnitThresholdEvent ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
itemType.level1	boolean		Level 1 note types are stored in this cash unit.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0,
        "physicalType": "string"
      }
    ]
  }
}
```

```
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    ],
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
```

## Dispenser.DelayedDispenseEvent

### Description

This event is generated if the start of a dispense operation has been delayed.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
delay	integer		The time in milliseconds by which the dispense operation will be delayed.

### Example Message (generated)

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

---

## Dispenser.StartDispenseEvent

---

### Description

This event is generated when a delayed dispense operation begins.

### Message Header

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

### Message Payload

---

Name	Type	Default	Description
reqID	string		The requestId of the original dispense command.

#### Example Message (generated)

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

---

## Dispenser.PartialDispenseEvent

---

### Description

This event is generated when a dispense operation is divided into several sub-dispense operations because the hardware capacity of the Dispenser is exceeded.

### Message Header

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

### Message Payload

Name	Type	Default	Description
dispNuminteger			The number of sub-dispense operations into which the dispense operation has been divided.

#### Example Message (generated)

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

## Dispenser.SubDispenseOkEvent

### Description

This event is generated when one of the sub-dispense operations into which the dispense operation was divided has finished successfully. Note that in this case the values in the payload structure report the amount and number of each denomination dispensed in the sub-dispense operation.

### Message Header

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

### Message Payload

Name	Type	Default	Description
currencies	array		Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.
currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
currencies.amount	number		The amount to be denominated or dispensed.

Name	Type	Default	Description
values	array		This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands. The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.
cashBox	integer		Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "currencies": [
      {
        "currencyID": "string",
        "amount": 0
      }
    ],
    "values": [
      0
    ],
    "cashBox": 0
  }
}
```

---

**Dispenser.IncompleteDispenseEvent**

---

**Description**

This event is generated during Dispenser.Dispense when it has not been possible to dispense the entire denomination but part of the requested denomination is on the intermediate stacker or in customer access. Note that in this case the values in this structure report the amount and number of each denomination that are in customer access or on the intermediate stacker.  
Dispenser.PresentStatus can be used to determine whether the items are in customer access.

**Message Header**

Name	Type	Default	Description

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

### Message Payload

Name	Type	Default	Description
currencies	array		Array of currency and amount combinations for denomination. There will be one array item for each currency in the denomination.
currencies.currencyID	string		Identification of currency in ISO format [Ref. 2].
currencies.amount	number		The amount to be denominated or dispensed.
values	array		This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned by the last CashManagement.CashUnitInfo command or set by the last CashManagement.SetCashUnitInfo commands. The first value in the array is related to the cash structure with the index number 1. This array contains a field for each possible cash unit. If a cash unit is not required in the denomination its corresponding field in this array should be set to zero. If the application does not wish to specify a denomination, it should omit the values property.
cashBox	integer		Only applies to Teller Dispensers. Amount to be paid from the teller's cash box.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "currencies": [
      {
        "currencyID": "string",
        "amount": 0
      }
    ],
    "values": [
      0
    ],
    "cashBox": 0
  }
}
```

## CashManagement.NoteErrorEvent

### Description

This event specifies the reason for a note detection error during the execution of a command.

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
reason	string		The reason for the notes detection error. Following values are possible:  "doubleNote": Double notes have been detected.  "longNote": A long note has been detected.  "skewedNote": A skewed note has been detected.  "incorrectCount": An item counting error has occurred.  "notesTooClose": Notes have been detected as being too close.  "otherNoteError": An item error not covered by the other values has been detected.  "shortNote": Short notes have been detected.

#### Example Message (generated)

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

---

## CashManagement.InfoAvailableEvent

---

#### Description

This execute event is generated when information is available for items detected during the cash processing operation.

#### Message Header

---

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

#### Message Payload

Name	Type	Default	Description
itemInfoSummary	array		Array of itemInfoSummary objects, one object for every level.
itemInfoSummary.level	string		<p>Defines the note level. Following values are possible:</p> <ul style="list-style-type: none"> <li>"level1": Information for level 1 notes.</li> <li>"level2": Information for level 2 notes.</li> <li>"level3": Information for level 3 notes.</li> <li>"level4": Information for level 4 notes.</li> </ul>
itemInfoSummary.numOfItems	integer		Number of items classified as <i>level</i> which have information available.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "itemInfoSummary": [
      {
        "level": "level1",
        "numOfItems": 0
      }
    ]
  }
}
```

## Dispenser.IncompleteRetractEvent

### Description

This event is sent when a retract or reset command has completed with an error and not all of the items have been retracted.

### Message Header

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

### Message Payload

Name	Type	Default	Description
itemNumberList	object		The values in this structure report the amount and number of each denomination that were successfully moved during the command prior to the failure.
itemNumberList.itemNumber	array		Array of item number objects.

Name	Type	Default	Description
itemNumberList.itemNumber.currencyID	string		A three character array storing the ISO format [Ref. 2] Currency ID; if the currency of the item is not known this is omitted.
itemNumberList.itemNumber.values	number		The value of a single item expressed as floating point value; or a zero value if the value of the item is not known.
itemNumberList.itemNumber.release	integer		The release of the item. The higher this number is, the newer the release. Zero means that there is only one release or the release is not known. This value has not been standardized and therefore a release number of the same item will not necessarily have the same value in different systems.
itemNumberList.itemNumber.count	integer		The count of items of the same type moved to the same destination during the execution of this command.
itemNumberList.itemNumber.number	integer		The logical number of the cash unit which received items during the execution of this command. This value will be zero if items were moved to the <i>retractArea</i> "transport" or "stacker".

Name	Type	Default	Description
reason	string		<p>The reason for not having retracted items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retractFailure": The retract has partially failed for a reason not covered by the other reasons listed in this event, for example failing to pick an item to be retracted.</li> <li>"retractAreaFull": The specified retract area (see input parameter <i>retractArea</i>) has become full during the retract operation.</li> <li>"foreignItemsDetected": Foreign items have been detected.</li> <li>"invalidBunch": An invalid bunch of items has been detected, e.g. it is too large or could not be processed.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "itemNumberList": {
      "itemNumber": [
        {
          "currencyID": "string",
          "values": 0,
          "release": 0,
          "count": 0,
          "number": 0
        }
      ]
    },
    "reason": "retractFailure"
  }
}
```

## CashManagement.CashUnitInfoChangedEvent

---

### Description

This service event is generated under the following circumstances:

- It is generated whenever *status* and/or *pStatus* changes. For instance, a physical cash unit has been removed or inserted, or a physical/logical cash unit has become empty or full.
- This event will also be generated for every cash unit changed in any way (including changes to counts, e.g. *count*, *rejectCount*, *initialCount*, *dispensedCount* and *presentedCount*) as a result of the following commands:

CashManagement.SetCashUnitInfo > CashManagement.EndExchange

- This event will also be fired when any change is made to a cash unit by the following commands, except for changes to counts (e.g. *count*, *rejectCount*, *initialCount*, *dispensedCount* and *presentedCount*):

Dispenser.CalibrateCashUnit > Dispenser.TestCashUnit

- In addition this event will be generated when a cash unit has been counted during the CashAcceptor.CashUnitCount command execution. When a physical cash unit is removed, the status of the physical cash unit becomes "missing". If there are no physical cash units of the same logical type remaining the status of the logical type becomes "missing". When a physical cash unit is inserted and this physical cash unit is of an existing logical type, both the logical and the physical cash unit structures will be updated. If a physical cash unit of a new logical type is inserted the cash unit structure reported by the last CashManagement.CashUnitInfo command is no longer valid. In that case an application should issue a CashManagement.CashUnitInfo command after receiving this event to obtain updated cash unit information.

### Message Header

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

### Message Payload

Name	Type	Default	Description
number	integer		Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

Name	Type	Default	Description
			<p>Type of cash unit. Following values are possible:</p> <p>"notApplicable": Not applicable. Typically means cash unit is missing.</p> <p>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</p> <p>"billCassette": Cash unit containing bills.</p> <p>"coinCylinder": Coin cylinder.</p> <p>"coinDispenser": Coin dispenser as a whole unit.</p> <p>"retractCassette": Retract cash unit.</p> <p>"coupon": Cash unit containing coupons or advertising material.</p> <p>"document": Cash unit containing documents.</p> <p>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</p> <p>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</p> <p>"cashIn": Cash-in cash unit.</p>
type	string		The Cash Unit Identifier.
unitID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
currencyID	string		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
values	number		

Name	Type	Default	Description
count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>uiCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> (<i>high</i>) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuelInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul> <p>If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.</p>
physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event CashManagement.CashUnitThresholdEvent ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
itemType.level1	boolean		Level 1 note types are stored in this cash unit.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0,
        "physicalType": "string"
      }
    ]
  }
}
```

XFS4  
risk  
All rig

```
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    ],
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
```

## XFS4IoT Preview Messaging for CashManagement Draft 0.0.3

This specification describes the functionality of an XFS4IoT compliant Cash Management interface. It defines the service-specific commands that can be issued to the service using the WebSocket endpoint.

This interface is to be used together with Dispenser and/or CashAcceptor interfaces to handle management of cash units, cash counts and banknote information.

## Commands

### CashManagement.CashUnitInfo

---

#### Description

This command is used to obtain information regarding the status and contents of cash units.

Where a logical cash unit is configured but there is no corresponding physical cash unit currently present in the device, information about the missing cash unit will still be returned in the *list* field of the completion message. The status of the cash unit will be reported as *missing*.

It is possible that one logical cash unit may be associated with more than one physical cash unit. In this case, the number of cash unit structures returned in *cashUnitInfo* will reflect the number of logical cash units. That is, if a system contains four physical cash units but two of these are treated as one logical cash unit, *cashUnitInfo* will contain information about the three logical cash units. Information about the physical cash unit(s) associated with a logical cash unit is contained in the *physical* objects representing the logical cash unit.

It is also possible that multiple logical cash units may be associated with one physical cash unit. This should only occur if the physical cash unit is capable of handling this situation, i.e. if it can store multiple denominations and report meaningful count and replenishment information for each denomination or if it can store retracted and rejected items as separate logical units and report meaningful count and replenishment information for each of them. In this case the information returned in *cashUnitInfo* will again reflect the number of logical cash units in the CDM.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
list	object		Array of cash unit objects.

### Example Message (generated)

```
{  
    "headers": {  
        "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
        "type": "command",  
        "name": "string"  
    },  
    "payload": {  
        "completionCode": "success",  
        "errorDescription": "string",  
        "list": [  
            {  
                "number": 0,  
                "type": "notApplicable",  
                "unitID": "string",  
                "currencyID": "string",  
                "values": 0,  
                "count": 0,  
                "maximum": 0,  
                "status": "ok",  
                "appLock": true,  
                "physical": [  
                    {  
                        "physicalPositionName": "string",  
                        "unitID": "string",  
                        "count": 0,  
                        "maximum": 0,  
                        "pStatus": "ok",  
                        "hardwareSensor": true,  
                        "initialCount": 0,  
                        "dispensedCount": 0,  
                        "presentedCount": 0,  
                        "retractedCount": 0,  
                        "rejectCount": 0,  
                        "cashInCount": 0,  
                        "extra": [  
                            "string"  
                        ]  
                    }  
                ],  
                "cashUnitName": "string",  
                "initialCount": 0,  
                "dispensedCount": 0,  
                "presentedCount": 0,  
                "retractedCount": 0,  
                "rejectCount": 0,  
                "minimum": 0,  
                "itemType": {  
                    "all": true,  
                    "unfit": true,  
                    "individual": true,  
                    "level1": true,  
                    "level2": true,  
                    "level3": true,  
                    "itemProcessor": true,  
                    "unfitIndividual": true  
                },  
                "cashInCount": 0,  
                "noteNumberList": {  
                    "list": [  
                        "string"  
                    ]  
                }  
            }  
        ]  
    }  
}
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
specification Preview Version 0.3
  "noteNumber": [
    {
      "noteID": 0,
      "count": 0
    }
  ],
  "noteIDs": [
    0
  ],
  "extra": [
    "string"
  ]
}
]
}
}
```

## Event Messages

## CashManagement.TellerInfo

## Description

This command only applies to Teller devices. It allows the application to obtain counts for each currency assigned to the teller. These counts represent the total amount of currency dispensed by the teller in all transactions. This command also enables the application to obtain the position assigned to each teller. If the input parameter is NULL, this command will return information for all tellers and all currencies. The teller information is persistent.

## **Command Message**

## Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
tellerID	integer		Identification of the teller. If the value of <i>tellerID</i> is not valid the error <i>invalidTellerID</i> is reported.
currencyID	string		Three character ISO format currency identifier [Ref 2].

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
tellerDetails	array		Array of teller detail objects.
tellerDetails.tellerID	integer		Identification of the teller.
			The input position assigned to the teller for cash entry. Following values are possible:
			"none": No position is assigned to the teller.
			"left": Left position is assigned to the teller.
			"right": Right position is assigned to the teller.
tellerDetails.inputPosition	string		"center": Center position is assigned to the teller.
			"top": Top position is assigned to the teller.
			"bottom": Bottom position is assigned to the teller.
			"front": Front position is assigned to the teller.
			"rear": Rear position is assigned to the teller.

Name	Type	Default	Description
tellerDetails.outputPosition	string		<p>The output position from which cash is presented to the teller. Following values are possible:</p> <ul style="list-style-type: none"> <li>"none": No position is assigned to the teller.</li> <li>"left": Left position is assigned to the teller.</li> <li>"right": Right position is assigned to the teller.</li> <li>"center": Center position is assigned to the teller.</li> <li>"top": Top position is assigned to the teller.</li> <li>"bottom": Bottom position is assigned to the teller.</li> <li>"front": Front position is assigned to the teller.</li> <li>"rear": Rear position is assigned to the teller.</li> </ul>
tellerDetails.tellerTotals	array		Array of teller total objects
tellerDetails.tellerTotals.currencyID	string		Three character ISO format currency identifier [Ref. 2].
tellerDetails.tellerTotals.itemsReceived	number		The total amount of items (other than coins) of the specified currency accepted. The amount is expressed as floating point value.

Name	Type	Default	Description
tellerDetails.tellerTotals.itemsDispensed	number		The total amount of items (other than coins) of the specified currency dispensed. The amount is expressed as floating point value.
tellerDetails.tellerTotals.coinsReceived	number		The total amount of coin currency accepted. The amount is expressed as floating point value.
tellerDetails.tellerTotals.coinsDispensed	number		The total amount of coin currency dispensed. The amount is expressed as floating point value.
tellerDetails.tellerTotals.cashBoxReceived	number		The total amount of cash box currency accepted. The amount is expressed as floating point value.
tellerDetails.tellerTotals.cashBoxDispensed	number		The total amount of cash box currency dispensed. The amount is expressed as floating point value.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "tellerDetails": [
      {
        "tellerID": 0,
        "inputPosition": "none",
        "outputPosition": "none",
        "tellerTotals": [
          {
            "currencyID": "string",
            "itemsReceived": 0,
            "itemsDispensed": 0,
            "coinsReceived": 0,
            "coinsDispensed": 0,
            "cashBoxReceived": 0,
            "cashBoxDispensed": 0
          }
        ]
      }
    ]
  }
}
```

## Event Messages

---

## CashManagement.GetItemInfo

---

### Description

This command is used to get information about detected items. It can be used to get information about individual items, all items of a certain level, or all items that have information available. This information is available from the point where the first CashManagement.InfoAvailableEvent event is generated until one of the following commands is executed:

CashAcceptor.CashInStart, CashAcceptor.CashIn, CashAcceptor.CashInRollback,  
 CashAcceptor.CashInEnd, CashAcceptor.Retract, CashAcceptor.Reset,  
 CashAcceptor.CreateP6Signature, CashAcceptor.Replenish, CashAcceptor.CashUnitCount.  
 Dispenser.Dispense, Dispenser.Count, Dispenser.Present, Dispenser.Retract, Dispenser.Reject,  
 Dispenser.OpenShutter, Dispenser.CloseShutter, Dispenser.Reset, CashManagement.StartExchange,  
 CashManagement.EndExchange, CashManagement.CalibrateCashUnit, Dispenser.TestCashUnits.

### Command Message

#### Message Header

Name	Type	Default	Description

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
level	object		<p>Defines the requested note level. Following values are possible:</p> <p>"level1": Information for level 1 notes. Only an image file can be retrieved for level 1 notes.</p> <p>"level2": Information for level 2 notes. On systems that do not classify notes as level 2 this value cannot be used and "invalidData" will be returned.</p> <p>"level3": Information for level 3 notes. On systems that do not classify notes as level 3 this value cannot be used and "invalidData" will be returned.</p> <p>"level4": Information for level 4 notes.</p> <p>"levelAll": Information for all levels and all items is to be returned with the <i>itemsList</i> output parameter.</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
index	integer		Specifies the index for the item information required. If no index is provided, all items of the specified /level/ will be returned. If /level/ is set to "levelAll", this property will be ignored.
itemInfoType	object		Specifies the type of information required. If nothing is specified, all available information will be returned.
itemInfoType.serialNumber	boolean		Serial number of the item.
itemInfoType.signature	boolean		Signature of the item.
itemInfoType.imageFile	boolean		Image file of the item.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "level": "level1",
    "index": 0,
    "itemInfoType": {
      "serialNumber": true,
      "signature": true,
      "imageFile": true
    }
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
itemsList	array		Array of "item info" objects.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "itemsList": [
      {
        "currencyID": "string",
        "value": 0,
        "release": 0,
        "level": "level1",
        "serialNumber": "string",
        "orientation": {
          "frontTop": true,
          "frontBottom": true,
          "backTop": true,
          "backBottom": true,
          "unknown": true,
          "notSupported": true
        },
        "p6Signature": "string",
        "imageFile": "string",
        "onClassificationList": "onClassificationList",
        "itemLocation": "device",
        "number": 0,
        "itemDeviceLocation": "stacker"
      }
    ]
  }
}
```

## CashManagement.GetClassificationList

---

### Description

This command is used to retrieve the entire note classification information pre-set inside the device or set via the CashManagement.SetClassificationList command. This provides the functionality to blacklist notes and allows additional flexibility, for example to specify that notes can be taken out of circulation by specifying them as unfit. Any items not returned in this list will be handled according to normal classification rules.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
timeoutinteger0	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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
	object		
version	string		This is an application defined string that sets the version identifier of the classification list. This property can be omitted if it has no version identifier.
classificationElements	array		Array of classification objects.
classificationElements.serialNumber	string		This string defines the serial number or a mask of serial numbers of one element with the defined currency and value. For a definition of the mask see Section 4 (TODO).
classificationElements.currencyID	string		The three character ISO format currency identifier [Ref. 2] of the element.

Name	Type	Default	Description
classificationElements.value	number		The value of the element. This field can be zero to represent all values.
classificationElements.level	string		<p>Specifies the note level. Following values are possible:</p> <ul style="list-style-type: none"> <li>"level1": The element specifies notes to be treated as level 1 notes.</li> <li>"level2": The element specifies notes to be treated as level 2 notes.</li> <li>"level3": The element specifies notes to be treated as level 3 notes.</li> <li>"level4": The element specifies notes to be treated as level 4 notes.</li> </ul>
classificationElements.unfit	boolean		Specifies whether the item is to be treated as unfit for dispensing. Applies only where <i>level</i> is "level4".

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "version": "string",
    "classificationElements": [
      {
        "serialNumber": "string",
        "currencyID": "string",
        "value": 0,
        "level": "level1",
        "unfit": true
      }
    ]
  }
}
```

## Event Messages

---

## CashManagement.SetTellerInfo

---

### Description

This command allows the application to initialize counts for each currency assigned to the teller. The values set by this command are persistent. This command only applies to Teller ATMs.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
action	string		<p>object</p> <p>The action to be performed. Following values are possible:</p> <ul style="list-style-type: none"> <li>"createTeller": A teller is to be added.</li> <li>"modifyTeller": Information about an existing teller is to be modified.</li> <li>"deleteTeller": A teller is to be removed.</li> </ul>
tellerDetails	object		Teller details object.
tellerDetails.tellerID	integer		Identification of the teller.

Name	Type	Default	Description
tellerDetails.inputPosition	string		<p>The input position assigned to the teller for cash entry. Following values are possible:</p> <ul style="list-style-type: none"> <li>"none": No position is assigned to the teller.</li> <li>"left": Left position is assigned to the teller.</li> <li>"right": Right position is assigned to the teller.</li> <li>"center": Center position is assigned to the teller.</li> <li>"top": Top position is assigned to the teller.</li> <li>"bottom": Bottom position is assigned to the teller.</li> <li>"front": Front position is assigned to the teller.</li> <li>"rear": Rear position is assigned to the teller.</li> </ul>

Name	Type	Default	Description
tellerDetails.outputPosition	string		<p>The output position from which cash is presented to the teller. Following values are possible:</p> <ul style="list-style-type: none"> <li>"none": No position is assigned to the teller.</li> <li>"left": Left position is assigned to the teller.</li> <li>"right": Right position is assigned to the teller.</li> <li>"center": Center position is assigned to the teller.</li> <li>"top": Top position is assigned to the teller.</li> <li>"bottom": Bottom position is assigned to the teller.</li> <li>"front": Front position is assigned to the teller.</li> <li>"rear": Rear position is assigned to the teller.</li> </ul>
tellerDetails.tellerTotals	array		Array of teller total objects
tellerDetails.tellerTotals.currencyID	string		Three character ISO format currency identifier [Ref. 2].
tellerDetails.tellerTotals.itemsReceived	number		The total amount of items (other than coins) of the specified currency accepted. The amount is expressed as floating point value.

Name	Type	Default	Description
tellerDetails.tellerTotals.itemsDispensed	number		The total amount of items (other than coins) of the specified currency dispensed. The amount is expressed as floating point value.
tellerDetails.tellerTotals.coinsReceived	number		The total amount of coin currency accepted. The amount is expressed as floating point value.
tellerDetails.tellerTotals.coinsDispensed	number		The total amount of coin currency dispensed. The amount is expressed as floating point value.
tellerDetails.tellerTotals.cashBoxReceived	number		The total amount of cash box currency accepted. The amount is expressed as floating point value.
tellerDetails.tellerTotals.cashBoxDispensed	number		The total amount of cash box currency dispensed. The amount is expressed as floating point value.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "action": "createTeller",
    "tellerDetails": {
      "tellerID": 0,
      "inputPosition": "none",
      "outputPosition": "none",
      "tellerTotals": [
        {
          "currencyID": "string",
          "itemsReceived": 0,
          "itemsDispensed": 0,
          "coinsReceived": 0,
          "coinsDispensed": 0,
          "cashBoxReceived": 0,
          "cashBoxDispensed": 0
        }
      ]
    }
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [CashManagement.TellerInfoChangedEvent](#)
- 

## CashManagement.SetCashUnitInfo

---

### Description

This command is used to adjust information about the status and contents of the cash units present in the device. Only fields that are to be changed need to be set in the payload of this command. Values that are not meant to change, can be omitted.

This command generates the CashManagement.CashUnitInfoCahngedEvent to inform applications that cash unit information has been changed.

This command can be used to change software counters, thresholds and the application lock. Other fields in the input structure will only take effect, if used in exchange state.

The following fields of the may be updated by this command outside of the exchange state:

*count  
cashInCount  
maximum  
appLock  
noteNumberList (contents must be consistent with ulCount)  
initialCount  
dispensedCount  
presentedCount  
retractedCount  
rejectCount  
minimum*

As may the following fields of the physical structure:

*cashInCount  
count  
initialCount  
dispensedCount  
presentedCount  
retractedCount  
rejectCount*

Any other changes must be performed while in an exchange state.

---

*The physical counts must be consistent with the logical cash unit counts. The Service controls whether*

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

the logical counts are maintained separately or are based on the sum of the physical counts.

If the fields *count* and *cashInCount* of *physical* are set to zero by this command, the application is indicating that it does not wish counts to be maintained for the physical cash units. Counts on the logical cash units will still be maintained and can be used by the application. If the physical counts are set by this command then the logical count will be the sum of the physical counts and any value sent as a logical count will be ignored.

The values set by this command are persistent.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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.
list	array		Array of cash unit objects.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "list": [
      {
        "number": 0,
        "type": "notApplicable",
        "unitID": "string",
        "currencyID": "string",
        "values": 0,
        "count": 0,
        "maximum": 0,
        "status": "ok",
        "appLock": true,
        "physical": [
          {
            "logical": 0
          }
        ]
      }
    ]
  }
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0,
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    },
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
]
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitInfoChangedEvent](#)
- [CashManagement.CashUnitErrorEvent](#)

## CashManagement.OpenSafeDoor

### Description

This command unlocks the safe door or starts the time delay count down prior to unlocking the safe door, if the device supports it. The command completes when the door is unlocked or the timer has started.

### Command Message

#### Message Header

Name	Type	Default	Description
------	------	---------	-------------

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

### CashManagement.StartExchange

---

#### Description

This command puts the ATM in an exchange state, i.e. a state in which cash units can be emptied, replenished, removed or replaced. Other than the updates which can be made via the CashManagement.SetCashUnitInfo command, outside of an exchange state, all changes to a cash unit must take place while the cash unit is in an exchange state.

The command returns current cash unit information in the form described in the documentation of the CashManagement.CashUnitInfo command. This command will also initiate any physical processes which may be necessary to make the cash units accessible. Before using this command an application should first have obtained exclusive control of the CashManagement interface.

This command may return a successful completion even if CashManagement.CashUnitErrorEvents are generated. If this command returns a successful completion the device is in an exchange state.

In Exchange state the CashManagement.SetCashUnitInfo command can be used multiple times to adjust the cash unit information, until the CashManagement.EndExchange command is performed.

While in an exchange state the device will process all requests, excluding cash related commands other than CashManagement.EndExchange, Dispenser.SetMixTable and Reset commands (e.g. Dispenser.Reset).

Any other command will result in the error "exchangeActive" being generated.

If an error is returned by this command, the CashManagement.CashUnitInfo command should be used to determine the cash unit information.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

Name	Type	Default	Description
name <b>(Required)</b>	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.
exchangeType	string		<p>Specifies the type of cash unit exchange operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"byHand": The cash units will be replenished manually either by filling or emptying the cash unit by hand or by replacing the cash unit.</li> <li>"toCassettes": Items will be moved from the replenishment container to the bill cash units.</li> <li>"clearRecycler": Items will be moved from a recycle cash unit to a cash unit or output position.</li> <li>"depositInto": Items will be moved from the deposit entrance to the bill cash units. See section 8.16 (TODO) for an example flow.</li> </ul>
tellerID	integer		Identifies the teller. If the device is a Self-Service ATM this field is ignored.
cUNumList	array		Array of integers containing the logical numbers of the cash units to be exchanged.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
output	object		This field is used when the <i>exchangeType</i> is "clearRecycler", i.e. a recycle cash unit is to be emptied.
output.logicalNumber	integer		Logical number of recycle cash unit to be emptied.
output.position	object		Determines to which position the cash should be moved as a combination of the following flags:
output.position.default	boolean		Move items to a cash unit. If no cash unit is specified in <i>number</i> , use the default output position.
output.position.left	boolean		Move items to the left output position.
output.position.right	boolean		Move items to the right output position.
output.position.center	boolean		Move items to the center output position.
output.position.top	boolean		Move items to the top output position.
output.position.bottom	boolean		Move items to the bottom output position.
output.position.front	boolean		Move items to the front output position.
output.position.rear	boolean		Move items to the rear output position.
output.number	integer		Logical number of the cash unit the items are to be moved to.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "exchangeType": "byHand",
    "tellerID": 0,
    "cUNumList": [
      0
    ],
    "output": {
      "logicalNumber": 0,
      "position": {
        "default": true,
        "left": true,
        "right": true,
        "center": true,
        "top": true,
        "bottom": true,
        "front": true,
        "rear": true
      },
      "number": 0
    }
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string	object	If not success, then this is optional vendor dependent information to provide additional information

Name	Type	Default	Description
list	array		Array of cash unit objects.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "list": [
      {
        "number": 0,
        "type": "notApplicable",
        "unitID": "string",
        "currencyID": "string",
        "values": 0,
        "count": 0,
        "maximum": 0,
        "status": "ok",
        "appLock": true,
        "physical": [
          {
            "physicalPositionName": "string",
            "unitID": "string",
            "count": 0,
            "maximum": 0,
            "pStatus": "ok",
            "hardwareSensor": true,
            "initialCount": 0,
            "dispensedCount": 0,
            "presentedCount": 0,
            "retractedCount": 0,
            "rejectCount": 0,
            "cashInCount": 0,
            "extra": [
              "string"
            ]
          }
        ],
        "cashUnitName": "string",
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "minimum": 0,
        "itemType": {
          "all": true,
          "unfit": true,
          "individual": true,
          "level1": true,
          "level2": true
        }
      }
    ],
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
      "all": true,
      "unfit": true,
      "individual": true,
      "level1": true,
      "level2": true
    }
  }
}
```

```
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
]
```

## Event Messages

- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitInfoChangedEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [CashAcceptor.ShutterStatusChangedEvent](#)

---

## CashManagement.EndExchange

---

### Description

This command will end the exchange state. If any physical action took place as a result of the CashManagement.StartExchange command then this command will cause the cash units to be returned to their normal physical state, including depositing any remaining items where *exchangeType* is "depositInto". Any necessary device testing will also be initiated.

CashManagement.SetCashUnitInfo does not need to be called if the Service can obtain cash unit information from self-configuring cash units.

If an error occurs during the execution of this command, then the application must issue a CashManagement.CashUnitInfo to determine the cash unit information.

**A CashManagement.CashUnitErrorEvent will be sent for any logical cash unit which cannot be successfully updated. If no cash units could be updated then a error code will be returned and CashManagement.CashUnitErrorEvent events generated for every logical cash unit that could not be updated.**

Even if this command does not return a successful completion the exchange state has ended.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

#### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitInfoChangedEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)

## CashManagement.CalibrateCashUnit

#### Description

This command will cause a vendor dependent sequence of hardware events which will calibrate one or more physical cash units associated with a logical cash unit. This is necessary if a new type of bank note is put into the cash unit as the command enables the ATM to obtain the measures of the new bank notes.

If more than one physical cash unit is associated with the cash unit, it is up to the Service to determine whether all the physical cash units need to be calibrated or if it is sufficient to calibrate for one physical unit and load the data into the others.

This command cannot be used to calibrate cash units which have been locked by the application. A error code will be returned and a CashManagement.CashUnitErrorEvent generated.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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	string		The message type, either command, response, event or completion.
name	string		(Required) 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.
number	integer		The logical number of the cash unit.
numOfBills	integer		The number of bills to be dispensed during the calibration process.
position	object		Specifies where the dispensed items should be moved to.
position.number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found.
position.retractArea	object		This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker, or the transport.

Name	Type	Default	Description
position.retractArea.outputPosition	string		<p>Output position from which to retract the items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>
position.retractArea.retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"itemCassette": Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</li> </ul>

Name	Type	Default	Description
position.retractArea.index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type "retractCassette" in command CashManagement.CashUnitInfo), <i>index</i> would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit. The maximum value of <i>index</i> is the sum of <i>maximum</i> of each retract cash unit. If <i>retractArea</i> is not set to "retract" the value of this field is ignored.</p>
position.outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>retractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "number": 0,
    "numOfBills": 0,
    "position": {
      "number": 0,
      "retractArea": {
        "outputPosition": "default",
        "retractArea": "retract",
        "index": 0
      },
      "outputPosition": "default"
    }
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
number	integer		The logical number of the cash unit.

Name	Type	Default	Description
numOfBills	integer		The number of bills to be dispensed during the calibration process.
position	object		Specifies where the dispensed items should be moved to.
position.number	integer		If non-zero, this value specifies the <i>number</i> (as specified by CashManagement.CashUnitInfo) of the single cash unit to be used for the storage of any items found.
position.retractArea	object		This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker, or the transport.
position.retractArea.outputPosition	string		<p>Output position from which to retract the items. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration information should be used.</li> <li>"left": Retract items from the left output position.</li> <li>"right": Retract items from the right output position.</li> <li>"center": Retract items from the center output position.</li> <li>"top": Retract items from the top output position.</li> <li>"bottom": Retract items from the bottom output position.</li> <li>"front": Retract items from the front output position.</li> <li>"rear": Retract items from the rear output position.</li> </ul>

Name	Type	Default	Description
position.retractArea.retractArea	string		<p>This value specifies the area to which the items are to be retracted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"retract": Retract the items to a retract cash unit.</li> <li>"transport": Retract the items to the transport.</li> <li>"stacker": Retract the items to the intermediate stacker area.</li> <li>"reject": Retract the items to a reject cash unit.</li> <li>"itemCassette": Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</li> </ul>
position.retractArea.index	integer		<p>If <i>retractArea</i> is set to "retract" this field defines the position inside the retract cash units into which the cash is to be retracted. <i>index</i> starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type "retractCassette" in command CashManagement.CashUnitInfo), <i>index</i> would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit. The maximum value of <i>index</i> is the sum of <i>maximum</i> of each retract cash unit. If <i>retractArea</i> is not set to "retract" the value of this field is ignored.</p>

Name	Type	Default	Description
position.outputPosition	string		<p>The output position to which items are to be moved. This field is only used if <i>number</i> is zero and <i>retractArea</i> is omitted. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "number": 0,
    "numOfBills": 0,
    "position": {
      "number": 0,
      "retractArea": {
        "outputPosition": "default",
        "retractArea": "retract",
        "index": 0
      },
      "outputPosition": "default"
    }
  }
}
```

#### Event Messages

- [CashManagement.CashUnitThresholdEvent](#)
- [CashManagement.CashUnitInfoChangedEvent](#)
- [CashManagement.CashUnitErrorEvent](#)
- [CashManagement.NoteErrorEvent](#)
- [CashManagement.InfoAvailableEvent](#)
- [Dispenser.ItemsTakenEvent](#)

## CashManagement.SetClassificationList

### Description

This command is used to specify the entire note classification list. Any items not specified in this list will be handled according to normal classification rules. This information is persistent. Information set by this command overrides any existing classification list. If a note is reclassified, it is handled as though it was a note of the new classification. For example, a fit note reclassified as unfit would be treated as though it were unfit, which may mean that the note is not dispensed. Reclassification cannot be used to change a note's classification to a higher level, for example, a note recognized as counterfeit by the device cannot be reclassified as genuine. In addition, it is not possible to re-classify a level 2 note as level 1. If two or more classification elements specify overlapping note definitions, but different *level* values then the first one takes priority.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
version	string		This is an application defined string that sets the version identifier of the classification list. This property can be omitted if it has no version identifier.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
classificationElements	array		Array of classification objects.
classificationElements.serialNumber	string		This string defines the serial number or a mask of serial numbers of one element with the defined currency and value. For a definition of the mask see Section 4 (TODO).
classificationElements.currencyID	string		The three character ISO format currency identifier [Ref. 2] of the element.
classificationElements.value	number		The value of the element. This field can be zero to represent all values.
classificationElements.level	string		<p>Specifies the note level. Following values are possible:</p> <ul style="list-style-type: none"> <li>"level1": The element specifies notes to be treated as level 1 notes.</li> <li>"level2": The element specifies notes to be treated as level 2 notes.</li> <li>"level3": The element specifies notes to be treated as level 3 notes.</li> <li>"level4": The element specifies notes to be treated as level 4 notes.</li> </ul>
classificationElements.unfit	boolean		Specifies whether the item is to be treated as unfit for dispensing. Applies only where <i>level</i> is "level4".

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "version": "string",
    "classificationElements": [
      {
        "serialNumber": "string",
        "currencyID": "string",
        "value": 0,
        "level": "level1",
        "unfit": true
      }
    ]
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

# Unsolicited Events

## CashManagement.SafeDoorOpenEvent

---

### Description

This event specifies that the safe door has been opened.

### Message Header

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

### Example Message (generated)

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

---

## CashManagement.SafeDoorClosedEvent

---

### Description

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This event specifies that the safe door has been closed.

### Message Header

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

### Example Message (generated)

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

---

## CashManagement.CashUnitInfoChangedEvent

---

### Description

This service event is generated under the following circumstances:

- It is generated whenever *status* and/or *pStatus* changes. For instance, a physical cash unit has been removed or inserted, or a physical/logical cash unit has become empty or full.
- This event will also be generated for every cash unit changed in any way (including changes to counts, e.g. *count*, *rejectCount*, *initialCount*, *dispensedCount* and *presentedCount*) as a result of the following commands:

CashManagement.SetCashUnitInfo > CashManagement.EndExchange

- This event will also be fired when any change is made to a cash unit by the following commands, except for changes to counts (e.g. *count*, *rejectCount*, *initialCount*, *dispensedCount* and *presentedCount*):

Dispenser.CalibrateCashUnit > Dispenser.TestCashUnit

- In addition this event will be generated when a cash unit has been counted during the CashAcceptor.CashUnitCount command execution. When a physical cash unit is removed, the status of the physical cash unit becomes "missing". If there are no physical cash units of the same logical type remaining the status of the logical type becomes "missing". When a physical cash unit is inserted and this physical cash unit is of an existing logical type, both the logical and the physical cash unit structures will be updated. If a physical cash unit of a new logical type is inserted the cash unit structure reported by the last CashManagement.CashUnitInfo command is no longer valid. In that case an application should issue a CashManagement.CashUnitInfo command after receiving this event to obtain updated cash unit information.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
number	integer		<p>Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.</p> <p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> </ul>
type	string		<p>"retractCassette": Retract cash unit.</p> <p>"coupon": Cash unit containing coupons or advertising material.</p> <p>"document": Cash unit containing documents.</p> <p>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</p> <p>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</p> <p>"cashIn": Cash-in cash unit.</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
unitID	string		The Cash Unit Identifier.
currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
count	integer		The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units ( <i>type</i> is not <i>retractCassette</i> ) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units ( <i>type</i> is <i>billCassette</i> , <i>coinCylinder</i> , <i>coinDispenser</i> , <i>coupon</i> , <i>document</i> or <i>recycling</i> ), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface ( <i>type</i> is <i>recycling</i> , <i>cashIn</i> , <i>retractCassette</i> or <i>rejectCassette</i> ) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit ( <i>type</i> is <i>rejectCassette</i> ), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit ( <i>type</i> is <i>retractCassette</i> ) this value specifies the number of retract operations which result in items entering the cash unit.

Name	Type	Default	Description
maximum	integer		<p>This field is only applicable to retract and reject cash units. When ulCount reaches this value the threshold event CashManagement.CashUnitThresholdEvent (<i>high</i>) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>
status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuellInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
appLock	boolean		<p>If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.</p>
physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent</i> ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
itemType.level1	boolean		Level 1 note types are stored in this cash unit.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0
      }
    ]
  }
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
        "maximum": 0,
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    },
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
}
```

---

## CashManagement.TellerInfoChangedEvent

---

### Description

This service event is generated when the counts assigned to a teller have changed. This event is only returned as a result of a [CashManagement.SetTellerInfo](#) command.

---

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
tellerID	integer		Integer holding the ID of the teller whose counts have changed.

#### Example Message (generated)

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

## CashManagement.CountsChangedEvent

#### Description

Deprecated

#### Message Header

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

#### Message Payload

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
cUNumList	array		Array of the number values of the cash units whose counts have changed.

#### Example Message (generated)

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

## CashManagement.CashUnitThresholdEvent

### Description

This user event is generated when a threshold condition has occurred in one of the logical cash units. This event can be triggered either by hardware sensors in the device or by the logical *count* reaching the *minimum* or *maximum* value as specified in the CashUnitInfo structure. The application can check if the device has hardware sensors by querying the *hardwareSensor* field of the physical cash unit structure. If any of the physical cash units associated with the logical cash unit have this capability then threshold events based on hardware sensors will be triggered if the *maximum* or *minimum* values are not used and are set to zero. In the situation where the cash unit is associated with multiple physical cash units the CashManagement.CashUnitInfoChangedEvent will be generated when any of the physical cash units reaches the threshold. When the final physical cash unit reaches the threshold, the CashManagement.CashUnitThresholdEvent as well as the CashManagement.CashUnitInfoChangedEvent event will be generated.

### Message Header

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

### Message Payload

Name	Type	Default	Description
	object		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
number	integer		<p>Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.</p>
type	string		<p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> <li>"retractCassette": Retract cash unit.</li> <li>"coupon": Cash unit containing coupons or advertising material.</li> <li>"document": Cash unit containing documents.</li> <li>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</li> <li>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</li> <li>"cashIn": Cash-in cash unit.</li> </ul>
unitID	string		The Cash Unit Identifier.
currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
count	integer		The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units ( <i>type</i> is not <i>retractCassette</i> ) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units ( <i>type</i> is <i>billCassette</i> , <i>coinCylinder</i> , <i>coinDispenser</i> , <i>coupon</i> , <i>document</i> or <i>recycling</i> ), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface ( <i>type</i> is <i>recycling</i> , <i>cashIn</i> , <i>retractCassette</i> or <i>rejectCassette</i> ) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit ( <i>type</i> is <i>rejectCassette</i> ), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit ( <i>type</i> is <i>retractCassette</i> ) this value specifies the number of retract operations which result in items entering the cash unit.
maximum	integer		This field is only applicable to retract and reject cash units. When uICount reaches this value the threshold event CashManagement.CashUnitThresholdEvent ( <i>high</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.

Name	Type	Default	Description
status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"manuelInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
physical	array		Array of pyhiscal cash unit objects.

Name	Type	Default	Description
cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.
initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a <i>type</i> of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

Name	Type	Default	Description
rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units ( <i>type</i> is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.
minimum	integer		This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event CashManagement.CashUnitThresholdEvent ( <i>low</i> ) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.
itemType	object		Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)
itemType.all	boolean		The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.
itemType.unfit	boolean		The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.
itemType.individual	boolean		The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.
itemType.level1	boolean		Level 1 note types are stored in this cash unit.

Name	Type	Default	Description
itemType.level2	boolean		If notes can be classified as level 2, then level 2 note types are stored in this cash unit.
itemType.level3	boolean		If notes can be classified as level 3, then level 3 note types are stored in this cash unit.
itemType.itemProcessor	boolean		The cash unit can accept items on the ItemProcessor interface.
itemType.unfitIndividual	boolean		The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.
cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>• If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>• If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>

Name	Type	Default	Description
noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.
noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "number": 0,
    "type": "notApplicable",
    "unitID": "string",
    "currencyID": "string",
    "values": 0,
    "count": 0,
    "maximum": 0,
    "status": "ok",
    "appLock": true,
    "physical": [
      {
        "physicalPositionName": "string",
        "unitID": "string",
        "count": 0,
        "maximum": 0,
        "physicalType": "string"
      }
    ]
  }
}
```

```
        "pStatus": "ok",
        "hardwareSensor": true,
        "initialCount": 0,
        "dispensedCount": 0,
        "presentedCount": 0,
        "retractedCount": 0,
        "rejectCount": 0,
        "cashInCount": 0,
        "extra": [
            "string"
        ]
    ],
    "cashUnitName": "string",
    "initialCount": 0,
    "dispensedCount": 0,
    "presentedCount": 0,
    "retractedCount": 0,
    "rejectCount": 0,
    "minimum": 0,
    "itemType": {
        "all": true,
        "unfit": true,
        "individual": true,
        "level1": true,
        "level2": true,
        "level3": true,
        "itemProcessor": true,
        "unfitIndividual": true
    },
    "cashInCount": 0,
    "noteNumberList": {
        "noteNumber": [
            {
                "noteID": 0,
                "count": 0
            }
        ]
    },
    "noteIDs": [
        0
    ],
    "extra": [
        "string"
    ]
}
```

## Events

### CashManagement.CashUnitErrorEvent

#### Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

This event is generated if there is a problem with a cash unit during the execution of a command.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
failure	string		<p>Specifies the kind of failure that occurred in the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": Specified cash unit is empty.</li> <li>"error": Specified cash unit has malfunctioned.</li> <li>"full": Specified cash unit is full.</li> <li>"locked": Specified cash unit is locked.</li> <li>"invalid": Specified cash unit is invalid.</li> <li>"config": An attempt has been made to change the settings of a self-configuring cash unit.</li> <li>"notConfigured": Specified cash unit is not configured.</li> </ul>
cashUnit.	object		Index number of the cash unit structure.
cashUnit.number	integer		Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

Name	Type	Default	Description
cashUnit.type	string		<p>Type of cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notApplicable": Not applicable. Typically means cash unit is missing.</li> <li>"rejectCassette": Reject cash unit. This type will also indicate a combined reject/retract cash unit.</li> <li>"billCassette": Cash unit containing bills.</li> <li>"coinCylinder": Coin cylinder.</li> <li>"coinDispenser": Coin dispenser as a whole unit.</li> <li>"retractCassette": Retract cash unit.</li> <li>"coupon": Cash unit containing coupons or advertising material.</li> <li>"document": Cash unit containing documents.</li> <li>"replenishmentContainer": Replenishment container. A cash unit can be refilled from a replenishment container.</li> <li>"recycling": Recycling cash unit. This unit is only present when the device implements the Dispenser and CashAcceptor interfaces.</li> <li>"cashIn": Cash-in cash unit.</li> </ul>
cashUnit.unitID	string		The Cash Unit Identifier.
cashUnit.currencyID	string		A three character string storing the ISO format [Ref. 2] Currency ID. This value will be omitted for cash units which contain items of more than one currency type or items to which currency is not applicable. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.
cashUnit.values	number		Supplies the value of a single item in the cash unit. This value is expressed as floating point value. If the <i>currencyID</i> field for this cash unit is omitted, then this field will contain zero. If the <i>status</i> field for this cash unit is <i>noValue</i> it is the responsibility of the application to assign a value to this field. This value is persistent.

Name	Type	Default	Description
cashUnit.count	integer		<p>The meaning of this count depends on the type of cash unit. This value is persistent. For all cash units except retract cash units (<i>type</i> is not <i>retractCassette</i>) this value specifies the number of items inside all the physical cash units associated with this cash unit. For all dispensing cash units (<i>type</i> is <i>billCassette</i>, <i>coinCylinder</i>, <i>coinDispenser</i>, <i>coupon</i>, <i>document</i> or <i>recycling</i>), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected. If the cash unit is usable from the CashAcceptor interface (<i>type</i> is <i>recycling</i>, <i>cashIn</i>, <i>retractCassette</i> or <i>rejectCassette</i>) then this value will be incremented as a result of a cash-in operation. Note that for a reject cash unit (<i>type</i> is <i>rejectCassette</i>), this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit (<i>type</i> is <i>retractCassette</i>) this value specifies the number of retract operations which result in items entering the cash unit.</p>
cashUnit.maximum	integer		<p>This field is only applicable to retract and reject cash units. When <i>ulCount</i> reaches this value the threshold event <i>CashManagement.CashUnitThresholdEvent (high)</i> will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>

Name	Type	Default	Description
cashUnit.status	string		<p>Supplies the status of the cash unit. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The cash unit is in a good state.</li> <li>"full": The cash unit is full.</li> <li>"high": The cash unit is almost full (i.e. reached or exceeded the threshold defined by <i>maximum</i>).</li> <li>"low": The cash unit is almost empty (i.e. reached or below the threshold defined by <i>minimum</i>).</li> <li>"empty": The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations.</li> <li>"inoperative": The cash unit is inoperative.</li> <li>"missing": The cash unit is missing.</li> <li>"noValue": The values of the specified cash unit are not available.</li> <li>"noReference": There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.</li> <li>"maneuverInsertion": The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.</li> </ul>
cashUnit.appLock	boolean		If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a CashManagement.CashUnitErrorEvent event will be generated and an error completion message will be returned. This value is persistent.
cashUnit.physical	array		Array of pyhiscal cash unit objects.
cashUnit.cashUnitName	string		A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type <i>document</i> where different documents can have the same currency and value. For example, travelers checks and bank checks may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the property can be omitted. This value is persistent.

Name	Type	Default	Description
cashUnit.initialCount	integer		Initial number of items contained in the cash unit. This value is persistent.
cashUnit.dispensedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.presentedCount	integer		The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a type of <i>rejectCassette</i> or <i>retractCassette</i> . This value is persistent.
cashUnit.retractedCount	integer		The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.
cashUnit.rejectCount	integer		The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and a Reject command has been called. For reject and retract cash units (type is <i>rejectCassette</i> or <i>retractCassette</i> ) this field does not apply and will be reported as zero. This value is persistent.

Name	Type	Default	Description
cashUnit.minimum	integer		<p>This field is not applicable to retract and reject cash units. For all other cash units, when <i>count</i> reaches this value the threshold event CashManagement.CashUnitThresholdEvent (/low) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if <i>hardwareSensor</i> is TRUE. This value is persistent.</p>
cashUnit.	object		
cashUnit.itemType	object		<p>Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags (TODO: include Table)</p>
cashUnit.itemType.all	boolean		<p>The cash unit takes all fit banknote types. These are level 4 notes which are fit for recycling.</p>
cashUnit.itemType.unfit	boolean		<p>The cash unit takes all unfit banknotes. These are level 4 notes which are unfit for recycling.</p>
cashUnit.itemType.individual	boolean		<p>The cash unit takes all types of fit banknotes specified in an individual list. These are level 4 notes which are fit for recycling.</p>
cashUnit.itemType.level1	boolean		<p>Level 1 note types are stored in this cash unit.</p>
cashUnit.itemType.level2	boolean		<p>If notes can be classified as level 2, then level 2 note types are stored in this cash unit.</p>
cashUnit.itemType.level3	boolean		<p>If notes can be classified as level 3, then level 3 note types are stored in this cash unit.</p>
cashUnit.itemType.itemProcessor	boolean		<p>The cash unit can accept items on the itemProcessor interface.</p>
cashUnit.itemType.unfitIndividual	boolean		<p>The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.</p>

Name	Type	Default	Description
cashUnit.cashInCount	integer		Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. This value is persistent.
cashUnit.noteNumberList	object		<p>Array of cash items inside the cash unit. The content of this structure is persistent. If the cash unit is Dispenser specific cash unit with type <i>billCassette</i> or the contents of the cash unit are not known this structure will be omitted. If the cash unit is of type <i>retractCassette</i> this pointer will be omitted except for the following cases:</p> <ul style="list-style-type: none"> <li>If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of level 2 notes.</li> <li>If items are recognized during retract operations then the number and type of notes retracted is returned in <i>noteNumberList</i> and <i>count</i> contains the number of retract operations. <i>cashInCount</i> contains the actual number of retracted items.</li> </ul>
cashUnit.noteNumberList.noteNumber	array		Array of banknote numbers the cash unit contains.
cashUnit.noteNumberList.noteNumber.noteID	integer		Identification of note type. The Note ID represents the note identifiers reported by the <i>CashAcceptor.BanknoteTypes</i> command. If this value is zero then the note type is unknown.
cashUnit.noteNumberList.noteNumber.count	integer		Actual count of cash items. The value is incremented each time cash items are moved to a cash unit. In the case of recycle cash units this count is decremented as defined in the description of the logical <i>count</i> field.

Name	Type	Default	Description
cashUnit.noteIDs	array		Array of integers which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to <i>individual</i> cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as <i>individual</i> then <i>noteIDs</i> will be omitted.
cashUnit.extra	array		Pointer to a list of vendor-specific information about the logical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "failure": "empty",
    "cashUnit": {
      "number": 0,
      "type": "notApplicable",
      "unitID": "string",
      "currencyID": "string",
      "values": 0,
      "count": 0,
      "maximum": 0,
      "status": "ok",
      "appLock": true,
      "physical": [
        {
          "physicalPositionName": "string",
          "unitID": "string",
          "count": 0,
          "maximum": 0,
          "pStatus": "ok",
          "hardwareSensor": true,
          "initialCount": 0,
          "dispensedCount": 0,
          "presentedCount": 0,
          "retractedCount": 0,
          "rejectCount": 0,
          "cashInCount": 0,
          "extra": [
            "string"
          ]
        }
      ],
      "cashUnitName": "string",
      "initialCount": 0,
      "maximum": 0
    }
  }
}
```

```
"dispensedCount": 0,  
"presentedCount": 0,  
"retractedCount": 0,  
"rejectCount": 0,  
"minimum": 0,  
"itemType": {  
    "all": true,  
    "unfit": true,  
    "individual": true,  
    "level1": true,  
    "level2": true,  
    "level3": true,  
    "itemProcessor": true,  
    "unfitIndividual": true  
},  
"cashInCount": 0,  
"noteNumberList": {  
    "noteNumber": [  
        {  
            "noteID": 0,  
            "count": 0  
        }  
    ]  
},  
"noteIDs": [  
    0  
],  
"extra": [  
    "string"  
]  
}  
}  
}
```

## CashManagement.NoteErrorEvent

### Description

This event specifies the reason for a note detection error during the execution of a command.

### Message Header

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

### Message Payload

Name	Type	Default	Description
reason	string		<p>The reason for the notes detection error.</p> <p>Following values are possible:</p> <ul style="list-style-type: none"> <li>"doubleNote": Double notes have been detected.</li> <li>"longNote": A long note has been detected.</li> <li>"skewedNote": A skewed note has been detected.</li> <li>"incorrectCount": An item counting error has occurred.</li> <li>"notesTooClose": Notes have been detected as being too close.</li> <li>"otherNoteError": An item error not covered by the other values has been detected.</li> <li>"shortNote": Short notes have been detected.</li> </ul>

#### Example Message (generated)

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

---

## CashManagement.InfoAvailableEvent

---

### Description

This execute event is generated when information is available for items detected during the cash processing operation.

### Message Header

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
itemInfoSummary	array		Array of itemInfoSummary objects, one object for every level.
itemInfoSummary.level	string		Defines the note level. Following values are possible: "level1": Information for level 1 notes. "level2": Information for level 2 notes. "level3": Information for level 3 notes. "level4": Information for level 4 notes.
itemInfoSummary.numOfItems	integer		Number of items classified as <i>level</i> which have information available.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "itemInfoSummary": [
      {
        "level": "level1",
        "numOfItems": 0
      }
    ]
  }
}
```

## CashAcceptor.ShutterStatusChangedEvent

### Description

Within the limitations of the hardware sensors this service event is generated whenever the status of a shutter changes. The shutter status can change because of an explicit, implicit or manual operation depending on how the shutter is operated.

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Specifies one of the input or output positions whose shutter status has changed. Following values are possible:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>

Name	Type	Default	Description
shutter	string		<p>Specifies the new state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is closed.</li> <li>"open": The shutter is opened.</li> <li>"jammed": The shutter is jammed.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> </ul>

#### Example Message (generated)

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

## Dispenser.ItemsTakenEvent

### Description

This event is generated when items presented to the user have been taken. This event may be generated at any time.

### Message Header

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

**Message Payload**

Name	Type	Default	Description
position	string		<p>The output position from which the items have been removed. Following values are possible:</p> <ul style="list-style-type: none"> <li>"default": The default configuration.</li> <li>"left": The left output position.</li> <li>"right": The right output position.</li> <li>"center": The center output position.</li> <li>"top": The top output position.</li> <li>"bottom": The bottom output position.</li> <li>"front": The front output position.</li> <li>"rear": The rear output position.</li> </ul>

**Example Message (generated)**

XFS4  
risk  
All rig

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

# XFS4IOT Common Interface Draft 0.0.3

TODO

## Commands

### Common.Status

---

#### Description

This command is used to obtain the overall status of any XFS4IoT service. The status includes common status information and can include zero or more interface specific status objects, depending on the implemented interfaces of the service. It may also return vendor-specific status information.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
common <b>(Required)</b>			Status information common to all XFS4IoT services.
common.device	string		Specifies the state of the device.
common.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.
common.guideLights	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.
common.guideLights.flashRate	string		Indicates the current flash rate of the guidelight.
common.guideLights.color	string		Indicates the current color of the guidelight.
common.guideLights.direction	string		Indicates the current direction of the guidelight.
common.devicePosition	string		Position of the device.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.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
common.antiFraudModule	string		Specifies the state of the anti-fraud module
cardReader.			
cardReader.timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
cardReader.			

Name	Type	Default	Description
cardReader.media	string		<p>Specifies the media state of the device as one of the following values. This status is independent of any media in the parking stations.</p> <p><b>notSupported</b>&lt;br&gt; Capability to report media position is not supported by the device (e.g. a typical swipe reader or contactless chip card reader).</p> <p><b>unknown</b>&lt;br&gt; The media state cannot be determined with the device in its current state (e.g. the value of device is noDevice, powerOff, offline or hwerror).</p> <p><b>present</b>&lt;br&gt; Media is present in the device, not in the entering position and not jammed. A card in a parking station is not considered to be present. On the latched dip device, this indicates that the card is present in the device and the card is unlatched.</p> <p><b>notPresent</b>&lt;br&gt; Media is not present in the device and not at the entering position.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device; operator intervention is required.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of a motorized device.</p> <p><b>latched</b>&lt;br&gt; Media is present and latched in a latched dip card unit. This means the card can be used for chip card dialog.</p>
cardReader.retainBin	string		Specifies the state of the retain bin.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cardReader.security	string		<p>Specifies the state of the security unit as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; No security module is available.</p> <p><b>notReady</b>&lt;br&gt; The security module is not ready to process cards or is inoperable.</p> <p><b>notPresent</b>&lt;br&gt; The security module is open and ready to process cards.</p>
cardReader.numberCards	integer		<p>The number of cards retained; applicable only to motor driven card units, for non-motorized card units this value is zero. This value is persistent it is reset to zero by the resetCount command.</p>

Name	Type	Default	Description
cardReader.chipPower	string		<p>Specifies the state of the chip controlled by this service. Depending on the value of capabilities response, this can either be the chip on the currently inserted user card or the chip on a permanently connected chip card. The state of the chip is one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability to report the state of the chip is not supported by the ID card unit device. This value is returned for contactless chip card readers.</p> <p><b>unknown</b>&lt;br&gt; The state of the chip cannot be determined with the device in its current state.</p> <p><b>online</b>&lt;br&gt; The chip is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).</p> <p><b>busy</b>&lt;br&gt; The chip is present, powered on, and busy (unable to process an Execute command at this time).</p> <p><b>poweredOff</b>&lt;br&gt; The chip is present, but powered off (i.e. not contacted).</p> <p><b>noDevice</b>&lt;br&gt; A card is currently present in the device, but has no chip.</p> <p><b>hwerror</b>&lt;br&gt; The chip is present, but inoperable due to a hardware error that prevents it from being used (e.g. MUTE, if there is an unresponsive card in the reader).</p> <p><b>noCard</b>&lt;br&gt; There is no card in the device.</p>
cashAcceptor	object		Status information for XFS4IoT services implementing the CashAcceptor interface. This will be omitted if the CashAcceptor interface is not supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty.</li> <li>"full": The intermediate stacker is full. This may also be reported during a cash-in transaction where a limit specified by CashAcceptor.SetCashInLimit has been reached.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>
cashAcceptor.stackertItems	string		<p>This field informs the application whether items on the intermediate stacker have been in customer access. Following values are possible:</p> <ul style="list-style-type: none"> <li>"customerAccess": Items on the intermediate stacker have been in customer access. If the device is a cash recycler then the items on the intermediate stacker may be there as a result of a previous cash-out operation.</li> <li>"noCustomerAccess": Items on the intermediate stacker have not been in customer access.</li> <li>"accessUnknown": It is not known if the items on the intermediate stacker have been in customer access.</li> <li>"noItems": There are no items on the intermediate stacker or the physical device has no intermediate stacker.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.banknoteReader	string		<p>Supplies the state of the banknote reader. Following values are possible:</p> <p>"ok": The banknote reader is in a good state.</p> <p>"inoperable": The banknote reader is inoperable.</p> <p>"unknown": Due to a hardware error or other condition, the state of the banknote reader cannot be determined.</p> <p>"notSupported": The physical device has no banknote reader.</p>
cashAcceptor.dropBox	boolean		<p>The drop box is an area within the CashAcceptor where items which have caused a problem during an operation are stored. This field specifies the status of the drop box. TRUE means that some items are stored in the drop box due to a cash-in transaction which caused a problem. FALSE indicates that the drop box is empty.</p>
cashAcceptor.positions	array		<p>Array of structures for each position from which items can be accepted.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.position	string		<p>Supplies the input or output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul>
cashAcceptor.positions.positionStatus	string		<p>The status of the input or output position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>
cashAcceptor.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous dispense operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>
cashAcceptor.mixedMode	string		<p>Reports if Mixed Media mode is active. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notActive": Mixed Media transactions are not supported by the device or Mixed Media mode is not activated.</li> <li>"active": Mixed Media mode using the CashAcceptor and ItemProcessor interfaces is activated.</li> </ul>
cashDispenser	object		<p>Status information for XFS4IoT services implementing the CashDispenser interface. This will be omitted if the CashDispenser interface is not supported.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. These bills are typically present on the intermediate stacker as a result of a retract operation or because a dispense has been performed without a subsequent present.</p> <p>Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty. The items have not been in customer access.</li> <li>"notEmptyCustomer": The intermediate stacker is not empty. The items have been in customer access. If the device is a recycler then the items on the intermediate stacker may be there as a result of a previous cash-in operation.</li> <li>"notEmptyUnknown": The intermediate stacker is not empty. It is not known if the items have been in customer access.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>
cashDispenser.positions	array		Array of structures for each position to which items can be dispensed or presented.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.position	string		<p>Supplies the output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"left": Left output position.</li> <li>"right": Right output position.</li> <li>"center": Center output position.</li> <li>"top": Top output position.</li> <li>"bottom": Bottom output position.</li> <li>"front": Front output position.</li> <li>"rear": Rear output position.</li> </ul>
cashDispenser.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.positionStatus	string		<p>Returns information regarding items which may be at the output position. If the device is a recycler it is possible that the output position will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul> <p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>
cashDispenser.positions.transport	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>
cashDispenser.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement	object		<p>Status information for XFS4IoT services implementing the CashManagement interface. This will be omitted if the CashManagement interface is not supported.</p>
cashManagement.safeDoor	string		<p>Supplies the state of the safe door. Following values are possible:</p> <ul style="list-style-type: none"> <li>"doorNotSupported": Physical device has no safe door or safe door state reporting is not supported.</li> <li>"doorOpen": Safe door is open.</li> <li>"doorClosed": Safe door is closed.</li> <li>"doorUnknown": Due to a hardware error or other condition, the state of the safe door cannot be determined.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement.dispenser	string		<p>Supplies the state of the logical cash units for dispensing. Following values are possible:</p> <p>"ok": All cash units present are in a good state.</p> <p>"cashUnitState": One or more of the cash units is in a low, empty, inoperative or manipulated condition. Items can still be dispensed from at least one of the cash units.</p> <p>"cashUnitStop": Due to a cash unit failure dispensing is impossible. No items can be dispensed because all of the cash units are in an empty, inoperative or manipulated condition. This state may also occur when a reject/retract cash unit is full or no reject/retract cash unit is present, or when an application lock is set on every cash unit which can be locked.</p> <p>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement.acceptor	string		<p>Supplies the state of the cash units for accepting cash. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": All cash units present are in a good state.</li> <li>"cashUnitState": One or more of the cash units is in a high, full, inoperative or manipulated condition. Items can still be accepted into at least one of the cash units.</li> <li>"cashUnitStop": Due to a cash unit failure accepting is impossible. No items can be accepted because all of the cash units are in a full, inoperative or manipulated condition. This state may also occur when a retract cash unit is full or no retract cash unit is present, or when an application lock is set on every cash unit, or when Level 2/3 notes are to be automatically retained within cash units, but all of the designated cash units for storing them are full or inoperative.</li> <li>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</li> </ul>
pinPad			Status information for XFS4IoT services implementing the PinPad interface. This will be omitted if the PinPad interface is not supported.
crypto	object		Status information for XFS4IoT services implementing the Crypto interface. This will be omitted if the Crypto interface is not supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
keyManagement	object		Status information for XFS4IoT services implementing the KeyManagement interface. This will be omitted if the KeyManagement interface is not supported.
keyboard	object		Status information for XFS4IoT services implementing the Keyboard interface. This will be omitted if the Keyboard interface is not supported.
textTerminal	object		Status information for XFS4IoT services implementing the TextTerminal interface. This will be omitted if the TextTerminal interface is not supported.
printer	object		Status information for XFS4IoT services implementing the Printer interface. This will be omitted if the Printer interface is not supported.

Name	Type	Default	Description
printer.media	string		<p>Specifies the state of the print media (i.e. receipt, statement, passbook, etc.) as one of the following values. This field does not apply to journal printers:</p> <p><b>notSupported</b>&lt;br&gt; The capability to report the state of the print media is not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; The state of the print media cannot be determined with the device in its current state.</p> <p><b>present</b>&lt;br&gt; Media is in the print position, on the stacker or on the transport (i.e. a passbook in the parking station is not considered to be present). On devices with continuous paper supplies, this value is set when paper is under the print head. On devices with no supply or individual sheet supplies, this value is set when paper/media is successfully inserted-loaded.</p> <p><b>notPresent</b>&lt;br&gt; Media is not in the print position or on the stacker.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of the device.</p> <p><b>retracted</b>&lt;br&gt; Media was retracted during the last command which controlled media.</p>

Name	Type	Default	Description
printer.paper	object		<p>Specifies the state of paper supplies as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The paper supply is full.</p> <p><b>low</b>&lt;br&gt; The paper supply is low.</p> <p><b>out</b>&lt;br&gt; The paper supply is empty.</p> <p><b>jammed</b>&lt;br&gt; The paper supply is jammed.</p>
printer.paper.upper	string		The state of the upper paper supply.
printer.paper.lower	string		The state of the lower paper supply.
printer.paper.external	string		The state of the external paper supply.
printer.paper.aux	string		The state of the auxiliary paper supply.
printer.paper.aux2	string		The state of the second auxiliary paper supply.
printer.paper.park	string		The state of the parking station paper supply.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
printer.toner	string		<p>Specifies the state of the toner or ink supply or the state of the ribbon as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of toner or ink supply or the ribbon cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The toner or ink supply is full or the ribbon is OK.</p> <p><b>low</b>&lt;br&gt; The toner or ink supply is low or the print contrast with a ribbon is weak.</p> <p><b>out</b>&lt;br&gt; The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.</p>
printer.ink	string		<p>Specifies the status of the stamping ink in the printer as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the stamping ink supply cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; Ink supply in device is full.</p> <p><b>low</b>&lt;br&gt; Ink supply in device is low.</p> <p><b>out</b>&lt;br&gt; Ink supply in device is empty.</p>

Name	Type	Default	Description
printer.lamp	string		<p>Specifies the status of the printer imaging lamp as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the imaging lamp cannot be determined with device in its current state.</p> <p><b>ok</b>&lt;br&gt; The lamp is OK.</p> <p><b>fading</b>&lt;br&gt; The lamp should be changed.</p> <p><b>inop</b>&lt;br&gt; The lamp is inoperative.</p>
printer.retractBins	array		<p>An array of bin state objects. If no retain bins are supported, the array will be empty.</p> <p>Specifies the state of the printer retract bin as one of the following:</p> <p><b>ok</b>&lt;br&gt; The retract bin of the printer is in a healthy state.</p> <p><b>full</b>&lt;br&gt; The retract bin of the printer is full.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>high</b>&lt;br&gt; The retract bin of the printer is nearly full.</p> <p><b>missing</b>&lt;br&gt; The retract bin is missing.</p>
printer.retractBins.state	string		
printer.retractBins.count	integer		<p>The number of media retracted to this bin. This value is persistent; it may be reset to zero by the Printer.ResetCount command.</p>
printer.mediaOnStacker	integer		<p>The number of media on stacker; applicable only to printers with stacking capability.</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
printer.paperType	object		<p>Specifies the type of paper loaded as one of the following:</p> <p><b>unknown</b>&lt;br&gt; No paper is loaded, reporting of this paper type is not supported or the paper type cannot be determined.</p> <p><b>single</b>&lt;br&gt; The paper can be printed on only one side.</p> <p><b>dual</b>&lt;br&gt; The paper can be printed on both sides.</p>
printer.paperType.upper	string		The upper paper supply paper type.
printer.paperType.lower	string		The lower paper supply paper type.
printer.paperType.external	string		The external paper supply paper type.
printer.paperType.aux	string		The auxiliary paper supply paper type.
printer.paperType.aux2	string		The second auxiliary paper supply paper type.
printer.paperType.park	string		The parking station paper supply paper type.
printer.blackMarkMode	string		<p>Specifies the status of the black mark detection and associated functionality:</p> <p><b>notSupported</b>&lt;br&gt; Black mark detection is not supported.</p> <p><b>unknown</b>&lt;br&gt; The status of the black mark detection cannot be determined.</p> <p><b>on</b>&lt;br&gt; Black mark detection and associated functionality is switched on.</p> <p><b>off</b>&lt;br&gt; Black mark detection and associated functionality is switched off.</p>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800a0-9ad2-4d50-89a2-92d1b13d554b"
  }
}
```

```
        "type": "command",
        "name": "string"
    },
    "payload": {
        "common": {
            "device": "online",
            "extra": [
                "string"
            ],
            "guideLights": [
                {}
            ],
            "devicePosition": "inposition",
            "powerSaveRecoveryTime": 0,
            "antiFraudModule": "notSupp"
        },
        "cardReader": {
            "timeout": "5000",
            "media": "notSupported",
            "retainBin": "notSupported",
            "security": "notSupported",
            "numberCards": 0,
            "chipPower": "notSupported"
        },
        "cashAcceptor": {
            "intermediateStacker": "empty",
            "stackerItems": "customerAccess",
            "banknoteReader": "customokerAccess",
            "dropBox": true,
            "positions": [
                {
                    "position": "inLeft",
                    "shutter": "closed",
                    "positionStatus": "empty",
                    "transport": "ok",
                    "transportStatus": "empty",
                    "jammedShutterPosition": "notSupported"
                }
            ],
            "mixedMode": "notActive"
        },
        "cashDispenser": {
            "intermediateStacker": "empty",
            "positions": [
                {
                    "position": "left",
                    "shutter": "closed",
                    "positionStatus": "empty",
                    "transport": "ok",
                    "transportStatus": "empty",
                    "jammedShutterPosition": "notSupported"
                }
            ]
        },
        "cashManagement": {
            "safeDoor": "doorNotSupported",
            "dispenser": "ok",
            "acceptor": "ok"
        },
        "pinPad": null,
        "crypto": {},
        "keyManagement": {},
        "keyboard": []
    }
}
```

```
    "textTerminal": {},
    "printer": {
        "media": "notSupported",
        "paper": {
            "upper": "notSupported",
            "lower": "notSupported",
            "external": "notSupported",
            "aux": "notSupported",
            "aux2": "notSupported",
            "park": "notSupported",
            "property1": "notSupported",
            "property2": "notSupported"
        },
        "toner": "notSupported",
        "ink": "notSupported",
        "lamp": "notSupported",
        "retractBins": [
            {
                "state": "unknown",
                "count": 0
            }
        ],
        "mediaOnStacker": 0,
        "paperType": {
            "upper": "unknown",
            "lower": "unknown",
            "external": "unknown",
            "aux": "unknown",
            "aux2": "unknown",
            "park": "unknown",
            "property1": "unknown",
            "property2": "unknown"
        },
        "blackMarkMode": "notSupported"
    }
}
}
```

## Event Messages

## Common.Capabilities

### Description

This command retrieves the capabilities of the device. It may also return vendor specific capability information.

### Command Message

#### Message Header

Name	Type	Default	Description
requestid <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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.

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

##### Message Payload

Name	Type	Default	Description
interfaces	(Required)	array	Array of interfaces supported by this XFS4IoT service.
interfaces.name	string		Name of supported XFS4IoT interface.
interfaces.commands	array		Full array of commands supported by this XFS4IoT interface.
interfaces.events	array		Full array of events supported by this XFS4IoT interface.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
interfaces.maximumRequests	integer		Specifies the maximum number of requests which can be queued by the Service. This will be omitted if no reported. This will be zero if the maximum number of requests is unlimited.
interfaces.authenticationRequired	array		Array of commands, which need to be authenticated using the security interface.
common <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	object		Capability information common to all XFS4IoT services.
common.serviceVersion	string		Specifies the Service Version.
common.deviceInformation	array		Array of deviceInformation structures. If the service uses more than one device there will be one array element for each device.
common.deviceInformation.modelName	string		Specifies the device model name. The property is omitted, if the device model name is unknown.
common.deviceInformation.serialNumber	string		Specifies the unique serial number of the device. The property is omitted, if the serial number is unknown.
common.deviceInformation.revisionNumber	string		Specifies the device revision number. The property is omitted, if the device revision number is unknown.
common.deviceInformation.modelDescription	string		Contains a description of the device. The property is omitted, if the model description is unknown.
common.deviceInformation.firmware	array		Array of firmware structures specifying the names and version numbers of the firmware that is present. Single or multiple firmware versions can be reported. If the firmware versions are not reported, then this property is omitted.
common.deviceInformation.firmware.firmwareName	string		Specifies the firmware name. The property is omitted, if the firmware name is unknown.
common.deviceInformation.firmware.firmwareVersion	string		Specifies the firmware version. The property is omitted, if the firmware version is unknown.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.deviceInformation.firmware.hardwareRevision	string		Specifies the hardware revision. The property is omitted, if the hardware revision is unknown.
common.deviceInformation.software	array		Array of software structures specifying the names and version numbers of the software components that are present. Single or multiple software versions can be reported. If the software versions are not reported, then this property is omitted.
common.deviceInformation.software.firmwareName	string		Specifies the firmware name. The property is omitted, if the firmware name is unknown.
common.deviceInformation.software.firmwareVersion	string		Specifies the firmware version. The property is omitted, if the firmware version is unknown.
common.deviceInformation.software.hardwareRevision	string		Specifies the hardware revision. The property is omitted, if the hardware revision is unknown.
common.vendorModelformation	object		Specifies additional information about the Service while in Vendor Dependent Mode. If omitted, all sessions must be closed before entry to VDM.
common.vendorModelformation.allowOpenSessions	boolean		If TRUE, sessions with this Service may remain open during Vendor Dependent Mode for the purposes of monitoring events, sending Info commands, or sending Execute commands listed in lpdwAllowedExecuteCommands. If FALSE, all sessions must be closed before entering Vendor Dependent Mode.
common.vendorModelformation.allowedExecuteCommands	array		Array of commands which can be accepted while in Vendor Dependent Mode. Any Execute command which is not included in this list will be rejected with a SequenceError as control of the device has been handed to the Vendor Dependent Application. If omitted, no Execute commands can be accepted.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.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
common.guideLights	array		Specifies which guidance lights are available
common.guideLights.flashRate	object		Indicates which flash rates are supported by the guidelight.
common.guideLights.flashRate.slow	boolean		The light can blink slowly.
common.guideLights.flashRate.medium	boolean		The light can blink medium frequency.
common.guideLights.flashRate.quick	boolean		The light can blink quickly.
common.guideLights.flashRate.continuous	boolean		The light can be continuous (steady).
common.guideLights.color	object		Indicates which colors are supported by the guidelight.
common.guideLights.color.red	boolean		The light can be red.
common.guideLights.color.green	boolean		The light can be green.
common.guideLights.color.yellow	boolean		The light can be yellow.
common.guideLights.color.blue	boolean		The light can be blue.
common.guideLights.color.cyan	boolean		The light can be cyan.
common.guideLights.color.magenta	boolean		The light can be magenta.
common.guideLights.color.white	boolean		The light can be white.
common.guideLights.direction	object		Indicates which directions are supported by the guidelight.
common.guideLights.direction.entry	boolean		The light can indicate entry.
common.guideLights.direction.exit	boolean		The light can indicate exit.
common.powerSaveControl	boolean		Specifies whether power saving control is available

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.antiFraudModule	boolean		Specifies whether the anti-fraud module is available
common.synchronizableCommands	array		list of commands support synchronization.
common.endToEndSecurity	boolean		True if this hardware supports End to End security, and requires security tokens as part of the data to secured operations. If true then operations may fail if a valid security token is not supplied.  If false then all operations can be performed without a security token.
common.hardwareSecurityElement	boolean		True if this hardware supports End to End security and has a Hardware Security Element which validates the security token. Otherwise false. If this valid is false it may mean that validation is performed in software, or that the device doesn't support End to End security.
common.responseSecurityEnabled	boolean		True if this device will return a security token as part of the response data to commands that support End to End security, for example, to validate the result of a dispense operation.
cardReader	object		Capability information for XFS4IoT services implementing the CardReader interface. This will be omitted if the CardReader interface is not supported.

Name	Type	Default	Description
cardReader.type	string		<p>Specifies the type of the ID card unit as one of the following:</p> <p><b>motor</b>&lt;br&gt; The ID card unit is a motor driven card unit.</p> <p><b>swipe</b>&lt;br&gt; The ID card unit is a swipe (pull-through) card unit.</p> <p><b>dip</b>&lt;br&gt; The ID card unit is a dip card unit. This dip type is not capable of latching cards entered.</p> <p><b>latchedDip</b>&lt;br&gt; The ID card unit is a latched dip card unit. This device type is used when a dip IDC device supports chip communication. The latch ensures the consumer cannot remove the card during chip communication. Any card entered will automatically latch when a request to initiate a chip dialog is made (via the <a href="#">CardReader.ReadRawData</a> command). The <a href="#">CardReader.EjectCard</a> command is used to unlatch the card.</p> <p><b>contactless</b>&lt;br&gt; The ID card unit is a contactless card unit, i.e. no insertion of the card is required.</p> <p><b>intelligentContactless</b>&lt;br&gt; The ID card unit is an intelligent contactless card unit, i.e. no insertion of the card is required and the card unit has built-in EMV or smart card application functionality that adheres to the EMVCo Contactless Specifications or individual payment system's specifications. The ID card unit is capable of performing both magnetic stripe emulation and EMV-like transactions.</p> <p><b>permanent</b>&lt;br&gt; The ID card unit is dedicated to a permanently housed chip card (no user interaction is available with this type of card).</p>
cardReader.readTracks	object		Specifies the tracks that can be read by the card reader.
cardReader.readTracks.track1	boolean		The card reader can access track 1.
cardReader.readTracks.track2	boolean		The card reader can access track 2.
cardReader.readTracks.track3	boolean		The card reader can access track 3.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cardReader.readTracks.watermark	boolean		The card reader can access the Swedish watermark track.
cardReader.readTracks.frontTrack1	boolean		The card reader can access front track 1.
cardReader.readTracks.frontImage	boolean		The card reader can read the front image of the card.
cardReader.readTracks.backImage	boolean		The card reader can read the back image of the card.
cardReader.readTracks.track1JIS	boolean		The card reader can access JIS I track 1.
cardReader.readTracks.track3JIS	boolean		The card reader can access JIS I track 3.
cardReader.readTracks.ddi	boolean		The card reader can provide dynamic digital identification of the magnetic strip.
cardReader.writeTracks	object		Specifies the tracks that can be read by the card reader.
cardReader.writeTracks.track1	boolean		The card reader can access track 1.
cardReader.writeTracks.track2	boolean		The card reader can access track 2.
cardReader.writeTracks.track3	boolean		The card reader can access track 3.
cardReader.writeTracks.frontTrack1	boolean		The card reader can access front track 1.
cardReader.writeTracks.track1JIS	boolean		The card reader can access JIS I track 1.
cardReader.writeTracks.track3JIS	boolean		The card reader can access JIS I track 3.
cardReader.chipProtocols	object		Specifies the chip card protocols that are supported by the card reader.
cardReader.chipProtocols.chipT0	boolean		The card reader can handle the T=0 protocol.
cardReader.chipProtocols.chipT1	boolean		The card reader can handle the T=0 protocol.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cardReader.chipProtocols.chipProtocolNotRequired	boolean		The carder is capable of communicating with the chip without requiring the application to specify any protocol.
cardReader.chipProtocols.chipTypeAPart3	boolean		The card reader can handle the ISO 14443 (Part3) Type A contactless chip card protocol.
cardReader.chipProtocols.chipTypeAPart4	boolean		The card reader can handle the ISO 14443 (Part4) Type A contactless chip card protocol.
cardReader.chipProtocols.chipTypeB	boolean		The card reader can handle the ISO 14443 Type B contactless chip card protocol.
cardReader.chipProtocols.chipTypeNFC	boolean		The card reader can handle the ISO 18092 (106/212/424kbps) contactless chip card protocol.
cardReader.maxCardCount	number		Specifies the maximum numbers of cards that the retain bin and card stacker module bin can hold (zero if not available).
cardReader.securityType	string		Specifies the type of security module.
cardReader.powerOnOption	string		<p>&lt;a name="capability-powerOnOption"&gt;&lt;/a&gt; Specifies the power-on (or off) capabilities of the device hardware as one of the following options (applicable only to motor driven ID card units):</p> <p><b>noAction</b>&lt;br&gt; No actions are supported by the device.</p> <p><b>eject</b>&lt;br&gt; The card will be ejected.</p> <p><b>retain</b>&lt;br&gt; The card will be retained.</p> <p><b>ejectThenRetain</b>&lt;br&gt; The card will be ejected for a finite time, then if not taken, the card will be retained. The time for which the card remains ejected is vendor dependent.</p> <p><b>readPosition</b>&lt;br&gt; The card will be moved to the read position.</p>
cardReader.powerOffOption	string		Specifies the power-off capabilities of the device hardware. See <a href="#">powerOnOption</a> .

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor	object		Capability information for XFS4IoT services implementing the CashAcceptor interface. This will be omitted if the CashAcceptor interface is not supported.
cashAcceptor.type	string		<p>Supplies the type of CashAcceptor. Following values are possible:</p> <ul style="list-style-type: none"> <li>"tellerBill": The CashAcceptor is a Teller Bill Acceptor.</li> <li>"selfServiceBill": The CashAcceptor is a Self-Service Bill Acceptor.</li> <li>"tellerCoin": The CashAcceptor is a Teller Coin Acceptor.</li> <li>"selfServiceCoin": The CashAcceptor is a Self-Service Coin Acceptor.</li> </ul>
cashAcceptor.maxCashInItems	integer		Supplies the maximum number of items that can be accepted in a single CashAcceptor.CashIn command. This value reflects the hardware limitations of the device and therefore it does not change as part of the CashAcceptor.SetCashInLimit command.
cashAcceptor.shutter	boolean		If this flag is TRUE then the device has a shutter and explicit shutter control through the commands OpenShutter and CloseShutter is supported. The definition of a shutter will depend on the h/w implementation. On some devices where items are automatically detected and accepted then a shutter is simply a latch that is opened and closed, usually under implicit control by the Service. On other devices, the term shutter refers to a door, which is opened and closed to allow the customer to place the items onto a tray. If a Service cannot detect when items are inserted and there is a shutter on the device, then it must provide explicit application control of the shutter.

Name	Type	Default	Description
cashAcceptor.shutterControl	boolean		If set to TRUE the shutter is controlled implicitly by the Service. If set to FALSE the shutter must be controlled explicitly by the application using the OpenShutter and the CloseShutter commands. In either case the PresentMedia command may be used if the presentControl field is reported as FALSE. The shutterControl field is always set to TRUE if the device has no shutter. This field applies to all shutters and all positions.
cashAcceptor.intermediateStacker	integer		Specifies the number of items the intermediate stacker for cash-in can hold. Zero means that there is no intermediate stacker for cash-in available.
cashAcceptor.itemsTakenSensor	boolean		Specifies whether or not the CashAcceptor can detect when items at the exit position are taken by the user. If set to TRUE the Service generates an accompanying CashAcceptor.ItemsTaken event. If set to FALSE this event is not generated. This field relates to all output positions.
cashAcceptor.itemsInsertedSensor	boolean		Specifies whether the CashAcceptor has the ability to detect when items have actually been inserted by the user. If set to TRUE the Service generates an accompanying CashAcceptor.ItemsInserted event. If set to FALSE this event is not generated. This field relates to all input positions. This flag should not be reported as TRUE unless item insertion can be detected.
cashAcceptor.positions	object		Specifies the CashAcceptor input and output positions which are available.
cashAcceptor.positions.inLeft	boolean		The CashAcceptor has a left input position.
cashAcceptor.positions.inRight	boolean		The CashAcceptor has a right input position.
cashAcceptor.positions.inCenter	boolean		The CashAcceptor has a center input position.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.inTop	boolean		The CashAcceptor has a top input position.
cashAcceptor.positions.inBottom	boolean		The CashAcceptor has a bottom input position.
cashAcceptor.positions.inFront	boolean		The CashAcceptor has a front input position.
cashAcceptor.positions.inRear	boolean		The CashAcceptor has a rear input position.
cashAcceptor.positions.outLeft	boolean		The CashAcceptor has a left output position.
cashAcceptor.positions.outRight	boolean		The CashAcceptor has a right output position.
cashAcceptor.positions.outCenter	boolean		The CashAcceptor has a center output position.
cashAcceptor.positions.outTop	boolean		The CashAcceptor has a top output position.
cashAcceptor.positions.outBottom	boolean		The CashAcceptor has a bottom output position.
cashAcceptor.positions.outFront	boolean		The CashAcceptor has a front output position.
cashAcceptor.positions.outRear	boolean		The CashAcceptor has a rear output position.
cashAcceptor.retractAreas	object		Specifies the area to which items may be retracted. If the device does not have a retract capability all flags will be set to false.
cashAcceptor.retractAreas.retract	boolean		The items may be retracted to a retract cash unit.
cashAcceptor.retractAreas.transport	boolean		The items may be retracted to the transport.
cashAcceptor.retractAreas.stack	boolean		The items may be retracted to the intermediate stacker.
cashAcceptor.retractAreas.reject	boolean		The items may be retracted to a reject cash unit.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.retractAreas.billCassette	boolean		The items may be retracted to the item cassettes, i.e. cash-in and recycle cash units.
cashAcceptor.retractAreas.cashIn	boolean		Items may be retracted to a cash-in cash unit.
cashAcceptor.retractTransportActions	object		Specifies the actions which may be performed on items which have been retracted to the transport. If the device does not have the capability to retract items to the transport or move items from the transport all flags will be set to false.
cashAcceptor.retractTransportActions.present	boolean		The items may be presented.
cashAcceptor.retractTransportActions.retract	boolean		The items may be moved to a retract cash unit.
cashAcceptor.retractTransportActions.reject	boolean		The items may be moved to a reject bin.
cashAcceptor.retractTransportActions.billCassette	boolean		The items may be moved to the item cassettes, i.e. cash-in and recycle cash units.
cashAcceptor.retractTransportActions.cashIn	boolean		Items may be retracted to a cash-in cash unit.
cashAcceptor.retractStackerActions	object		Specifies the actions which may be performed on items which have been retracted to the stacker. If the device does not have the capability to retract items to the stacker or move items from the stacker all flags will be set to false.
cashAcceptor.retractStackerActions.present	boolean		The items may be presented.
cashAcceptor.retractStackerActions.retract	boolean		The items may be moved to a retract cash unit.
cashAcceptor.retractStackerActions.reject	boolean		The items may be moved to a reject bin.
cashAcceptor.retractStackerActions.billCassette	boolean		The items may be moved to the item cassettes, i.e. cash-in and recycle cash units.
cashAcceptor.retractStackerActions.cashIn	boolean		Items may be retracted to a cash-in cash unit.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.compareSignatures	boolean		Specifies if the Service has the ability to compare signatures through command CashAcceptor.CompareP6Signature
cashAcceptor.replenish	boolean		If set to TRUE the CashAcceptor.ReplenishTarget and CashAcceptor.Replenish commands are supported.
cashAcceptor.cashInLimit	object		Specifies whether the cash-in limitation is supported or not for the CashAcceptor.SetCashInLimit command. If the device does not have the capability to limit the amount or the number of items during cash-in operations all flags will be set to false.
cashAcceptor.cashInLimit.byTotalItems	boolean		The number of successfully processed cash-in items can be limited by specifying the total number of items.
cashAcceptor.cashInLimit.byAmount	boolean		The number of successfully processed cash-in items can be limited by specifying the maximum amount of a specific currency.
cashAcceptor.cashInLimit.multiple	boolean		CashAcceptor.SetCashInLimit may be called multiple times in a cash-in transaction to update previously specified amount limits. Only valid if combined with "byAmount".
cashAcceptor.cashInLimit.refuseOther	boolean		If multiple currencies can be accepted and an amount limit is specified for one or more currencies any other unspecified currencies are refused. If not specified, there is no amount limit for unspecified currencies. Only valid if specified with "byAmount".
cashAcceptor.countActions	object		Specifies the count action supported by the CashAcceptor.CashUnitCount command. If the device does not support counting then all flags will be set to false.
cashAcceptor.countActions.individual	boolean		The counting of individual cash units via the input structure of the CashAcceptor.CashUnitCount command is supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.countActions.all	boolean		The counting of all cash units via the empty payload structure of the CashAcceptor.CashUnitCount command is supported.
cashAcceptor.deviceLockControl	boolean		Specifies whether the CashAcceptor supports physical lock/unlock control of the CashAcceptor device and/or the cash units. If this value is set to TRUE, the device and/or the cash units can be locked and unlocked by the CashAcceptor.DeviceLockControl command, and the lock status can be retrieved by the CashAcceptor.DeviceLockStatus command. If this value is set to FALSE, the CashAcceptor will not support the physical lock/unlock control of the CashAcceptor device or the cash units;
cashAcceptor.mixedMode	boolean		Specifies whether the device supports accepting and processing items other than the types defined in the CashAcceptor specification. For a description of Mixed Media transactions see section ATM Mixed Media Transaction Flow – Application Guidelines.
cashAcceptor.mixedDepositAndRollback	boolean		Specifies whether the device can deposit one type of media and rollback the other in the same Mixed Media transaction. Where mixedDepositAndRollback is TRUE the Service can accept CashAcceptor.CashInEnd and ItemProcessor.MediaInRollback or CashAcceptor.CashInRollback and ItemProcessor.MediaInEnd to complete the current transaction. This value can only be TRUE where mixedMode == TRUE. When mixedDepositAndRollback is FALSE applications must either deposit or return ALL items to complete a transaction. Where Mixed Media transactions are not supported mixedDepositAndRollback is FALSE.
cashAcceptor.deplete	boolean		If set to TRUE the CashAcceptor.Deplete command is supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser	object		Capability information for XFS4IoT services implementing the CashDispenser interface. This will be omitted if the CashDispenser interface is not supported.
cashDispenser.type	string		<p>Supplies the type of Dispenser. Following values are possible:</p> <ul style="list-style-type: none"> <li>"tellerBill": The Dispenser is a Teller Bill Dispenser.</li> <li>"selfServiceBill": The Dispenser is a Self-Service Bill Dispenser.</li> <li>"tellerCoin": The Dispenser is a Teller Coin Dispenser.</li> <li>"selfServiceCoin": The Dispenser is a Self-Service Coin Dispenser.</li> </ul>
cashDispenser.maxDispenseItems	integer		Supplies the maximum number of items that can be dispensed in a single dispense operation. If no limit applies this value will be zero - in this case, if an attempt is made to dispense more items than the hardware limitations will allow, the Service will implement the dispense as a series of sub-dispense operations (see section Sub-Dispensing Command Flow).
cashDispenser.shutter	boolean		Specifies whether or not the commands Dispenser.OpenShutter and Dispenser.CloseShutter are supported.
cashDispenser.shutterControl	boolean		If set to TRUE the shutter is controlled implicitly by the Service. If set to FALSE the shutter must be controlled explicitly by the application using the Dispenser.OpenShutter and the Dispenser.CloseShutter commands. This field is always set to TRUE if the device has no shutter. This field applies to all shutters and all output positions.
cashDispenser.retractAreas	object		Specifies the area to which items may be retracted. If the device does not have a retract capability all flags will be set to false.
cashDispenser.retractAreas.retract	boolean		The items may be retracted to a retract cash unit.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.retractAreas.transport	boolean		The items may be retracted to the transport.
cashDispenser.retractAreas.stackert	boolean		The items may be retracted to the intermediate stacker.
cashDispenser.retractAreas.reject	boolean		The items may be retracted to a reject cash unit.
cashDispenser.retractAreas.itemCassette	boolean		The items may be retracted to the item cassettes, i.e. cassettes that can be dispensed from.
cashDispenser.retractTransportActions	object		Specifies the actions which may be performed on items which have been retracted to the transport. If the device does not have the capability to retract items to the transport or move items from the transport all flags will be set to false.
cashDispenser.retractTransportActions.present	boolean		The items may be presented.
cashDispenser.retractTransportActions.retract	boolean		The items may be moved to a retract cash unit.
cashDispenser.retractTransportActions.reject	boolean		The items may be moved to a reject bin.
cashDispenser.retractTransportActions.itemCassette	boolean		The items may be moved to the item cassettes, i.e. cassettes that can be dispensed from.
cashDispenser.retractStackerActions	object		Specifies the actions which may be performed on items which have been retracted to the stacker. If the device does not have the capability to retract items to the stacker or move items from the stacker all flags will be set to false.
cashDispenser.retractStackerActions.present	boolean		The items may be presented.
cashDispenser.retractStackerActions.retract	boolean		The items may be moved to a retract cash unit.
cashDispenser.retractStackerActions.reject	boolean		The items may be moved to a reject bin.
cashDispenser.retractStackerActions.itemCassette	boolean		The items may be moved to the item cassettes, i.e. cassettes that can be dispensed from.

Name	Type	Default	Description
cashDispenser.intermediateStacker	boolean		Specifies whether or not the Dispenser supports stacking items to an intermediate position before the items are moved to the exit position. If this value is TRUE, the field "present" of the Dispenser.DISPENSE command can be set to FALSE.
cashDispenser.itemsTakenSensor	boolean		Specifies whether the Dispenser can detect when items at the exit position are taken by the user. If set to TRUE the Service generates an accompanying Dispenser.ItemsTakenEvent. If set to FALSE this event is not generated. This field applies to all output positions.
cashDispenser.positions	object		Specifies the Dispenser output positions which are available.
cashDispenser.positions.left	boolean		The Dispenser has a left output position.
cashDispenser.positions.right	boolean		The Dispenser has a right output position.
cashDispenser.positions.center	boolean		The Dispenser has a center output position.
cashDispenser.positions.top	boolean		The Dispenser has a top output position.
cashDispenser.positions.bottom	boolean		The Dispenser has a bottom output position.
cashDispenser.positions.front	boolean		The Dispenser has a front output position.
cashDispenser.positions.rear	boolean		The Dispenser has a rear output position.
cashDispenser.moveItems	object		Specifies the Dispenser move item options which are available.
cashDispenser.moveItems.fromCashUnit	boolean		The Dispenser can dispense items from the cash units to the intermediate stacker while there are items on the transport.
cashDispenser.moveItems.toCashUnit	boolean		The Dispenser can retract items to the cash units while there are items on the intermediate stacker.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.moveItems.toTransport	boolean		The Dispenser can retract items to the transport while there are items on the intermediate stacker.
cashDispenser.moveItems.toStacker	boolean		The Dispenser can dispense items from the cash units to the intermediate stacker while there are already items on the intermediate stacker that have not been in customer access. Items remaining on the stacker from a previous dispense may first need to be rejected explicitly by the application if they are not to be presented.
cashDispenser.prepareDispense	boolean		On some hardware it can take a significant amount of time for the dispenser to get ready to dispense media. On this type of hardware the Dispenser.PrepareDispense command can be used to improve transaction performance. This flag indicates if the hardware requires the application to use the Dispenser.PrepareDispense command to maximize transaction performance. If this flag is TRUE then the Dispenser.PrepareDispense command is supported and can be used to improve transaction performance. If this flag is FALSE then the Dispenser.PrepareDispense command is not supported.
cashManagement	object		Capability information for XFS4IoT services implementing the CashManagement interface. This will be omitted if the CashManagement interface is not supported.
cashManagement.safeDoor	boolean		Specifies whether or not the CashManagement.OpenSafeDoor command is supported.
cashManagement.cashBox	boolean		This field is only applicable to teller type devices. It specifies whether or not tellers have been assigned a cash box.
cashManagement.exchangeType	object		Specifies the type of cash unit exchange operations supported by the device.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement.exchangeType.byHand	boolean		The device supports manual replenishment either by filling the cash unit by hand or by replacing the cash unit.
cashManagement.exchangeType.toCassettes	boolean		The device supports moving items from the replenishment cash unit to another cash unit.
cashManagement.exchangeType.clearRecycler	boolean		The device supports the emptying of recycle cash units.
cashManagement.exchangeType.depositInto	boolean		The device supports moving items from the deposit entrance to the bill cash units.
cashManagement.itemInfoTypes	object		Specifies the types of information that can be retrieved through the CashManagement.GetItemInfo command.
cashManagement.itemInfoTypes.serialNumber	boolean		Serial Number of the item.
cashManagement.itemInfoTypes.signature	boolean		Signature of the item.
cashManagement.itemInfoTypes.imageFile	boolean		Image file of the item.
cashManagement.classificationList	boolean		Specifies whether the device has the capability to maintain a classification list of serial numbers as well as supporting the associated operations. This can either be TRUE if the device has the capability or FALSE if it does not.
cashManagement.physicalNoteList	boolean		Specifies whether the Service supports note number lists on physical cash units. This can either be TRUE if the Service has the capability or FALSE if it does not.
pinPad	object		Capability information for XFS4IoT services implementing the PinPad interface. This will be omitted if the PinPad interface is not supported.
pinPad.pinFormats <span style="border: 1px solid black; padding: 2px;">(Required)</span>	object		Supported PIN format
pinPad.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.

Name	Type	Default	Description
pinPad.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)
pinPad.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).
pinPad.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).
pinPad.pinFormats.eci2	boolean		PIN left justified, filled with padding characters, PIN only 4 digits.
pinPad.pinFormats.eci3	boolean		PIN is preceded by the length (digit), PIN length 4-6 digits, the padding character can range from 0x0 through 0xF
pinPad.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).
pinPad.pinFormats.diebold	boolean		PIN is padded with the padding character and may be not encrypted single encrypted or double encrypted.
pinPad.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.

Name	Type	Default	Description
pinPad.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.
pinPad.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.
pinPad.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.
pinPad.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.
pinPad.presentationAlgorithms <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	object		Supported presentation algorithms
pinPad.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
pinPad.display <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	object		Specifies the type of the display used in the PIN pad module
pinPad.display.none	boolean		No display unit
pinPad.display.ledThrough	boolean		Lights next to text guide user
pinPad.display.display	boolean		A real display is available (this doesn't apply for self-service).
pinPad.idConnect <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	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

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
pinPad.validationAlgorithms <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	object		Specifies the algorithms for PIN validation supported by the service
pinPad.validationAlgorithms.des	boolean		DES algorithm
pinPad.validationAlgorithms.visa	boolean		visa algorithm
pinPad.pinCanPersistAfterUse <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	boolean		Specifies whether the device can retain the PIN after a PIN processing command
pinPad.typeCombined <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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.
pinPad.setPinblockDataRequired <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	boolean		Specifies whether the command SetPinblockData must be called before the PIN is entered via GetPin and retrieved via GetPinblock.
pinPad.pinBlockAttributes <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	array		Array of attributes supported by the PinBlock command.
pinPad.pinBlockAttributes.keyUsage <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		Specifies the key usages supported by the PINBLOCK command.
pinPad.pinBlockAttributes.algorithm <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		Specifies the encryption algorithms supported by the PINBLOCK command as one of the following values
pinPad.pinBlockAttributes.modeOfUse <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		Specifies the encryption modes supported by the PINBLOCK command as one of the following values
pinPad.pinBlockAttributes.cryptoMethod <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by Algorithm.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
crypto	object		Capability information for XFS4IoT services implementing the Crypto interface. This will be omitted if the Crypto interface is not supported.
crypto.algorithms <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	object		Supported encryption modes.
crypto.algorithms.ecb	boolean		Electronic Code Book.
crypto.algorithms.cbc	boolean		Cipher Block Chaining.
crypto.algorithms.cfb	boolean		Cipher Feed Back.
crypto.algorithms.rsa	boolean		RSA Encryption.
crypto.algorithms.cma	boolean		ECMA Encryption.
crypto.algorithms.desMac	boolean		MAC calculation using CBC.
crypto.algorithms.triDesEcb	boolean		Triple DES with Electronic Code Book.
crypto.algorithms.triDesCbc	boolean		Triple DES with Cipher Block Chaining.
crypto.algorithms.triDesCfb	boolean		Triple DES with Cipher Feed Back.
crypto.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.
crypto.algorithms.maaMac	boolean		MAC calculation using the Message authenticator algorithm as defined in ISO 8731-2.
crypto.algorithms.triDesMac2805	boolean		Triple DES MAC calculation as defined in ISO 16609:2004 and and Australian Standard 2805.4.
crypto.algorithms.sm4	boolean		M4 block cipher algorithm as defined in Password industry standard of the People's Republic of China GM/T 0002-2012.

Name	Type	Default	Description
crypto.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 GMT 0002-2012. and and in PBOC3.0 JR/T 0025.17-2013.
crypto.emvHashAlgorithm <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	object		Specifies which hash algorithm is supported for the calculation of the HASH.
crypto.emvHashAlgorithm.sha1Digest	boolean		The SHA 1 digest algorithm is supported by the Digest command.
crypto.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.
crypto.cryptAttributes <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	array		Array of attributes supported by the CRYPT command.
crypto.cryptAttributes.keyUsage <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the key usage supported by the crypt command
crypto.cryptAttributes.algorithm <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the encryption algorithms supported by CRYPT command
crypto.cryptAttributes.modeOfUse <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the encryption mode supported by CRYPT command.
crypto.cryptAttributes.cryptoMethod <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the cryptographic method supported by the CRYPT command.
keyManagement	object		Capability information for XFS4IoT services implementing the KeyManagement interface. This will be omitted if the KeyManagement interface is not supported.
keyManagement.keyNum <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	integer		Number of the keys which can be stored in the encryption/decryption module
keyManagement.idKey <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	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.
keyManagement.idKey.initialization	boolean		ID key is returned by the Initialization command

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
keyManagement.idKey.import	boolean		ID key is required as input for the ImportKey and DeriveKey command
keyManagement.keyCheckModes <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	object		Specifies the key check modes that are supported to check the correctness of an imported key value
keyManagement.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.
keyManagement.keyCheckModes.zero	boolean		The key check value is created by encrypting a zero value with the key.
keyManagement.hsmVendor	string		Identifies the hsm Vendor. hsmVendor is an empty string or this field is not set when the hsm Vendor is unknown or the HSM is not supported.
keyManagement.rsaAuthenticationScheme <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	boolean		Specifies which type of Remote Key Loading/Authentication
keyManagement.rsaAuthenticationScheme.2partySig	boolean		Two-party Signature based authentication.
keyManagement.rsaAuthenticationScheme.3partyCert	boolean		Three-party Certificate based authentication.
keyManagement.rsaAuthenticationScheme.3partyCertTr34	boolean		Three-party Certificate based authentication described by X9 TR34-2012.
keyManagement.rsaSignatureAlgorithm <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	object		Specifies which type of rsa Signature Algorithm
keyManagement.rsaSignatureAlgorithm.pkcs1V15	boolean		pkcs1V15 Signatures supported.
keyManagement.rsaSignatureAlgorithm.pss	boolean		pss Signatures supported.
keyManagement.rsaCryptAlgorithm <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	object		Specifies which type of rsa Encipherment Algorith
keyManagement.rsaCryptAlgorithm.pkcs1V15	boolean		pkcs1V15 algorithm supported.
keyManagement.rsaCryptAlgorithm.oaep	boolean		oaep algorithm supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
keyManagement.rsaKeyCheckMode <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	object		Specifies which algorithm/method used to generate the public key check value/thumb print
keyManagement.rsaKeyCheckMode.sha1	boolean		sha1 is supported as defined in Ref. 3.
keyManagement.rsaKeyCheckMode.sha256	boolean		sha256 is supported as defined in ISO/IEC 10118-3:2004 and FIPS 180-2.
keyManagement.signatureScheme <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	object		Specifies which capabilities are supported by the Signature scheme
keyManagement.signatureScheme.genRsaKeyPair	boolean		Specifies if the Service Provider supports the rsa Signature Scheme GenerateRSAKeyPair and ExportRSAEPSSignedItem commands.
keyManagement.signatureScheme.randomNumber	boolean		Specifies if the Service Provider returns a random number from the StartKeyExchange GE command within the rsa Signature Scheme.
keyManagement.signatureScheme.exportEppId	boolean		Specifies if the Service Provider supports exporting the EPP Security Item within the rsa Signature Scheme.
keyManagement.signatureScheme.enhancedRkI	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.
keyManagement.emvImportSchemes <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	object		Identifies the supported emv Import Scheme(s)
keyManagement.emvImportSchemes.plainCA	boolean		A plain text CA public key is imported with no verification.
keyManagement.emvImportSchemes.chksumCA	boolean		A plain text CA public key is imported using the EMV 2000 verification algorithm.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
keyManagement.emvImportSchemes.epiCA	boolean		A CA public key is imported using the selfsign scheme defined in the Europay International, epi CA Module Technical - Interface specification.
keyManagement.emvImportSchemes.issuer	boolean		An Issuer public key is imported as defined in EMV 2000 Book II
keyManagement.emvImportSchemes.icc	boolean		An ICC public key is imported as defined in EMV 2000 Book II
keyManagement.emvImportSchemes.iccPin	boolean		An ICC PIN public key is imported as defined in EMV 2000 Book II
keyManagement.emvImportSchemes.pkcs15CA	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.
keyManagement.keyBlockImportFormats <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	object		Supported key block formats
keyManagement.keyBlockImportFormats.ansTr31KeyBlock	boolean		Supports ANS TR-31A Keyblock format key import.
keyManagement.keyBlockImportFormats.ansTr31KeyBlockB	boolean		Supports ANS TR-31B Keyblock format key import.
keyManagement.keyBlockImportFormats.ansTr31KeyBlockC	boolean		Supports ANS TR-31C Keyblock format key import.
keyManagement.keyImportThroughParts <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	boolean		Specifies whether the device is capable of importing keys in multiple parts. TRUE means the device supports the key import in multiple parts
keyManagement.desKeyLength <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	object		Specifies which length of DES keys are supported.
keyManagement.desKeyLength.single	boolean		8 byte DES keys are supported.
keyManagement.desKeyLength.double	boolean		16 byte DES keys are supported.
keyManagement.desKeyLength.triple	boolean		24 byte DES keys are supported.
keyManagement.certificateTypes <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	object		Specifies supported certificate types
keyManagement.certificateTypes.encKey	boolean		Supports the EPP public encryption certificate.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
keyManagement.certificateTypes.verificationKey	boolean		Supports the EPP public verification certificate.
keyManagement.certificateTypes.hostKey	boolean		Supports the Host public certificate.
keyManagement.loadCertOptions <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	array		Specifying the options supported by the LoadCertificate command
keyManagement.loadCertOptions.signer	string		Specifies the signers supported by the LoadCertificate command.
keyManagement.loadCertOptions.option	object		Specifies the load options supported by the LoadCertificate command
keyManagement.loadCertOptions.option.newHost	boolean		Load a new Host certificate, where one has not already been loaded.
keyManagement.loadCertOptions.option.replaceHost	boolean		Replace the epp to a new Host certificate, where the new Host certificate is signed by signer.
keyManagement.crklLoadOptions <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	object		Supported options to load the Key Transport Key using the Certificate Remote Key Loading protocol.
keyManagement.crklLoadOptions.noRandom	boolean		Import a Key Transport Key without generating and using a random number.
keyManagement.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.
keyManagement.crklLoadOptions.random	boolean		Import a Key Transport Key by generating and using a random number.
keyManagement.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.
keyManagement.restrictedKeyEncKeySupport <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	array		A array of object specifying the loading methods that support the RestrictedKeyEncKey usage flag and the allowable usage flag combinations.
keyManagement.restrictedKeyEncKeySupport.loadingMethod	string		Specifies the loading methods supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
keyManagement.restrictedKeyEncKeySupport.uses	object		Specifies one or more usage flags that can be used in combination with the RestrictedKeyEncKey
keyManagement.restrictedKeyEncKeySupport.uses.crypt	boolean		Key is used for encryption and decryption.
keyManagement.restrictedKeyEncKeySupport.uses.function	boolean		Key is used for Pin block creation.
keyManagement.restrictedKeyEncKeySupport.uses.macing	boolean		Key is using for macing.
keyManagement.restrictedKeyEncKeySupport.uses.pinlocal	boolean		Key is used only for local PIN check.
keyManagement.restrictedKeyEncKeySupport.uses.svenckey	boolean		Key is used as cbc start Value encryption key.
keyManagement.restrictedKeyEncKeySupport.uses.pinremote	boolean		Key is used only for PIN block creation.
keyManagement.symmetricKeyManagementMethods <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	object		Specifies the Symmentric Key Management modes
keyManagement.symmetricKeyManagementMethods.fixedKey	boolean		This method of key management uses fixed keys for transaction processing.
keyManagement.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.
keyManagement.symmetricKeyManagementMethods.tdesDukpt	boolean		This method uses TDES Derived Unique Key Per Transaction (see reference 45).
keyManagement.keyAttributes <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	array		Array of attributes supported by ImportKey command for the key to be loaded.
keyManagement.keyAttributes.keyUsage. keyManagement.keyAttributes.keyUsage.	integer string		
keyManagement.keyAttributes.propkeyUsage	integer 0		if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
keyManagement.keyAttributes.algorithm. keyManagement.keyAttributes.algorithm.	integer string		
keyManagement.keyAttributes.propAlgorithm	integer 0		if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
keyManagement.keyAttributes.modeOfUse. keyManagement.keyAttributes.modeOfUse.	integer string		
keyManagement.keyAttributes.propmodeOfUse	integer 0		if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
keyManagement.keyAttributes.cryptoMethod <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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
keyManagement.decryptAttributes <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	array		Array of attributes supported by the Import command for the key used to decrypt or unwrap the key being imported.
keyManagement.decryptAttributes.algorithm. keyManagement.decryptAttributes.algorithm.	integer string		
keyManagement.decryptAttributes.propAlgorithm	integer 0		if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
keyManagement.decryptAttributes.cryptoMethod <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by bAlgorithm.
keyManagement.verifyAttributes <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	array		Array of attributes supported by Import command for the key used for verification before importing the key.
keyManagement.verifyAttributes.keyUsage. keyManagement.verifyAttributes.keyUsage.	integer string		
keyManagement.verifyAttributes.propkeyUsage	integer 0		if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
keyManagement.verifyAttributes.algorithm. keyManagement.verifyAttributes.algorithm.	integer string		
keyManagement.verifyAttributes.propAlgorithm	integer 0		if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
keyManagement.verifyAttributes.modeOfUse. keyManagement.verifyAttributes.modeOfUse.	integer string		
keyManagement.verifyAttributes.propmodeOfUse	integer 0		if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
keyManagement.verifyAttributes.cryptoMethod <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
keyboard	object		Capability information for XFS4IoT services implementing the Keyboard interface. This will be omitted if the Keyboard interface is not supported.
keyboard.autoBeep <span style="border: 1px solid black; padding: 2px;">(Required)</span>	object		Specifies whether the device will emit a key beep tone on key presses of active keys or inactive keys, and if so, which mode it supports
keyboard.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.
keyboard.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.
keyboard.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.
keyboard.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.
keyboard.etsCaps	array		Specifies the capabilities of the ets device.
keyboard.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.
keyboard.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.
keyboard.etsCaps.xSize	integer		Specifies the width of the ets in Windows virtual screen coordinates.
keyboard.etsCaps.ySize	integer		Specifies the height of the ets in Windows virtual screen coordinates.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
keyboard.etsCaps.maximumTouchFrames	integer		Specifies the maximum number of Touch-Frames that the device can support in a touch keyboard definition.
keyboard.etsCaps.maximumTouchKeys	integer		Specifies the maximum number of Touch-Keys that the device can support within any a touchframe.
keyboard.etsCaps.floatFlags	object		Specifies if the device can float the touch keyboards. FloatNone if the PIN device cannot randomly shift the layout.
keyboard.etsCaps.floatFlags.x	boolean		Specifies that the PIN device will randomly shift the layout in a horizontal direction
keyboard.etsCaps.floatFlags.y	boolean		Specifies that the PIN device will randomly shift the layout in a vertical direction.
textTerminal	object		Capability information for XFS4IoT services implementing the TextTerminal interface. This will be omitted if the TextTerminal interface is not supported.
textTerminal.type <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	object		Specifies the type of the text terminal unit.
textTerminal.resolutions <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	array		Array specifies the resolutions supported by the physical display device. The resolution indicated in the first position is the default resolution and the device will be placed in this resolution when the Service Provider is initialized or reset through the reset command.
textTerminal.resolutions.sizeX	integer		Specifies the horizontal size of the display of the text terminal unit (the number of columns that can be displayed).
textTerminal.resolutions.sizeY	integer		Specifies the vertical size of the display of the text terminal unit (the number of rows that can be displayed).
textTerminal.keyLock <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	boolean		Specifies whether the text terminal unit has a key lock switch.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
textTerminal.displayLight	boolean		Specifies whether the text terminal unit has a display light that can be switched ON and OFF with the dispLight command.
textTerminal.cursor <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	boolean		Specifies whether the text terminal unit display supports a cursor.
textTerminal.forms <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	boolean		Specifies whether the text terminal unit service supports forms oriented input and output.
textTerminal.charSupport <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>			For charSupport, a Service Provider can support ONLY ascii forms or can support BOTH ascii and unicode forms. A Service Provider can not support UNICODE forms without also supporting ASCII forms."
textTerminal.charSupport.ascii	boolean		Ascii is supported for forms.
textTerminal.charSupport.unicode	boolean		Unicode is supported for forms.
textTerminal.leds	array		Specifies which LEDs are available. The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the LED is capable of handling. If the LED only supports one color then no value of type C is returned.
textTerminal.leds.off	boolean		The LED can be off. Type:(A)
textTerminal.leds.slowFlash	boolean		The LED can be blinking. Type:(B)
textTerminal.leds.mediumFlash	boolean		The LED can be blinking medium frequency. Type:(B)
textTerminal.leds.quickFlash	boolean		The LED can be blinking quickly. Type:(B)
textTerminal.leds.continuous	boolean		The LED can be turned on continuous(steady). Type:(B)
textTerminal.leds.red	boolean		The LED can be red. Type:(C)
textTerminal.leds.green	boolean		The LED can be green. Type:(C)
textTerminal.leds.yellow	boolean		The LED can be yellow. Type:(C)

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
textTerminal.leds.blue	boolean		The LED can be blue. Type:(C)
textTerminal.leds.cyan	boolean		The LED can be cyan. Type:(C)
textTerminal.leds.magenta	boolean		The LED can be magenta. Type:(C)
textTerminal.leds.white	boolean		The LED can be white. Type:(C)
printer	object		Capability information for XFS4IoT services implementing the Printer interface. This will be omitted if the Printer interface is not supported.
printer.type	object		Specifies the type(s) of the physical device driven by the logical service.
printer.type.receipt	boolean		The device is a receipt printer.
printer.type.passbook	boolean		The device is a passbook printer.
printer.type.journal	boolean		The device is a journal printer.
printer.type.document	boolean		The device is a document printer.
printer.type.scanner	boolean		The device is a scanner that may have printing capabilities.
printer.resolution	object		Specifies at which resolution(s) the physical device can print. Used by the application to select the level of print quality desired; does not imply any absolute level of resolution, only relative.
printer.resolution.low	boolean		The device can print low resolution.
printer.resolution.medium	boolean		The device can print medium resolution.
printer.resolution.high	boolean		The device can print high resolution.
printer.resolution.veryHigh	boolean		The device can print very high resolution.
printer.readForm	object		Specifies whether the device can read data from media, as a combination of the following flags.
printer.readForm.ocr	boolean		Device has OCR capability.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.readForm.micr	boolean		Device has MICR capability.
printer.readForm.msf	boolean		Device has MSF capability.
printer.readForm.barcode	boolean		Device has Barcode capability.
printer.readForm.pageMark	boolean		Device has Page Mark capability.
printer.readForm.readImage	boolean		Device has imaging capability.
printer.readForm.readEmptyLine	boolean		Device has capability to detect empty print lines for passbook printing.
printer.writeForm	object		Specifies whether the device can write data to the media, as a combination of the following flags.
printer.writeForm.text	boolean		Device has Text capability.
printer.writeForm.graphics	boolean		Device has Graphics capability.
printer.writeForm.ocr	boolean		Device has OCR capability.
printer.writeForm.micr	boolean		Device has MICR capability.
printer.writeForm.msf	boolean		Device has MSF capability.
printer.writeForm.barcode	boolean		Device has Barcode capability.
printer.writeForm.stamp	boolean		Device has stamping capability.
printer.extents	object		Specifies whether the device is able to measure the inserted media, as a combination of the following flags.
printer.extents.horizontal	boolean		Device has horizontal size detection capability.
printer.extents.vertical	boolean		Device has vertical size detection capability.
printer.control	object		Specifies the manner in which media can be controlled, as a combination of the following flags.
printer.control.eject	boolean		Device can eject media.
printer.control.perforate	boolean		Device can perforate media.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.control.cut	boolean		Device can cut media.
printer.control.skip	boolean		Device can skip to mark.
printer.control.flush	boolean		Device can be sent data that is buffered internally, and flushed to the printer on request.
printer.control.retract	boolean		Device can retract media under application control.
printer.control.stack	boolean		Device can stack media items before ejecting as a bundle.
printer.control.partialCut	boolean		Device can partially cut the media.
printer.control.alarm	boolean		Device can ring a bell, beep or otherwise sound an audible alarm.
printer.control.pageForward	boolean		Capability to turn one page forward.
printer.control.pageBackward	boolean		Capability to turn one page backward.
printer.control.turnMedia	boolean		Device can turn inserted media.
printer.control.stamp	boolean		Device can stamp on media.
printer.control.park	boolean		Device can park a document into the parking station.
printer.control.expel	boolean		Device can expel media out of the exit slot.
printer.control.ejectToTransport	boolean		Device can move media to a position on the transport just behind the exit slot.
printer.control.rotate180	boolean		Device can rotate media 180 degrees in the printing plane.
printer.control.clearBuffer	boolean		The Service Provider can clear buffered data.
printer.maxMediaOnStacker	integer		Specifies the maximum number of media items that the stacker can hold (zero if not available).

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
printer.acceptMedia	boolean		Specifies whether the device is able to accept media while no execute command is running that is waiting explicitly for media to be inserted.
printer.multiPage	boolean		Specifies whether the device is able to support multiple page print jobs.
printer.paperSources	object		Specifies the Paper sources available for this printer as a combination of the following flags
printer.paperSources.upper	boolean		Indicates an upper paper source is available; devices with only one paper supply must indicate upper as being available.
printer.paperSources.lower	boolean		Indicates a lower paper source is available.
printer.paperSources.external	boolean		Indicates an external paper source (such as envelope tray or single sheet feed) is available.
printer.paperSources.aux	boolean		An auxiliary paper source is available.
printer.paperSources.aux2	boolean		A second auxiliary paper source is available.
printer.paperSources.park	boolean		A parking station is available.
printer.mediaTaken	boolean		<a name="printer-capability-mediataken"></a>Specifies whether the device is able to detect when the media is taken from the exit slot. If false, the <a href="#">Printer.MediaTakenEvent</a> event is not fired.
printer.retractBins	integer		<a name="printer-capability-retractbins"></a>Specifies the number of retract bins (zero if not supported).
printer.maxRetract	array		An array of the length <a href="#">retractBins</a> with the maximum number of media items that each retract bin can hold (one count for each supported bin, starting from zero for bin number one to retractBins - 1 for bin number retractBins). This will be omitted if there are no retract bins.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.imageType	object		Specifies the image format supported by this device, as a combination of following flags.
printer.imageType.tif	boolean		The device can return scanned images in TIFF 6.0 format.
printer.imageType.wmf	boolean		The device can return scanned images in WMF (Windows Metafile) format.
printer.imageType.bmp	boolean		The device can return scanned images in Windows BMP format.
printer.imageType.jpg	boolean		The device can return scanned images in JPG format.
printer.frontImageColorFormat	object		Specifies the front image color formats supported by this device, as a combination of following flags.
printer.frontImageColorFormat.binary	boolean		The device can return scanned images in binary (image contains two colors, usually the colors black and white).
printer.frontImageColorFormat.grayscale	boolean		The device can return scanned images in gray scale (image contains multiple gray colors).
printer.frontImageColorFormat.full	boolean		The device can return scanned images in full color (image contains colors like red, green, blue etc.).
printer.backImageColorFormat	object		Specifies the back image color formats supported by this device, as a combination of following flags.
printer.backImageColorFormat.binary	boolean		The device can return scanned images in binary (image contains two colors, usually the colors black and white).
printer.backImageColorFormat.grayScale	boolean		The device can return scanned images in gray scale (image contains multiple gray colors).
printer.backImageColorFormat.full	boolean		The device can return scanned images in full color (image contains colors like red, green, blue etc.).

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.codelineFormat	object		Specifies the code line (MICR data) formats supported by this device, as a combination of following flags.
printer.codelineFormat.cmc7	boolean		The device can read CMC7 code lines.
printer.codelineFormat.e13b	boolean		The device can read E13B code lines.
printer.codelineFormat.ocr	boolean		The device can read code lines using Optical Character Recognition.
printer.imageSource	object		Specifies the source for the read image command supported by this device, as a combination of the following flags.
printer.imageSource.imageFront	boolean		The device can scan the front image of the document.
printer.imageSource.imageBack	boolean		The device can scan the back image of the document.
printer.imageSource.codeLine	boolean		The device can recognize the code line.
printer.dispensePaper	boolean		Specifies whether the device is able to dispense paper.
printer.osPrinter	string		Specifies the name of the default logical operating system printer that is associated with this Service Provider. Applications should use this printer name to generate native printer files to be printed through the <a href="#">Printer.PrintRawFile</a> command. This value will be omitted if the Service Provider does not support the <a href="#">Printer.PrintRawFile</a> command.
printer.mediaPresented	boolean		Specifies whether the device is able to detect when the media is presented to the user for removal. If true, the <a href="#">Printer.MediaPresentedEvent</a> event is fired. If false, the <a href="#">Printer.MediaPresentedEvent</a> event is not fired.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.autoRetractPeriod	integer		<a name="printer-capability-autoretractperiod"></a>Specifies the number of seconds before the device will automatically retract the presented media. If the command that generated the media is still active when the media is automatically retracted, the command will complete with an error. If the device does not retract media automatically this value will be zero.
printer.retractToTransport	boolean		Specifies whether the device is able to retract the previously ejected media to the transport.
printer.coercivityType	object		Specifies the form write modes supported by this device, as a combination of the following flags.
printer.coercivityType.low	boolean		This device can write the magnetic stripe by low coercivity mode.
printer.coercivityType.high	boolean		This device can write the magnetic stripe by high coercivity mode.
printer.coercivityType.auto	boolean		The Service Provider or the device is capable of automatically determining whether low or high coercivity magnetic stripe should be written.
printer.controlPassbook	object		<a name="printer-capability-controlpassbook"></a>Specifies how the passbook can be controlled with the <a href="#">Printer.ControlPassbook</a> command, as a combination of the following flags.
printer.controlPassbook.turnForward	boolean		The device can turn forward multiple pages of the passbook.
printer.controlPassbook.turnBackward	boolean		The device can turn backward multiple pages of the passbook.
printer.controlPassbook.closeForward	boolean		The device can close the passbook forward.
printer.controlPassbook.closeBackward	boolean		The device can close the passbook backward.

Name	Type	Default	Description
printer.printSides	string		<p>Specifies on which sides of the media this device can print as one of the following values.</p> <p><b>notSupp</b>&lt;br&gt; The device is not capable of printing on any sides of the media.</p> <p><b>single</b>&lt;br&gt; The device is capable of printing on one side of the media.</p> <p><b>dual</b>&lt;br&gt; The device is capable of printing on two sides of the media.</p>

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "interfaces": [
      {
        "name": "Common",
        "commands": [
          "string"
        ],
        "events": [
          "string"
        ],
        "maximumRequests": 0,
        "authenticationRequired": [
          "string"
        ]
      }
    ],
    "common": {
      "serviceVersion": "string",
      "deviceInformation": [
        {
          "modelName": "string",
          "serialNumber": "string",
          "revisionNumber": "string",
          "modelDescription": "string",
          "firmware": [
            {
              "firmwareName": "string",
              "firmwareVersion": "string",
              "hardwareRevision": "string"
            }
          ],
          "software": [
            {
              "firmwareName": "string",
              "firmwareVersion": "string",
              "hardwareRevision": "string"
            }
          ]
        }
      ]
    }
  }
}
```

```
        ],
        "vendorModeInformation": {
            "allowOpenSessions": true,
            "allowedExecuteCommands": [
                "string"
            ]
        },
        "extra": [
            "string"
        ],
        "guideLights": [
            {
                "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"
        ],
        "endToEndSecurity": true,
        "hardwareSecurityElement": true,
        "responseSecurityEnabled": true
    },
    "cardReader": {
        "type": "motor",
        "readTracks": {
            "track1": true,
            "track2": true,
            "track3": true,
            "watermark": true,
            "frontTrack1": true,
            "frontImage": true,
            "backImage": true,
            "track1JIS": true,
            "track3JIS": true,
            "ddi": true
        },
        "writeTracks": {
            "track1": true,
            "track2": true,
            "track3": true,
            "frontTrack1": true,
            "track1JIS": true,
            "track3JIS": true
        }
    }
}
```

```
        },
        "chipProtocols": {
            "chipT0": true,
            "chipT1": true,
            "chipProtocolNotRequired": true,
            "chipTypeAPart3": true,
            "chipTypeAPart4": true,
            "chipTypeB": true,
            "chipTypeNFC": true
        },
        "maxCardCount": 0,
        "securityType": "notSupported",
        "powerOnOption": "noAction",
        "powerOffOption": "noAction"
    },
    "cashAcceptor": {
        "type": "tellerBill",
        "maxCashInItems": 0,
        "shutter": true,
        "shutterControl": true,
        "intermediateStacker": 0,
        "itemsTakenSensor": true,
        "itemsInsertedSensor": true,
        "positions": {
            "inLeft": true,
            "inRight": true,
            "inCenter": true,
            "inTop": true,
            "inBottom": true,
            "inFront": true,
            "inRear": true,
            "outLeft": true,
            "outRight": true,
            "outCenter": true,
            "outTop": true,
            "outBottom": true,
            "outFront": true,
            "outRear": true
        },
        "retractAreas": {
            "retract": true,
            "transport": true,
            "stacker": true,
            "reject": true,
            "billCassette": true,
            "cashIn": true
        },
        "retractTransportActions": {
            "present": true,
            "retract": true,
            "reject": true,
            "billCassette": true,
            "cashIn": true
        },
        "retractStackerActions": {
            "present": true,
            "retract": true,
            "reject": true,
            "billCassette": true,
            "cashIn": true
        },
        "compareSignatures": true,
        "replenish": true,
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
"cashInLimit": {
    "byTotalItems": true,
    "byAmount": true,
    "multiple": true,
    "refuseOther": true
},
"countActions": {
    "individual": true,
    "all": true
},
"deviceLockControl": true,
"mixedMode": true,
"mixedDepositAndRollback": true,
"deplete": true
},
"cashDispenser": {
    "type": "tellerBill",
    "maxDispenseItems": 0,
    "shutter": true,
    "shutterControl": true,
    "retractAreas": {
        "retract": true,
        "transport": true,
        "stacker": true,
        "reject": true,
        "itemCassette": true
    },
    "retractTransportActions": {
        "present": true,
        "retract": true,
        "reject": true,
        "itemCassette": true
    },
    "retractStackerActions": {
        "present": true,
        "retract": true,
        "reject": true,
        "itemCassette": true
    },
    "intermediateStacker": true,
    "itemsTakenSensor": true,
    "positions": {
        "left": true,
        "right": true,
        "center": true,
        "top": true,
        "bottom": true,
        "front": true,
        "rear": true
    },
    "moveItems": {
        "fromCashUnit": true,
        "toCashUnit": true,
        "toTransport": true,
        "toStacker": true
    },
    "prepareDispense": true
},
"cashManagement": {
    "safeDoor": true,
    "cashBox": true,
    "exchangeType": {
        "byHand": true,
        "byMachine": true
    }
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
        "toCassettes": true,
        "clearRecycler": true,
        "depositInto": true
    },
    "itemInfoTypes": {
        "serialNumber": true,
        "signature": true,
        "imageFile": true
    },
    "classificationList": true,
    "physicalNoteList": true
},
"pinPad": {
    "pinFormats": {
        "3624": true,
        "ansi": true,
        "iso0": true,
        "isol": true,
        "eci2": true,
        "eci3": true,
        "visa": true,
        "diebold": true,
        "dieboldCo": true,
        "visa3": true,
        "emv": true,
        "iso3": true,
        "ap": true
    },
    "presentationAlgorithms": {
        "presentClear": true
    },
    "display": {
        "none": true,
        "ledThrough": true,
        "display": true
    },
    "idConnect": true,
    "validationAlgorithms": {
        "des": true,
        "visa": true
    },
    "pinCanPersistAfterUse": true,
    "typeCombined": true,
    "setPinblockDataRequired": true,
    "pinBlockAttributes": []
},
"crypto": {
    "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
    }
},
```

```
"emvHashAlgorithm": {
    "shalDigest": true,
    "sha256Digest": true
},
"cryptAttributes": [],
},
"keyManagement": {
    "keyNum": 0,
    "idKey": {
        "initialization": true,
        "import": true
    },
    "keyCheckModes": {
        "self": true,
        "zero": true
    },
    "hsmVendor": "string",
    "rsaAuthenticationScheme": true,
    "rsaSignatureAlgorithm": {
        "pkcs1V15": true,
        "pss": true
    },
    "rsaCryptAlgorithm": {
        "pkcs1V15": true,
        "oaeP": true
    },
    "rsaKeyCheckMode": {
        "shal": true,
        "sha256": true
    },
    "signatureScheme": {
        "genRsaKeyPair": true,
        "randomNumber": true,
        "exportEppId": true,
        "enhancedRki": true
    },
    "emvImportSchemes": {
        "plainCA": true,
        "chksumCA": true,
        "epiCA": true,
        "issuer": true,
        "icc": true,
        "iccPin": true,
        "pkcsv15CA": true
    },
    "keyBlockImportFormats": {
        "ansTr31KeyBlock": true,
        "ansTr31KeyBlockB": true,
        "ansTr31KeyBlockC": true
    },
    "keyImportThroughParts": true,
    "desKeyLength": {
        "single": true,
        "double": true,
        "triple": true
    },
    "certificateTypes": {
        "encKey": true,
        "verificationKey": true,
        "hostKey": true
    },
    "loadCertOptions": [
        {
            "key": "string"
        }
    ]
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
        "signer": "certHost",
        "option": {
            "newHost": true,
            "replaceHost": true
        }
    },
    "crkLoadOptions": {
        "noRandom": true,
        "noRandomCrl": true,
        "random": true,
        "randomCrl": true
    },
    "restrictedKeyEncKeySupport": [
        {
            "loadingMethod": "rsaAuth2partySig",
            "uses": [
                "crypt": true,
                "function": true,
                "macing": true,
                "pinlocal": true,
                "svenckey": true,
                "pinremote": true
            ]
        }
    ],
    "symmetricKeyManagementMethods": {
        "fixedKey": true,
        "masterKey": true,
        "tdesDukpt": true
    },
    "keyAttributes": [],
    "decryptAttributes": [],
    "verifyAttributes": []
},
"keyboard": {
    "autoBeep": {
        "activeAvailable": true,
        "activeSelectable": true,
        "inactiveAvailable": true,
        "inactiveSelectable": true
    },
    "etsCaps": [
        {
            "xPos": 0,
            "yPos": 0,
            "xSize": 0,
            "ySize": 0,
            "maximumTouchFrames": 0,
            "maximumTouchKeys": 0,
            "floatFlags": {
                "x": true,
                "y": true
            }
        }
    ]
},
"textTerminal": {
    "type": "fixed",
    "resolutions": [
        {
            "sizeX": 0,
            "sizeY": 0
        }
    ]
}
```

```
        }
    ],
    "keyLock": true,
    "displayLight": true,
    "cursor": true,
    "forms": true,
    "charSupport": {
        "ascii": true,
        "unicode": true
    },
    "leds": [
        {
            "off": true,
            "slowFlash": true,
            "mediumFlash": true,
            "quickFlash": true,
            "continuous": true,
            "red": true,
            "green": true,
            "yellow": true,
            "blue": true,
            "cyan": true,
            "magenta": true,
            "white": true
        }
    ]
},
"printer": {
    "type": {
        "receipt": true,
        "passbook": true,
        "journal": true,
        "document": true,
        "scanner": true
    },
    "resolution": {
        "low": true,
        "medium": true,
        "high": true,
        "veryHigh": true
    },
    "readForm": {
        "ocr": true,
        "micr": true,
        "msf": true,
        "barcode": true,
        "pageMark": true,
        "readImage": true,
        "readEmptyLine": true
    },
    "writeForm": {
        "text": true,
        "graphics": true,
        "ocr": true,
        "micr": true,
        "msf": true,
        "barcode": true,
        "stamp": true
    },
    "extents": {
        "horizontal": true,
        "vertical": true
    }
},
```

```
"control": {
    "eject": true,
    "perforate": true,
    "cut": true,
    "skip": true,
    "flush": true,
    "retract": true,
    "stack": true,
    "partialCut": true,
    "alarm": true,
    "pageForward": true,
    "pageBackward": true,
    "turnMedia": true,
    "stamp": true,
    "park": true,
    "expel": true,
    "ejectToTransport": true,
    "rotate180": true,
    "clearBuffer": true
},
"maxMediaOnStacker": 0,
"acceptMedia": true,
"multiPage": true,
"paperSources": {
    "upper": true,
    "lower": true,
    "external": true,
    "aux": true,
    "aux2": true,
    "park": true
},
"mediaTaken": true,
"retractBins": 0,
"maxRetract": [
    0
],
"imageType": {
    "tif": true,
    "wmf": true,
    "bmp": true,
    "jpg": true
},
"frontImageColorFormat": {
    "binary": true,
    "grayscale": true,
    "full": true
},
"backImageColorFormat": {
    "binary": true,
    "grayScale": true,
    "full": true
},
"codelineFormat": {
    "cmc7": true,
    "e13b": true,
    "ocr": true
},
"imageSource": {
    "imageFront": true,
    "imageBack": true,
    "codeLine": true
},
"dispensePaper": true,
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
        "osPrinter": "string",
        "mediaPresented": true,
        "autoRetractPeriod": 0,
        "retractToTransport": true,
        "coercivityType": {
            "low": true,
            "high": true,
            "auto": true
        },
        "controlPassbook": {
            "turnForward": true,
            "turnBackward": true,
            "closeForward": true,
            "closeBackward": true
        },
        "printSides": "notSupp"
    }
}
```

## Event Messages

## Common.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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
command.color	string		Indicates which colors are supported by the guidelight.
command.direction	string		Indicates which directions are supported by the guidelight. and it's 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

## Event Messages

---

## Common.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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

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

#### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [Common.PowerSaveChangeEvent](#)
- 

## Common.SynchronizeCommand

---

### Description

---

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

The list of execute commands which this command supports for synchronization is retrieved in the *lpdwSynchronizableCommands* parameter of the WFS\_INF\_CDM\_CAPABILITIES.

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

### Command Message

#### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

## Common.SetTransactionState

#### Description

This command allows the application to specify the transaction state, which the Service Provider can then utilize in order to optimize performance. After receiving this command, this Service Provider can perform the necessary processing to start or end the customer transaction. This command should be called for every Service Provider that could be used in a customer transaction. The transaction state applies to every session.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
name <b>(Required)</b>	string	The original message name, for example "CardReader.Status"	

#### Message Payload

Name	Type	Default	Description
			Specifies the transaction state. Following values are possible:
state	string	"active": A customer transaction is in progress. "inactive": No customer transaction is in progress.	
transactionID	string		Specifies a string which identifies the transaction ID. The value returned in this parameter is an application defined customer transaction identifier, which was previously set in the Common.SetTransactionState command
extra	array		A list of vendor-specific, or any other extended, transaction information. The information is set as a series of "key=value" strings. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "state": "active",
    "transactionID": "string",
    "extra": [
      "string"
    ]
  }
}
```

#### Completion Message

##### Message Header

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

## Common.GetTransactionState

#### Description

This command can be used to get the transaction state.

#### Command Message

##### Message Header

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

##### Message Payload

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload**

Name	Type	Default	Description
state	string		Specifies the transaction state. Following values are possible: "active": A customer transaction is in progress. "inactive": No customer transaction is in progress.
transactionID	string		Specifies a string which identifies the transaction ID. The value returned in this parameter is an application defined customer transaction identifier, which was previously set in the Common.SetTransactionState command

Name	Type	Default	Description
extra	array		A list of vendor-specific, or any other extended, transaction information. The information is set as a series of "key=value" strings. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "state": "active",
    "transactionID": "string",
    "extra": [
      "string"
    ]
  }
}
```

## Event Messages

---

### Common.GetCommandRandomNumber

---

#### Description

Get a random number to be included in an Authorisation Token for a command that will be used to ensure end to end security.

The hardware will overwrite any existing stored Command Random Number with this new value. The value will be stored for future authentication. Any Authorisation Token received will be compared with this stored random number and if the Token doesn't contain the same random number it will be considered invalid and rejected, causing the command that contains that Authentication Token to fail.

The random number must match the algorithm used. For example, HMAC means the random number must be 128 bit/16 bytes.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

##### Message Payload

Name	Type	Default	Description
CommandRandomNumber	string		A random number that should be included in the authorisation token in a command used to provide end to end protection. The random number will be given as HEX (upper case.)

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "CommandRandomNumber": "646169ECDD0E440C2CECC8DDD7C27C22",
    "completionCode": "success",
    "errorDescription": "string"
  }
}
```

## Event Messages

---

# Unsolicited Events

## Common.PowerSaveChangeEvent

---

### Description

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

#### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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
  }
}
```

## Dispenser.DevicePositionEvent

---

### Description

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

### Message Header

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

### Message Payload

Name	Type	Default	Description
Position	string		Position of the device

### Example Message (generated)

XFS4I  
All rights reserved

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

---

## Events

# XFS4IOT Sample Messaging for Crypto Interface Draft 0.0.4

## Documentation

### **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.

<b>Item</b>	<b>Description</b>
Name	This key represents the IPEK, the derived future keys stored during import of the IPEK "dukptipek" and the variant per transaction keys (PIN and optionally data and MAC).

## Commands

### **Crypto.GenerateRandom**

---

#### **Description**

This command is used to generate a random number.

#### **Command Message**

##### **Message Header**

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

Name	Type	Default	Description
randomNumber <b>(Required)</b>	string		The generated random number.

**Example Message (generated)**

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

**Event Messages**

## Crypto.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 cannot be used for random number generation. For random number generation, the GenerateRandom 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload 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 the stored key. This field is not required, if mode equals random.
startValueKey	string		Specifies the name of the stored key used to decrypt the startValue to obtain the initialization vector. If this field is not set, startValue is used as the initialization vector. This field is not required, if mode equals random.
startValue <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		DES and Triple DES initialization vector for cbc / cfb encryption and macing. This value is not required, if mode equals random.
padding	integer		Specifies the padding character. The padding character is a full byte, e.g. 0xFF. This value is not required, 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 field is not required, 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 can be omitted.
cryptAttributes <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	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 <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the key usage supported by the crypt command
cryptAttributes.algorithm <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the encryption algorithms supported by CRYPT command
cryptAttributes.modeOfUse <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the encryption mode supported by CRYPT command.

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

Name	Type	Default	Description
cryptData <b>(Required)</b>	string		The encrypted or decrypted data, mac value or signature. This parameter will be NULL if the cryptAttributes.modeOfUse is 'V'.

#### Example Message (generated)

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

## Event Messages

- [Pinpad.DUKPTKSNEvent](#)
- 

## Crypto.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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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
hashAlgorithm <b>(Required)</b>	string		Specifies which hash algorithm should be used to calculate the hash. See the Capabilities section for valid algorithms.
digestInput <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
digestOutput <b>(Required)</b>	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": {  
        "completionCode": "success",  
        "errorDescription": "string",  
        "digestOutput": "string"  
    }  
}
```

## Event Messages

---

# Unsolicited Events

## Crypto.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 <span>(Required)</span>	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 <span>(Required)</span>	string		The message type, either command, response, event or completion.
name <span>(Required)</span>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
keyName <span>(Required)</span>	string		Specifies the name of the key that caused the error.
errorCode <span>(Required)</span>	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"  
    }  
}
```

## Events

### Pinpad.DUKPTKSNEvent

#### Description

This event sends the DUKPT KSN of the key used in the command. The receiving TRSM uses this to derive the key from the BDK.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
key <small>(Required)</small>	string		Specifies the name of the DUKPT Key derivation key.
ksn <small>(Required)</small>	string		structure that contains the KSN formatted in base64.

#### Example Message (generated)

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

# XFS4IOT Sample Messaging for KeyManagement Draft 0.0.4

This section describes the general interface for the following functions:

- Loading of encryption keys
- EMV 4.0 PIN blocks, EMV 4.0 public key loading, static and dynamic data verification

Important Notes:

- This revision of this specification does not define all key management procedures; some key management is still vendor-specific.
- Key space management is customer-specific, and is therefore handled by vendor-specific mechanisms.

Key values are passed to the API as binary hexadecimal values, for example: 0123456789ABCDEF = 0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF When hex values are passed to the API within strings, the hex digits 0xA to 0xF can be represented by characters in the ranges 'a' to 'f' or 'A' to 'F'. The following commands and events were initially added to support the German ZKA standard, but may also be used for other national standards:

Certain levels of the PCI EPP security standards specify that if a key encryption key is deleted or replaced, then all keys in the hierarchy under that key encryption key are also removed. Key encryption keys have the keyEncKey type of access. Applications can check impact of key deletion using Keydetail.

## Documentation

### Appendix-A

---

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

#### Definitions and Abbreviations

ATM	Automated Teller Machine, used here for any type of self-service terminal, regardless whether it actually dispenses cash
CA	Certificate Authority
Certificate	A data structure that contains a public key and a name that allows certification of a public key belonging to a specific individual. This is certified using digital signatures.
HOST	The remote system that an ATM communicates with.
KTK	Key Transport Key
PKI	Public Key Infrastructure
Private Key	That key of an entity's key pair that should only be used by that entity.

### Definitions

#### and

#### Abbreviations

Public Key	That key of an entity's key pair that can be made public.
Symmetric Key	A key used with symmetric cryptography
verification Key	A key that is used to verify the validity of a certificate
signatureIssuer	An entity that signs the ATM's public key at production time, may be the ATM manufacturer

#### Notation of Cryptographic

#### Items and Functions

SKE	The private key belonging to entity E
PKE	The public key 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
KTK	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
Des3K	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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 encryption module's Unique Identifiers. The Signature Issuer installs a Signature Sign[SK SI]( UI ATM ) for the Unique Id (UI ATM ) on the ATM encryption module. The Signature Issuer uses SKSI to do this.
5. The Signature Issuer installs its Public Key (PK SI) on the ATM encryption module. It also derives and installs the Signature Sign[SK SI](PK ATM) of the ATM encryption module's Public Key (PK ATM) on the ATM encryption module. The Signature Issuer uses SKSI to do this.
6. The ATM encryption module additionally contains its own Public (PK ATM) and Private Key (SK ATM).

**Step 1**

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

The ATM KeyManagement 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 encryption module's 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 encryption module unique identifier. The ATM KeyManagement 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 Encryption Module Unique Identifier. It can then check this against the list it received from the Signature Issuer.

The command used to export the encryption module Unique Identifier is ExportRsaIssuerSignedItem.

**Step 3 (Enhanced Remote Key Loading only)**

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

The Host sends its Root Public Key (PKROOT) and associated Signature. The ATM encryption module 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 KeyManagement:

The Host sends its Public Key (PK HOST) and associated Signature. The ATM encryption module 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 <br>**

The ATM KeyManagement 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 encryption module will then validate the signature using PKHOST and then obtain the master key by decrypting using SKATM.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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 KeyManagement 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 encryption module 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 decrypt 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.

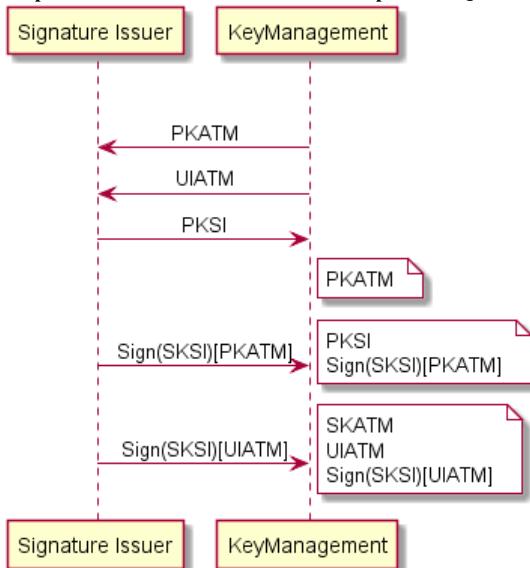
In addition the following optional keys can be loaded during manufacture.

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

---

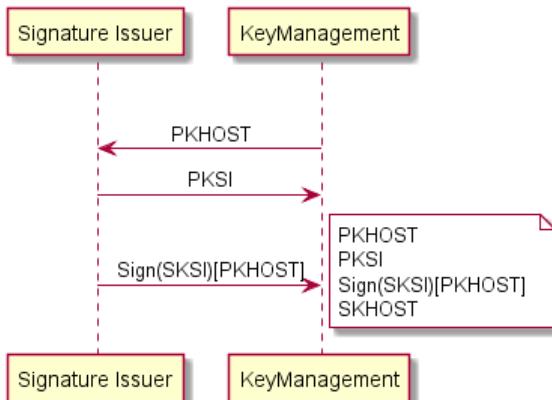
This would typically occur in a secure manufacturing environment.



## Initialization Phase – Signature Issuer and Host

---

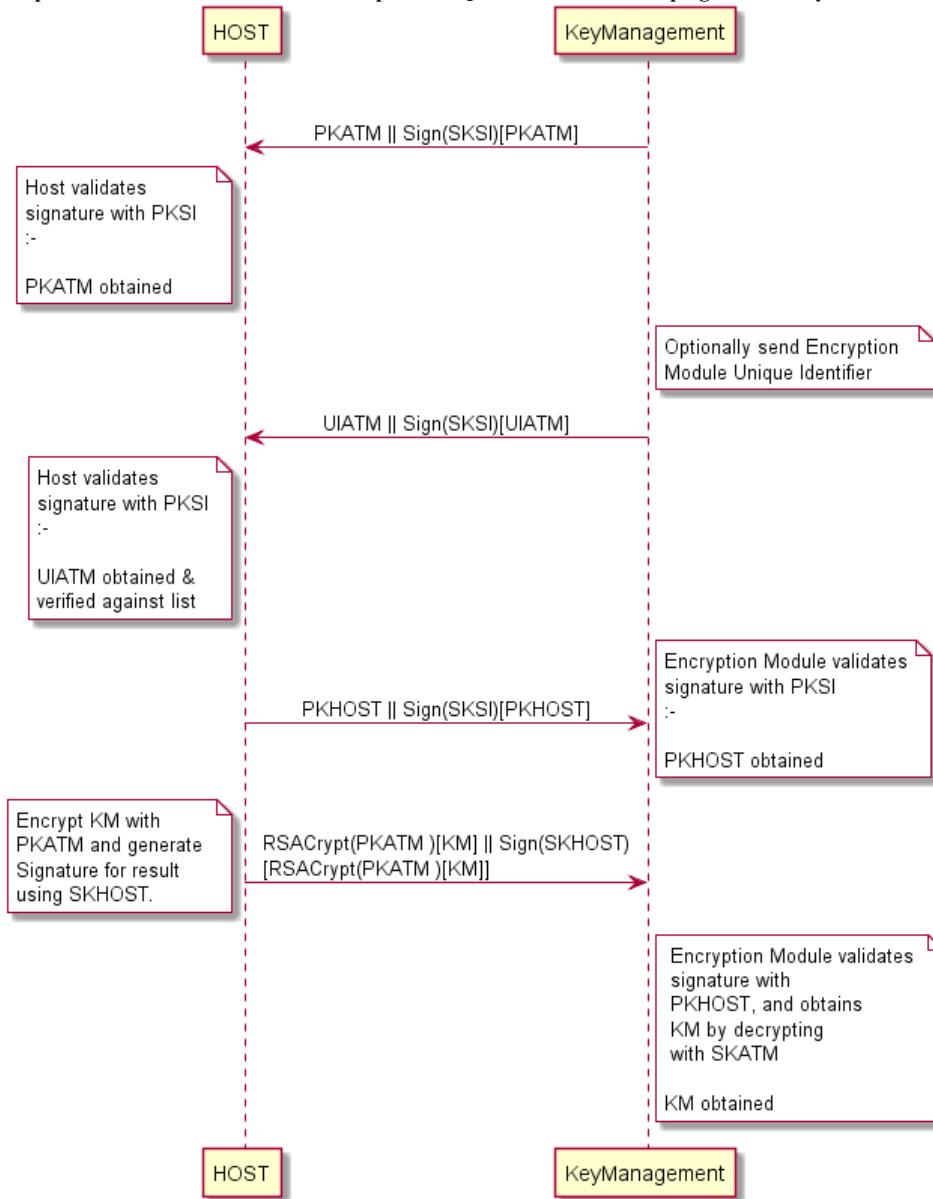
This would typically occur in a secure offline environment.



## Key Exchange – Host and ATM PIN

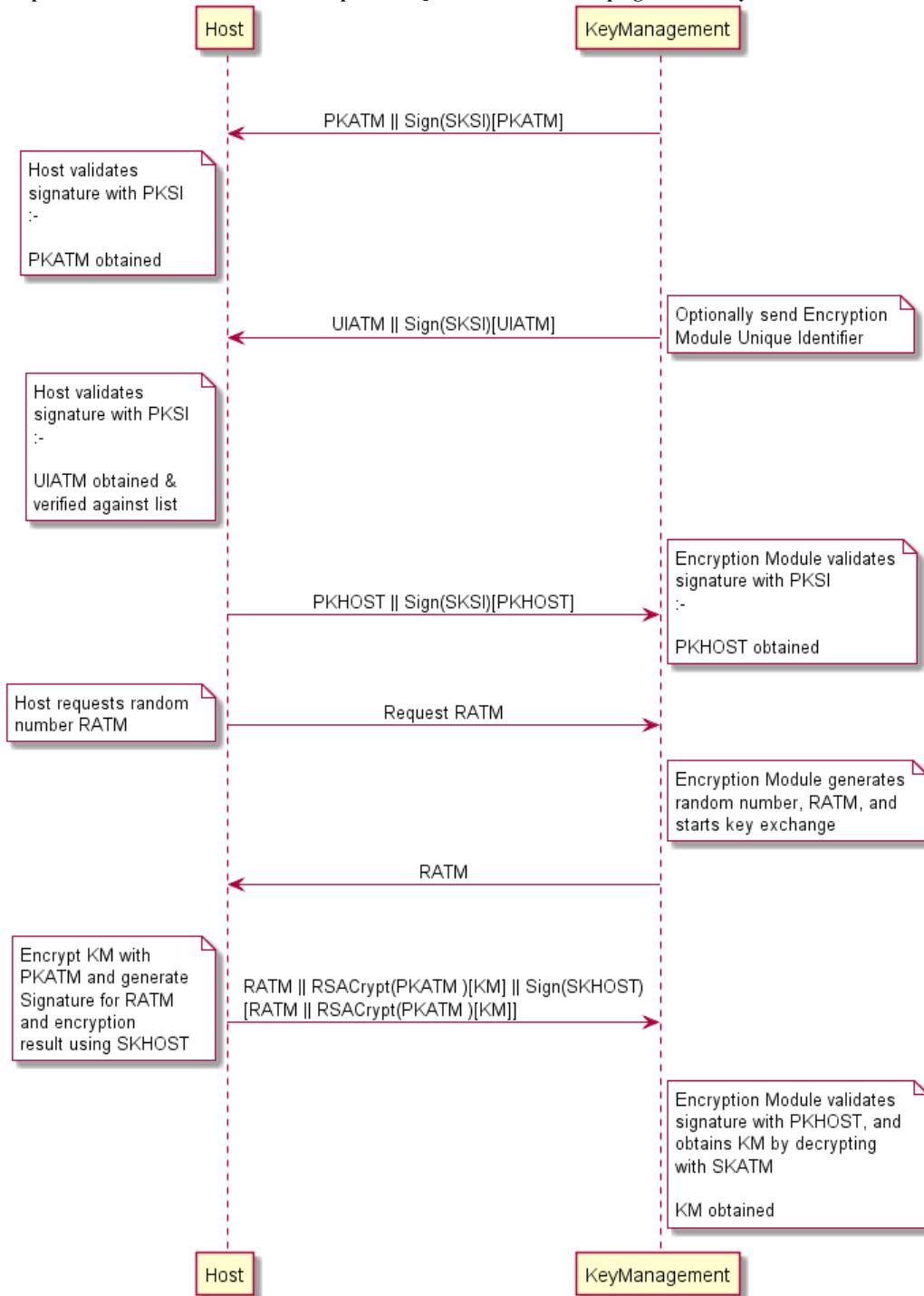
---

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.



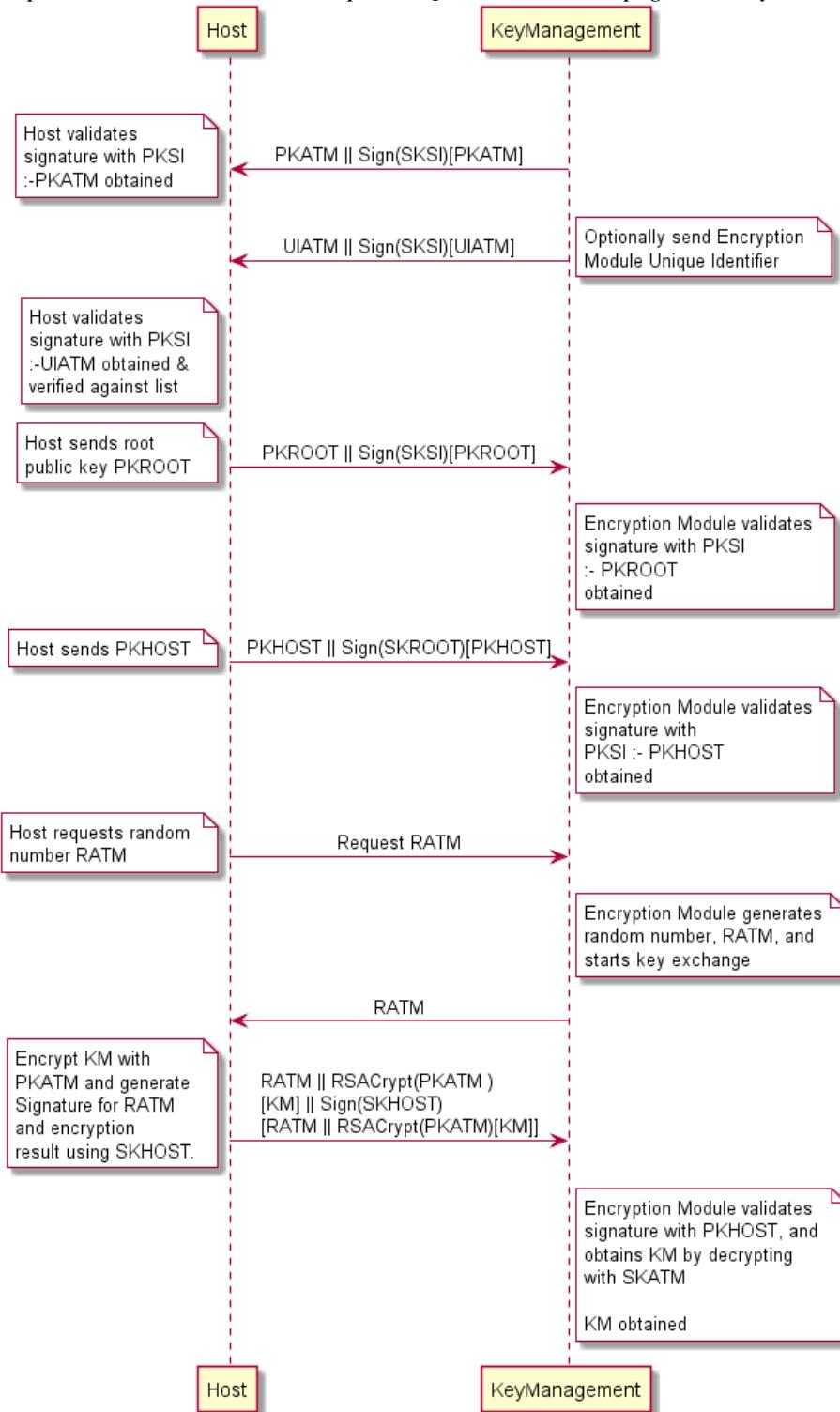
## Key Exchange (with random number) – Host and ATM PIN

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 KeyManagement interface to accomplish Remote Key Loading using Certificates. Beginning with Section 8.2.5, there are sequence

## Certificate Exchange and Authentication

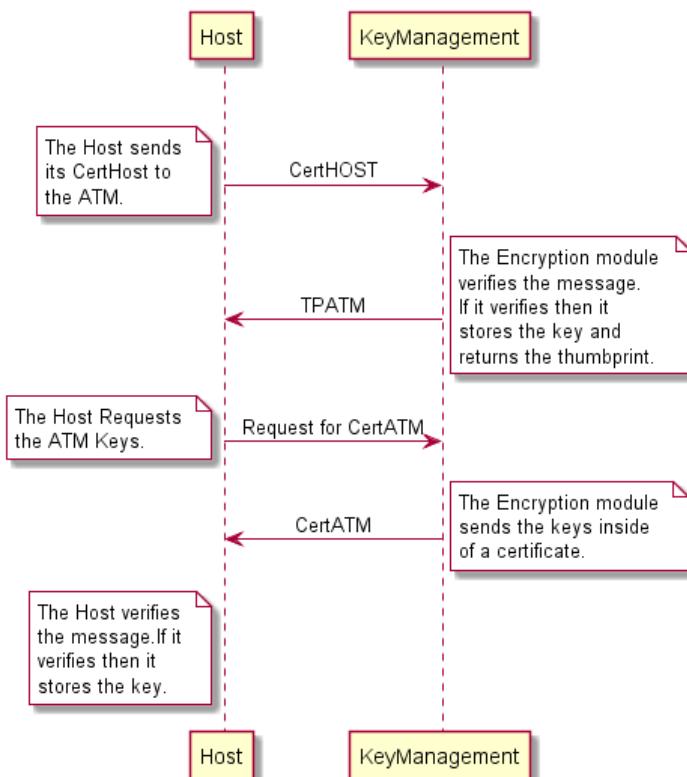
---

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

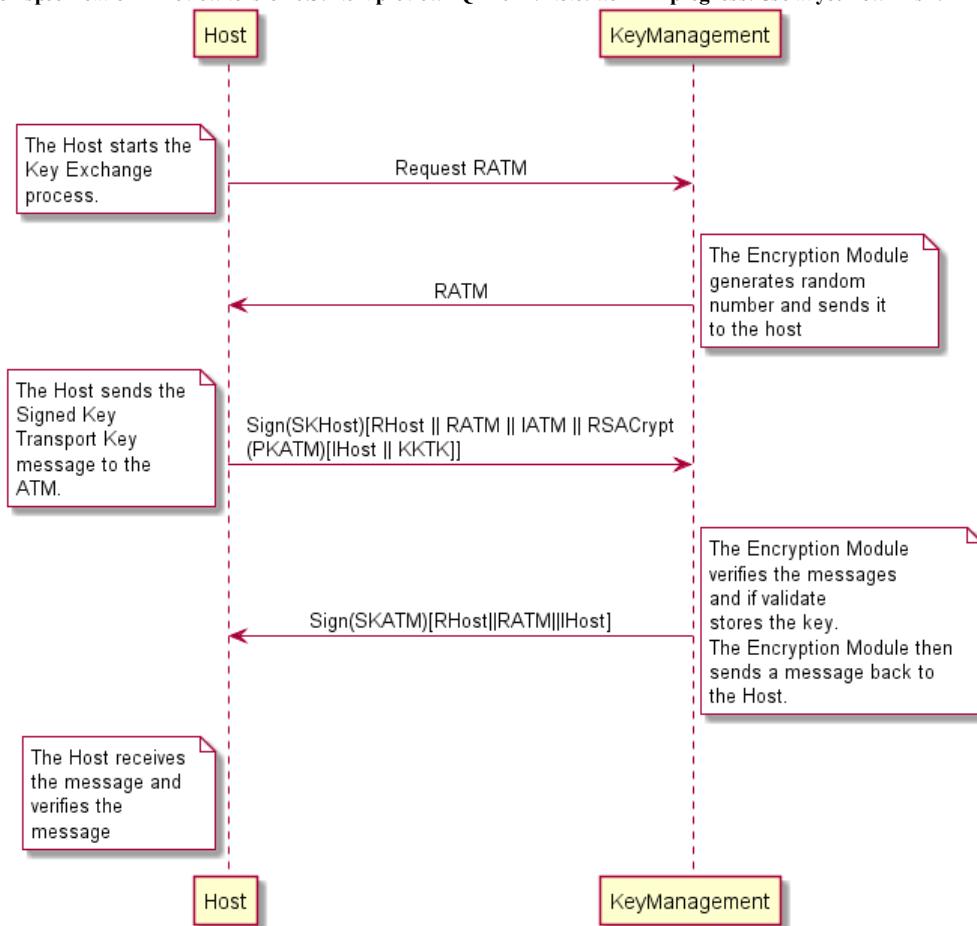
---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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:

1. First, the StartKeyExchange is called. This returns RATM from the encryptor to be used in the authenticating the ImportRSAEnchiperdPKCS7Key message.
2. 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, KTK, 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.

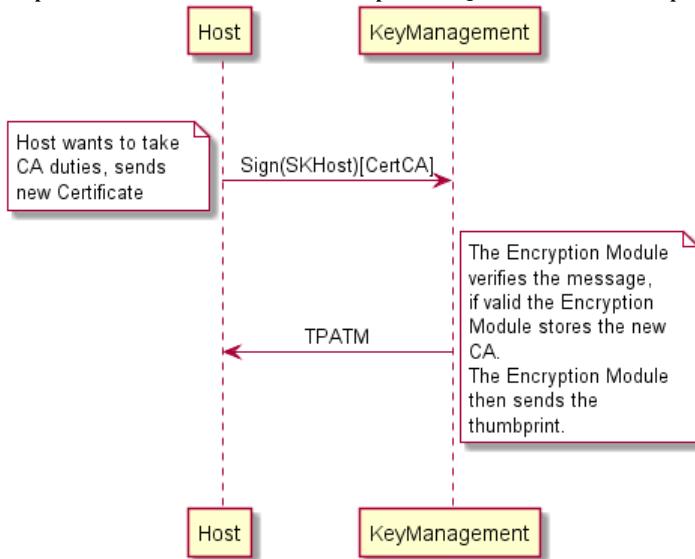


## Replace Certificate

---

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 oversign 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

---

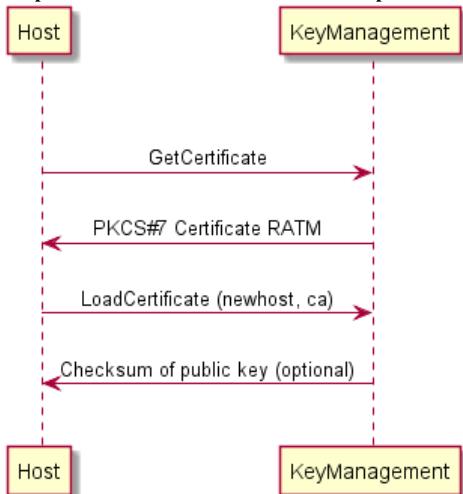
Primary and Secondary Certificates for both the Public Verification Key and Public Encipherment Key are pre-loaded into the encryptor. Primary Certificates will be used until told otherwise by the host via the LoadCertificate or ReplaceCertificate commands. This change in state will be specified in the 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.



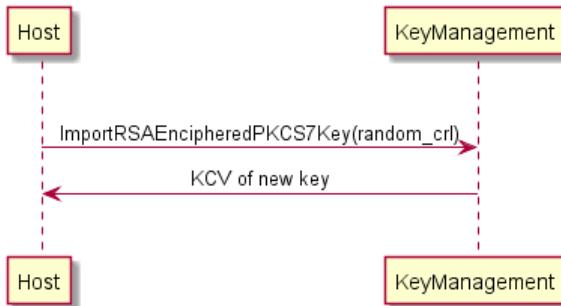
## TR34 Key Transport

---

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.

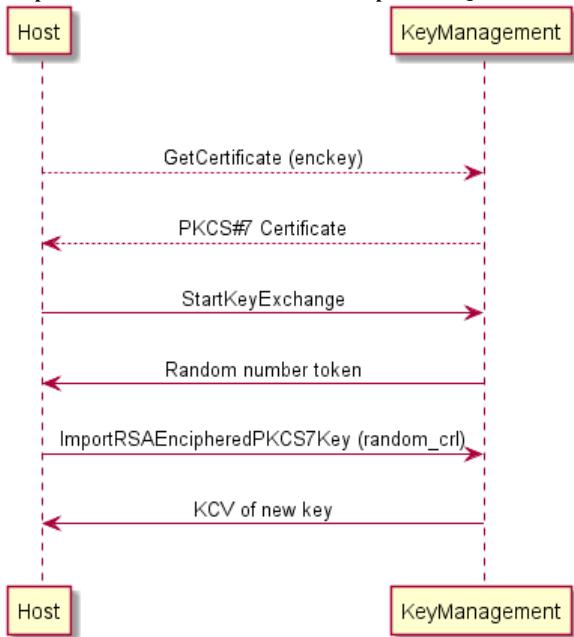
### One Pass

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 KeyManagement is bound to the host.



### Two Pass

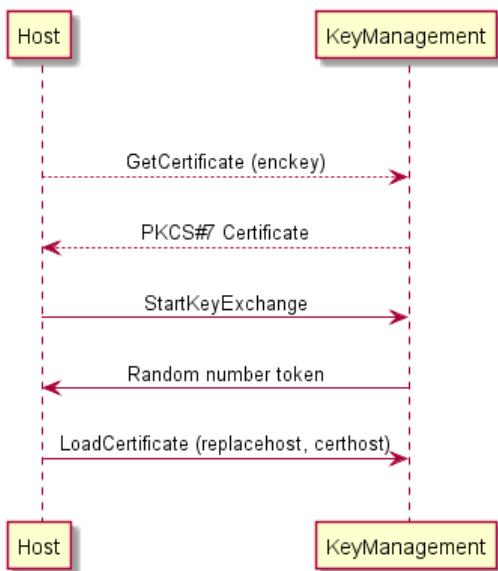
This section defines the command to use when transferring a TR34 KEY token (2-pass) as defined in reference. Pre-condition: A successful BIND command has completed such that the KeyManagement is bound to the host.



## 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 KeyManagement is bound to the host.



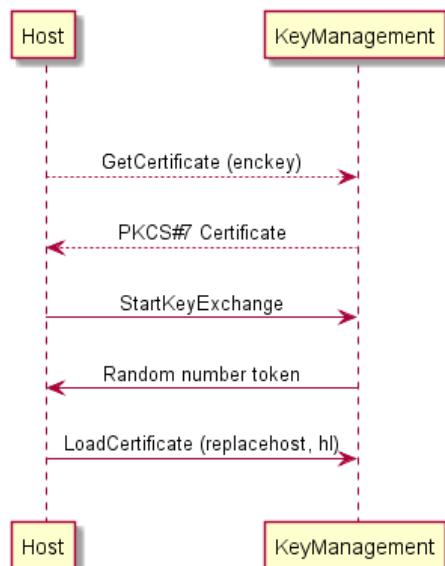
NB: Dotted lines represent commands that are only required if the KeyManagement encryption certificate has not been previously stored by the host.

## TR34 Force REBIND To New Host

---

#### XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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 KeyManagement is bound to the host.

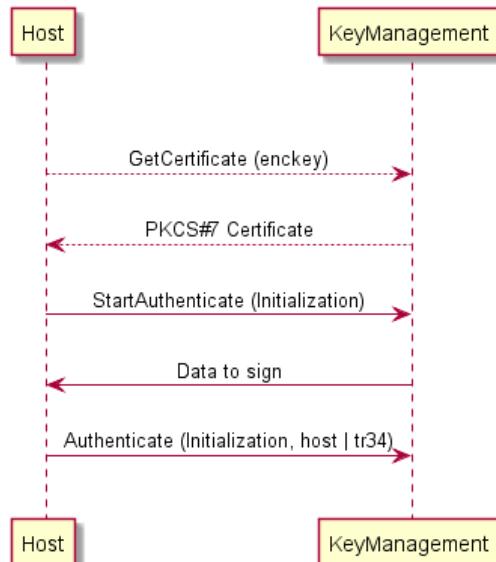


NB: Dotted lines represent commands that are only required if the KeyManagement encryption certificate has not been previously stored by the host. Although the random number token is requested as part of this operation, it is discarded by the host and is not actually used in the Force Rebind token.

## 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 KeyManagement is bound to the host.



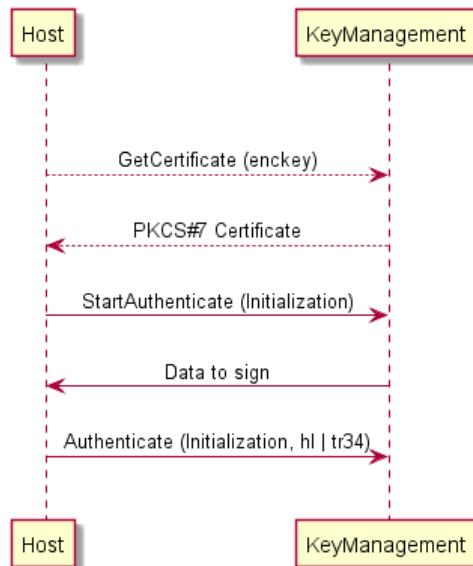
NB: Dotted lines represent commands that are only required if the KeyManagement encryption certificate has not been previously stored by the host

---

## 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 KeyManagement is bound to the host.



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

## EMV Support

---

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

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

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

### Keys loading

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

#### Certification Authority keys

These keys are provided in the following formats:

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

- 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 CAformat**

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	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
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.

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

### **SHA-1 Digest**

---

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

## **ImportKey command Input-Output Parameters**

---

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

### **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**

<b>Parameter Name</b>	<b>Example Value</b>
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
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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'

Parameter Name	Example Value
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'
keyAttributes.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'

---

Parameter Name	Example Value
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'

Parameter Name	Example Value
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 KeyManagement 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
"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*

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

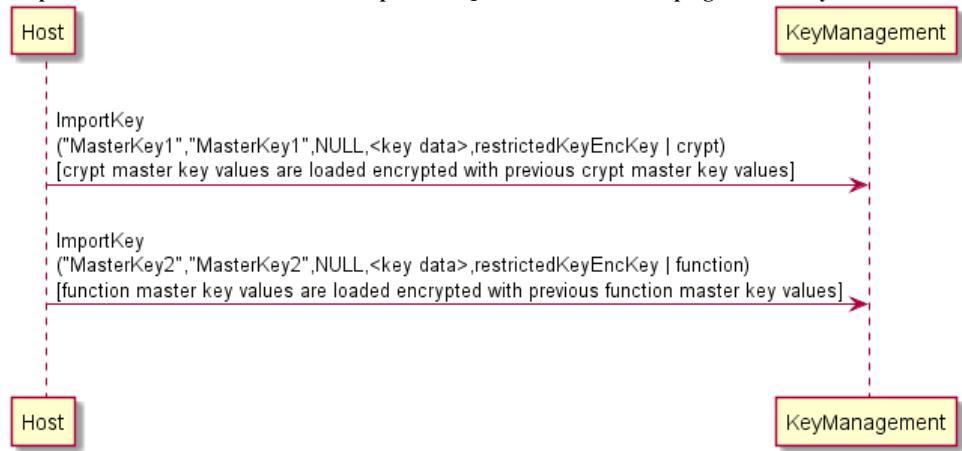
## RestrictedKeyEndKey Command Usage

---

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:



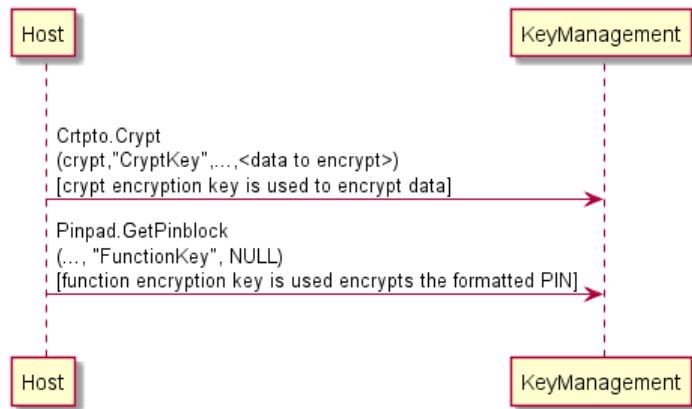
New master keys loaded with restrictedKeyEncKey flag, encrypted with themselves



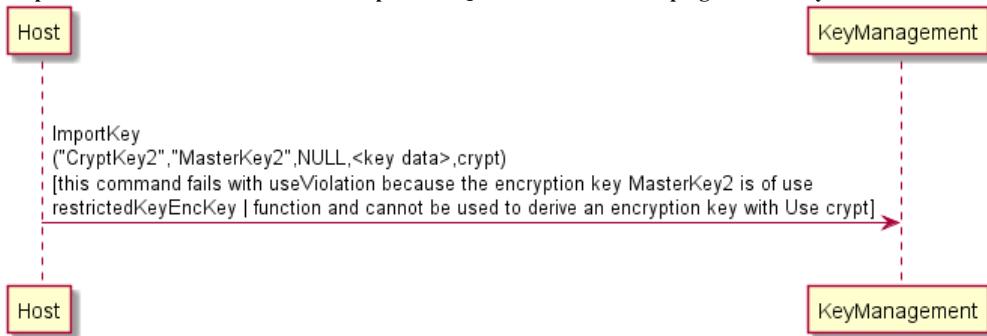
Loading derived keys:



Usage sample for derived keys



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



## Commands

### KeyManagement.GetStatus

---

#### Description

This command returns several kinds of status information

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		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"
  }
}
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
common <b>(Required)</b>			Status information common to all XFS4IoT services.
common.device	string		Specifies the state of the device.
common.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.
common.guideLights	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.
common.guideLights.flashRate	string		Indicates the current flash rate of the guidelight.
common.guideLights.color	string		Indicates the current color of the guidelight.
common.guideLights.direction	string		Indicates the current direction of the guidelight.
common.devicePosition	string		Position of the device.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.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
common.antiFraudModule	string		Specifies the state of the anti-fraud module
cardReader.			
cardReader.timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
cardReader.			

Name	Type	Default	Description
cardReader.media	string		<p>Specifies the media state of the device as one of the following values. This status is independent of any media in the parking stations.</p> <p><b>notSupported</b>&lt;br&gt; Capability to report media position is not supported by the device (e.g. a typical swipe reader or contactless chip card reader).</p> <p><b>unknown</b>&lt;br&gt; The media state cannot be determined with the device in its current state (e.g. the value of device is noDevice, powerOff, offline or hwerror).</p> <p><b>present</b>&lt;br&gt; Media is present in the device, not in the entering position and not jammed. A card in a parking station is not considered to be present. On the latched dip device, this indicates that the card is present in the device and the card is unlatched.</p> <p><b>notPresent</b>&lt;br&gt; Media is not present in the device and not at the entering position.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device; operator intervention is required.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of a motorized device.</p> <p><b>latched</b>&lt;br&gt; Media is present and latched in a latched dip card unit. This means the card can be used for chip card dialog.</p>
cardReader.retainBin	string		Specifies the state of the retain bin.

Name	Type	Default	Description
cardReader.security	string		<p>Specifies the state of the security unit as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; No security module is available.</p> <p><b>notReady</b>&lt;br&gt; The security module is not ready to process cards or is inoperable.</p> <p><b>notPresent</b>&lt;br&gt; The security module is open and ready to process cards.</p>
cardReader.numberCards	integer		<p>The number of cards retained; applicable only to motor driven card units, for non-motorized card units this value is zero. This value is persistent it is reset to zero by the resetCount command.</p>

Name	Type	Default	Description
cardReader.chipPower	string		<p>Specifies the state of the chip controlled by this service. Depending on the value of capabilities response, this can either be the chip on the currently inserted user card or the chip on a permanently connected chip card. The state of the chip is one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability to report the state of the chip is not supported by the ID card unit device. This value is returned for contactless chip card readers.</p> <p><b>unknown</b>&lt;br&gt; The state of the chip cannot be determined with the device in its current state.</p> <p><b>online</b>&lt;br&gt; The chip is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).</p> <p><b>busy</b>&lt;br&gt; The chip is present, powered on, and busy (unable to process an Execute command at this time).</p> <p><b>poweredOff</b>&lt;br&gt; The chip is present, but powered off (i.e. not contacted).</p> <p><b>noDevice</b>&lt;br&gt; A card is currently present in the device, but has no chip.</p> <p><b>hwerror</b>&lt;br&gt; The chip is present, but inoperable due to a hardware error that prevents it from being used (e.g. MUTE, if there is an unresponsive card in the reader).</p> <p><b>noCard</b>&lt;br&gt; There is no card in the device.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor	object		<p>Status information for XFS4IoT services implementing the CashAcceptor interface. This will be omitted if the CashAcceptor interface is not supported.</p>
cashAcceptor.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty.</li> <li>"full": The intermediate stacker is full. This may also be reported during a cash-in transaction where a limit specified by CashAcceptor.SetCashInLimit has been reached.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
cashAcceptor.stackerItems	string		<p>This field informs the application whether items on the intermediate stacker have been in customer access. Following values are possible:</p> <ul style="list-style-type: none"> <li>"customerAccess": Items on the intermediate stacker have been in customer access. If the device is a cash recycler then the items on the intermediate stacker may be there as a result of a previous cash-out operation.</li> <li>"noCustomerAccess": Items on the intermediate stacker have not been in customer access.</li> <li>"accessUnknown": It is not known if the items on the intermediate stacker have been in customer access.</li> <li>"noItems": There are no items on the intermediate stacker or the physical device has no intermediate stacker.</li> </ul> <p>Supplies the state of the banknote reader. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The banknote reader is in a good state.</li> <li>"inoperable": The banknote reader is inoperable.</li> <li>"unknown": Due to a hardware error or other condition, the state of the banknote reader cannot be determined.</li> <li>"notSupported": The physical device has no banknote reader.</li> </ul>
cashAcceptor.banknoteReader	string		

Name	Type	Default	Description
cashAcceptor.dropBox	boolean		The drop box is an area within the CashAcceptor where items which have caused a problem during an operation are stored. This field specifies the status of the drop box. TRUE means that some items are stored in the drop box due to a cash-in transaction which caused a problem. FALSE indicates that the drop box is empty.
cashAcceptor.positions	array		<p>Array of structures for each position from which items can be accepted.</p> <p>Supplies the input or output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>
cashAcceptor.positions.position	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul> <p>The status of the input or output position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>
cashAcceptor.positions.positionStatus	string		

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <p>"ok": The transport is in a good state.</p> <p>"inoperative": The transport is inoperative due to a hardware failure or media jam.</p> <p>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</p> <p>"notSupported": The physical device has no transport or transport state reporting is not supported.</p>
cashAcceptor.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous dispense operation. Following values are possible:</p> <p>"empty": The transport is empty.</p> <p>"notEmpty": The transport is not empty.</p> <p>"notEmptyCustomer": Items which a customer has had access to are on the transport.</p> <p>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</p> <p>"notSupported": The device is not capable of reporting whether items are on the transport.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>
cashAcceptor.mixedMode	string		<p>Reports if Mixed Media mode is active. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notActive": Mixed Media transactions are not supported by the device or Mixed Media mode is not activated.</li> <li>"active": Mixed Media mode using the CashAcceptor and ItemProcessor interfaces is activated.</li> </ul>
cashDispenser	object		<p>Status information for XFS4IoT services implementing the CashDispenser interface. This will be omitted if the CashDispenser interface is not supported.</p>

Name	Type	Default	Description
cashDispenser.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. These bills are typically present on the intermediate stacker as a result of a retract operation or because a dispense has been performed without a subsequent present.</p> <p>Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty. The items have not been in customer access.</li> <li>"notEmptyCustomer": The intermediate stacker is not empty. The items have been in customer access. If the device is a recycler then the items on the intermediate stacker may be there as a result of a previous cash-in operation.</li> <li>"notEmptyUnknown": The intermediate stacker is not empty. It is not known if the items have been in customer access.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>
cashDispenser.positions	array		Array of structures for each position to which items can be dispensed or presented.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.position	string		<p>Supplies the output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"left": Left output position.</li> <li>"right": Right output position.</li> <li>"center": Center output position.</li> <li>"top": Top output position.</li> <li>"bottom": Bottom output position.</li> <li>"front": Front output position.</li> <li>"rear": Rear output position.</li> </ul>
cashDispenser.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul>

Name	Type	Default	Description
cashDispenser.positions.positionStatus	string		<p>Returns information regarding items which may be at the output position. If the device is a recycler it is possible that the output position will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>
cashDispenser.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>

Name	Type	Default	Description
cashDispenser.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>
cashDispenser.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement	object		<p>Status information for XFS4IoT services implementing the CashManagement interface. This will be omitted if the CashManagement interface is not supported.</p>
cashManagement.safeDoor	string		<p>Supplies the state of the safe door. Following values are possible:</p> <ul style="list-style-type: none"> <li>"doorNotSupported": Physical device has no safe door or safe door state reporting is not supported.</li> <li>"doorOpen": Safe door is open.</li> <li>"doorClosed": Safe door is closed.</li> <li>"doorUnknown": Due to a hardware error or other condition, the state of the safe door cannot be determined.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement.dispenser	string		<p>Supplies the state of the logical cash units for dispensing. Following values are possible:</p> <p>"ok": All cash units present are in a good state.</p> <p>"cashUnitState": One or more of the cash units is in a low, empty, inoperative or manipulated condition. Items can still be dispensed from at least one of the cash units.</p> <p>"cashUnitStop": Due to a cash unit failure dispensing is impossible. No items can be dispensed because all of the cash units are in an empty, inoperative or manipulated condition. This state may also occur when a reject/retract cash unit is full or no reject/retract cash unit is present, or when an application lock is set on every cash unit which can be locked.</p> <p>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</p>

Name	Type	Default	Description
cashManagement.acceptor	string		<p>Supplies the state of the cash units for accepting cash. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": All cash units present are in a good state.</li> <li>"cashUnitState": One or more of the cash units is in a high, full, inoperative or manipulated condition. Items can still be accepted into at least one of the cash units.</li> <li>"cashUnitStop": Due to a cash unit failure accepting is impossible. No items can be accepted because all of the cash units are in a full, inoperative or manipulated condition. This state may also occur when a retract cash unit is full or no retract cash unit is present, or when an application lock is set on every cash unit, or when Level 2/3 notes are to be automatically retained within cash units, but all of the designated cash units for storing them are full or inoperative.</li> <li>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</li> </ul>
pinPad			<p>Status information for XFS4IoT services implementing the PinPad interface. This will be omitted if the PinPad interface is not supported.</p>
crypto	object		<p>Status information for XFS4IoT services implementing the Crypto interface. This will be omitted if the Crypto interface is not supported.</p>

Name	Type	Default	Description
keyManagement	object		Status information for XFS4IoT services implementing the KeyManagement interface. This will be omitted if the KeyManagement interface is not supported.
keyboard	object		Status information for XFS4IoT services implementing the Keyboard interface. This will be omitted if the Keyboard interface is not supported.
textTerminal	object		Status information for XFS4IoT services implementing the TextTerminal interface. This will be omitted if the TextTerminal interface is not supported.
printer	object		Status information for XFS4IoT services implementing the Printer interface. This will be omitted if the Printer interface is not supported.

Name	Type	Default	Description
printer.media	string		<p>Specifies the state of the print media (i.e. receipt, statement, passbook, etc.) as one of the following values. This field does not apply to journal printers:</p> <p><b>notSupported</b>&lt;br&gt; The capability to report the state of the print media is not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; The state of the print media cannot be determined with the device in its current state.</p> <p><b>present</b>&lt;br&gt; Media is in the print position, on the stacker or on the transport (i.e. a passbook in the parking station is not considered to be present). On devices with continuous paper supplies, this value is set when paper is under the print head. On devices with no supply or individual sheet supplies, this value is set when paper/media is successfully inserted/loaded.</p> <p><b>notPresent</b>&lt;br&gt; Media is not in the print position or on the stacker.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of the device.</p> <p><b>retracted</b>&lt;br&gt; Media was retracted during the last command which controlled media.</p>

Name	Type	Default	Description
printer.paper	object		<p>Specifies the state of paper supplies as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The paper supply is full.</p> <p><b>low</b>&lt;br&gt; The paper supply is low.</p> <p><b>out</b>&lt;br&gt; The paper supply is empty.</p> <p><b>jammed</b>&lt;br&gt; The paper supply is jammed.</p>
printer.paper.upper	string		The state of the upper paper supply.
printer.paper.lower	string		The state of the lower paper supply.
printer.paper.external	string		The state of the external paper supply.
printer.paper.aux	string		The state of the auxiliary paper supply.
printer.paper.aux2	string		The state of the second auxiliary paper supply.
printer.paper.park	string		The state of the parking station paper supply.

Name	Type	Default	Description
printer.toner	string		<p>Specifies the state of the toner or ink supply or the state of the ribbon as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of toner or ink supply or the ribbon cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The toner or ink supply is full or the ribbon is OK.</p> <p><b>low</b>&lt;br&gt; The toner or ink supply is low or the print contrast with a ribbon is weak.</p> <p><b>out</b>&lt;br&gt; The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.</p>
printer.ink	string		<p>Specifies the status of the stamping ink in the printer as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the stamping ink supply cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; Ink supply in device is full.</p> <p><b>low</b>&lt;br&gt; Ink supply in device is low.</p> <p><b>out</b>&lt;br&gt; Ink supply in device is empty.</p>

Name	Type	Default	Description
printer.lamp	string		<p>Specifies the status of the printer imaging lamp as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the imaging lamp cannot be determined with device in its current state.</p> <p><b>ok</b>&lt;br&gt; The lamp is OK.</p> <p><b>fading</b>&lt;br&gt; The lamp should be changed.</p> <p><b>inop</b>&lt;br&gt; The lamp is inoperative.</p>
printer.retractBins	array		<p>An array of bin state objects. If no retain bins are supported, the array will be empty.</p> <p>Specifies the state of the printer retract bin as one of the following:</p> <p><b>ok</b>&lt;br&gt; The retract bin of the printer is in a healthy state.</p> <p><b>full</b>&lt;br&gt; The retract bin of the printer is full.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>high</b>&lt;br&gt; The retract bin of the printer is nearly full.</p> <p><b>missing</b>&lt;br&gt; The retract bin is missing.</p>
printer.retractBins.count	integer		<p>The number of media retracted to this bin. This value is persistent; it may be reset to zero by the Printer.ResetCount command.</p>
printer.mediaOnStacker	integer		<p>The number of media on stacker; applicable only to printers with stacking capability.</p>

Name	Type	Default	Description
printer.paperType	object		<p>Specifies the type of paper loaded as one of the following:</p> <p><b>unknown</b>&lt;br&gt; No paper is loaded, reporting of this paper type is not supported or the paper type cannot be determined.</p> <p><b>single</b>&lt;br&gt; The paper can be printed on only one side.</p> <p><b>dual</b>&lt;br&gt; The paper can be printed on both sides.</p>
printer.paperType.upper	string		The upper paper supply paper type.
printer.paperType.lower	string		The lower paper supply paper type.
printer.paperType.external	string		The external paper supply paper type.
printer.paperType.aux	string		The auxilliary paper supply paper type.
printer.paperType.aux2	string		The second auxilliary paper supply paper type.
printer.paperType.park	string		The parking station paper supply paper type.
printer.blackMarkMode	string		<p>Specifies the status of the black mark detection and associated functionality:</p> <p><b>notSupported</b>&lt;br&gt; Black mark detection is not supported.</p> <p><b>unknown</b>&lt;br&gt; The status of the black mark detection cannot be determined.</p> <p><b>on</b>&lt;br&gt; Black mark detection and associated functionality is switched on.</p> <p><b>off</b>&lt;br&gt; Black mark detection and associated functionality is switched off.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
encryptionState <span style="background-color: black; color: white; padding: 2px;">(Required)</span>	string		Specifies the state of the encryption module
certificateState <span style="background-color: black; color: white; padding: 2px;">(Required)</span>	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": {
    "common": {
      "device": "online",
      "extra": [
        "string"
      ],
      "guideLights": [
        {}
      ],
      "devicePosition": "inposition",
      "powerSaveRecoveryTime": 0,
      "antiFraudModule": "notSupp"
    },
    "cardReader": {
      "timeout": "5000",
      "media": "notSupported",
      "retainBin": "notSupported",
      "security": "notSupported",
      "numberCards": 0,
      "chipPower": "notSupported"
    },
    "cashAcceptor": {
      "intermediateStacker": "empty",
      "stackerItems": "customerAccess",
      "banknoteReader": "customokerAccess",
      "dropBox": true,
      "positions": [
        {
          "position": "inLeft",
          "shutter": "closed",
          "positionStatus": "empty",
          "transport": "ok",
          "transportStatus": "empty",
          "jammedShutterPosition": "notSupported"
        }
      ],
      "mixedMode": "notActive"
    },
    "cashDispenser": {
      "intermediateStacker": "empty"
    }
  }
}
```

```
        "positions": [
            {
                "position": "left",
                "shutter": "closed",
                "positionStatus": "empty",
                "transport": "ok",
                "transportStatus": "empty",
                "jammedShutterPosition": "notSupported"
            }
        ],
        "cashManagement": {
            "safeDoor": "doorNotSupported",
            "dispenser": "ok",
            "acceptor": "ok"
        },
        "pinPad": null,
        "crypto": {},
        "keyManagement": {},
        "keyboard": {},
        "textTerminal": {},
        "printer": {
            "media": "notSupported",
            "paper": {
                "upper": "notSupported",
                "lower": "notSupported",
                "external": "notSupported",
                "aux": "notSupported",
                "aux2": "notSupported",
                "park": "notSupported",
                "property1": "notSupported",
                "property2": "notSupported"
            },
            "toner": "notSupported",
            "ink": "notSupported",
            "lamp": "notSupported",
            "retractBins": [
                {
                    "state": "unknown",
                    "count": 0
                }
            ],
            "mediaOnStacker": 0,
            "paperType": {
                "upper": "unknown",
                "lower": "unknown",
                "external": "unknown",
                "aux": "unknown",
                "aux2": "unknown",
                "park": "unknown",
                "property1": "unknown",
                "property2": "unknown"
            },
            "blackMarkMode": "notSupported"
        },
        "encryptionState": "ready",
        "certificateState": "unknown"
    }
}
```

## Event Messages

### KeyManagement.GetKeyDetail

#### Description

This command returns extended detailed information about the keys in the encryption module, including des,dukpt,aes,rsa private and public keys. This command will also return information on all keys loaded during manufacture that can be used by applications. Details relating to the keys loaded using OPT (via the ZKA ProtlsoPs 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	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	The original message name, for example "CardReader.Status"

##### Message Payload

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

##### Example Message (generated)

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

#### Completion Message

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
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 <b>(Required)</b>	object		Specifies whether the key has been loaded (imported from Application or locally from Operator).

Name	Type	Default	Description
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 <b>(Required)</b>	string		Specifies the intended function of the key. See [Reference 35. ANS X9 TR-31 2018] for all possible values.
keyDetail.keyBlockInfo.algorithm <b>(Required)</b>	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 <b>(Required)</b>	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.

Name	Type	Default	Description
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 <b>(Required)</b>	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": null
}
```

**Event Messages****KeyManagement.Initialization****Description**

The encryption module must be initialized before any encryption function can be used. Every call to Initialization destroys all application keys that have been loaded or imported; it does not affect those keys loaded during manufacturing. Usually this command is called by an operator task and not by the application program. Public keys imported under the rsa Signature based remote key loading scheme when public key deletion authentication is required will not be affected. However, if this command is requested in authenticated mode, public keys that require authentication for deletion will be deleted. This includes public keys imported under either the rsa Signature based remote key loading scheme or the TR34 RSA Certificate based remote key loading scheme. Initialization also involves loading 'initial' 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

### XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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. this field is not required if an indent is not required.
key	string		The value of the encryption key formatted in base64. this field is not required if no specific key name required.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
identification	string		The value of the ID key encrypted by the encryption key formatted in base64. This value can be used as authorization for the ImportKey command, this field is not set if no authorization required.

### Example Message (generated)

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

### Event Messages

---

## KeyManagement.DeriveKey

---

### Description

The encryption key in the secure key buffer or passed by the application is loaded in the encryption module. The key can be passed in clear text mode or encrypted with an accompanying 'key encryption key'. A key can be loaded in multiple unencrypted parts by combining the construct or secureConstruct value with the final usage flags within the use field. If the construct flag is used then the application must provide the key data through the value parameter, If secureConstruct is used then the encryption key part in the secure key buffer previously populated with the SecureKeyEntry command is used and

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	integer		Specifies the algorithm that is used for derivation.
key <b>(Required)</b>	string		Specifies the name where the derived key will be stored.
keyGenKey <b>(Required)</b>	string		Specifies the name of the key generating key that is used for the derivation.
startValueKey	string		Specifies the name of the stored key used to decrypt the startValue to obtain the Initialization Vector. If this field is not set, startValue is used as the Initialization Vector.
startValue <b>(Required)</b>	string		Des initialization vector for the encryption step within the derivation.
padding <b>(Required)</b>	integer		Specifies the padding character for the encryption step within the derivation. The valid range is 0x00 to 0xFF
inputData <b>(Required)</b>	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 initialization command. See idKey in Capabilities for whether this value is required. If not required, this field should not be set. The use of this parameter is vendor dependent.

### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

### KeyManagement.Reset

---

#### Description

Sends a service reset to the Service Provider.

#### Command Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

## KeyManagement.ImportKey

---

## Description

The encryption key passed by the application is loaded in the encryption module. For secret keys, the key must be passed encrypted with an accompanying "key encrypting key" or "key block protection key". For public keys, they key is not required to be encrypted but is required to have verification data in order to be loaded.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	string		Specifies the name of key being loaded.
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.
keyAttributes.keyUsage. keyAttributes.keyUsage.	integer string		
keyAttributes.propkeyUsage	integer	0	if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.algorithm. keyAttributes.algorithm.	integer string		
keyAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.modeOfUse. keyAttributes.modeOfUse.	integer string		
keyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
keyAttributes.cryptoMethod <b>(Required)</b>	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 <b>(Required)</b>	string		Specifies the value of key to be loaded formatted in base64. If it is an rsa key the first 4 bytes contain the exponent and the following 128 the modulus.
decryptKey	string		Specifies the name of the key used to decrypt the key being loaded. If value contains a TR-31 key block, then decryptKey is the name of the key block protection key that is used to verify and decrypt the key block. This field is not required if the data in Value is not encrypted.
decryptMethod	string		Specifies the cryptographic method that shall be used with the key specified by decryptKey. The device shall use this method to decrypt the encrypted value in the value parameter. For a list of valid values see the cryptoMethod field in the decryptAttributes capability field. 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. if this field is not set when no verification is needed before importing the key.
verifyKey	string		Specifies the name of the previously loaded key which will be used to verify the verificationData. This field is not required when no verification is needed before importing the key.
verifyAttributes	object		This parameter specifies the encryption algorithm, cryptographic method, and mode to be used to verify this command or to generate verification output data. Verifying input data will result in no verification output data. For a list of valid values see the verifyAttributes capability fields. This field must not be set if verificationData is not required.
verifyAttributes.keyUsage. verifyAttributes.keyUsage.	integer string		
verifyAttributes.propkeyUsage	integer 0		if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.algorithm. verifyAttributes.algorithm.	integer string		
verifyAttributes.propAlgorithm	integer 0		if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.modeOfUse.	integer		

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
verifyAttributes.modeOfUse	string		
verifyAttributes.propmodeOfUse	integer 0		if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.cryptoMethod <b>(Required)</b>	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.
vendorAttributes	string		Specifies the vendor attributes of the key to be imported. Refer to vendor documentation for details. If no vendor attributes are used, then this field must not be set.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
verificationData	string		The verification data. This field is not set if there is no verification data.
verifyAttributes	object		This parameter specifies the encryption algorithm, cryptographic method, and mode used to verify this command For a list of valid values see the verifyAttributes capability fields. This field is not set if there is no verification data.
verifyAttributes.keyUsage. verifyAttributes.keyUsage.	integer string		
verifyAttributes.propkeyUsage	integer 0		if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.algorithm. verifyAttributes.algorithm.	integer string		
verifyAttributes.propAlgorithm	integer 0		if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.modeOfUse. verifyAttributes.modeOfUse.	integer string		
verifyAttributes.propmodeOfUse	integer 0		if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.cryptoMethod <b>(Required)</b>	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.
keyLength <b>(Required)</b>	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": 0,
      "propKeyUsage": 0,
      "algorithm": 0,
      "propAlgorithm": 0,
      "modeOfUse": 0,
      "propModeOfUse": 0,
      "cryptoMethod": "none"
    },
    "keyLength": 0
  }
}
```

## Event Messages

---

### **KeyManagement.DeleteKey**

---

#### Description

This command can 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	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	The original message name, for example "CardReader.Status"

##### Message Payload

Name	Type	Default	Description
key	string		Specifies the name of key being deleted. if this field is not specified or an empty string. all keys are deleted.

##### Example Message (generated)

---

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

## KeyManagement.ExportRSAIssuerSignedItem

---

## Description

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

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

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
exportItemType	string		Defines the type of data item to be exported from the device.
name	string		Specifies the name of the public key to be exported. 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

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

### Example Message (generated)

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

## Event Messages

---

# KeyManagement.GenerateRSAKeyPair

---

### Description

This command will generate a new rsa key pair. The public key generated as a result of this command can subsequently be obtained by calling ExportRSAEPPSignedItem. The newly generated key pair can only be used for the use defined in the dwUse flag. This flag defines the use of the private key; its public key can only be used for the inverse function.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
timeout	integer0		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.

Name	Type	Default	Description
use	string		Specifies what the private key component of the key pair can be used for. The public key part can only be used for the inverse function. For example, if the rsaPrivateSign use is specified, then the private key can only be used for signature generation and the partner public key can only be used for verification.
modulusLength	integer		Specifies the number of bits for the modulus of the rsa key pair to be generated. When zero is specified then the device will be responsible for defining the length.
exponentValue	string		Specifies the value of the exponent of the rsa key pair to be generated

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

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

## Event Messages

---

## KeyManagement.ExportRSAEPPSignedItem

---

### Description

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

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

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
exportItemType	string		Defines the type of data item to be exported from the device

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
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": {
    "completionCode": "success",
    "errorDescription": "string",
    "value": "string",
    "selfSignature": "string",
    "signature": "string"
  }
}
```

#### Event Messages

## KeyManagement.GetCertificate

#### Description

This command is used to read out the encryptor'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 <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
certificate <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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": {  
        "completionCode": "success",  
        "errorDescription": "string",  
        "certificate": "string"  
    }  
}
```

## Event Messages

### KeyManagement.ReplaceCertificate

#### Description

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

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
newCertificateData <b>(Required)</b>	string		A pkcs #7 using a Digest-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": {
    "completionCode": "success",
    "errorDescription": "string",
    "newCertificateData": "string"
  }
}
```

**Event Messages****KeyManagement.StartKeyExchange****Description**

This command is used to start communication with the host, including transferring the host'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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
randomItem	string		A randomly generated number created by the encryptor formatted in base 64. If the device does not support random number generation and verification, a zero length random number is returned and an empty string is returned.

#### Example Message (generated)

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

## Event Messages

---

# KeyManagement.GenerateKCV

---

### Description

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

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	string		Specifies the name of key that should be used to generate the kcv.
keyCheckMode <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
kcv	<b>(Required)</b>	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": {
    "completionCode": "success",
    "errorDescription": "string",
    "kcv": "string"
  }
}
```

## KeyManagement.LoadCertificate

---

### Description

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

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	string		Specifies the method to use to load the certificate
signer <b>(Required)</b>	string		Specifies the signer of the certificate to be loaded
certificateData <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

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

### Example Message (generated)

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

## Event Messages

---

### KeyManagement.StartAuthenticate

---

#### Description

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

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		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 <b>(Required)</b>		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.

```
{
  "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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
internalCmdResult <b>(Required)</b>	string		Result from the command referenced by execution command. If the data within payload is invalid or cannot be used for some reason, then hInternalCmdResult will return an error but the result of this command will be ok.
dataToSign	string		The data that must be signed by one of the authorities indicated by signers before the command referenced by execution command can be executed. If the command specified by execution command does not require authentication, then this field is not set string and the command result is success.
signers <b>(Required)</b>	object		Specifies the allowed signers of the data as a combination
signers.none	boolean		Authentication is not required

Name	Type	Default	Description
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 IpsKey and the CBC MAC algorithm.
signers.cmac	boolean		A MAC is calculated over the data using IpsKey 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": {
    "completionCode": "success",
    "errorDescription": "string",
    "internalCmdResult": "string",
    "dataToSign": "string",
    "signers": {
      "none": true,
      "certhost": true,
      "sighost": true,
      "ca": true,
      "hl": true,
      "tr34": true,
      "cbcmac": true,
      "cmac": true,
      "reserved_1": true,
      "reserved_2": true,
      "reserved_3": true
    }
  }
}
```

## Event Messages

---

# Unsolicited Events

## **KeyManagement.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	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		The original message name, for example "CardReader.Status"

### Message Payload

---

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

**Example Message (generated)**

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

## KeyManagement.IllegalKeyAccessEvent

**Description**

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

**Message Header**

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

**Message Payload**

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

**Example Message (generated)**

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

## KeyManagement.CertificateChangeEvent

### Description

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

### Message Header

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

### Message Payload

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

### Example Message (generated)

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

## XFS4IOT Sample Messaging for Keyboard Draft 0.0.4

This section describes the general interface for the following functions:

- Entering Personal Identification Numbers (PINs)
- Clear text data handling
- Function key handling

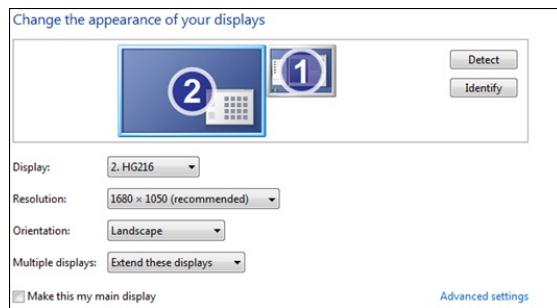
If the PIN pad device has local display capability, display handling should be handled using the Text Terminal Unit (TTU) interface. The adoption of this specification does not imply the adoption of a specific security standard.

- Only numeric PIN pads are handled in this specification.

### Documentation

#### Encrypting Touch Screen (ETS)

An encrypting touch screen device is a touch screen securely attached to a cryptographic device. It can be used as an alternative to an encrypting pin pad (EPP). It supports key management, encryption and decryption. It is assumed that the ETS is a combined device. It overlays a display monitor which is used to display lead-through for a transaction. It is assumed that the display monitor is part of the Windows desktop, and can be the Windows primary monitor or any other monitor on the desktop. E.g. the following diagram shows 2 monitors extended across the desktop, with monitor 1 being the primary monitor and the ETS being overlaid on monitor 2 whose origin is (-1680.0).



The touch screen can optionally be used as a "mouse" for application purposes, while PIN operations are not in progress or optionally when non-secure PIN commands are in progress.

The CEN interface supports two types of ETS

- Those which activate touch areas defined by the application.<br>
- Those which activate a random variation of touch areas defined by the application.<br>

The Service Provider, when reporting its capabilities, reports the absolute position of the ETS in Windows desktop coordinates. This allows the application to locate the ETS device in a multi-monitor system and relate it to a monitor on the desktop.

At any point in time, a single touch area of the ETS can operate in one of 4 modes:-

- **Mouse mode** - <br> a "touch" simulates a mouse click. This mode is optional. This may not be supported by some ETS devices. Configuration of the click is vendor specific. e.g. WM\_LBUTTONDOWN. This is also the mode that, if supported, is active when none of the other modes are active.
- **Data mode** - <br> a "touch" maps to an key and the value of the key is returned in an event (as in clear numeric entry using GetData).
- **PIN mode** - <br> a "touch" maps to an key and the value of the key is returned in an event only if the key pressed is not Fk0 through Fk9 (as in PIN entry using GetPin).
- **Secure mode** - <br> a "touch" maps to an key and the value of the key is returned in an event only if the key pressed is not Fk0 through Fk9 and not Fk\_A through Fk\_F (as in key entry using SecureKeyEntry).

The following concepts are introduced to define the relationship between the monitor and the ETS:-

- **Touch Key** – <br> an area of the monitor which reacts to touch in Data, PIN and Secure modes.
- **Touch Frame** – <br> an area of the monitor onto which Touch Keys can be placed. There can be one or more Touch Frames. There may be just one Touch Frame which covers the whole monitor. Areas within a Touch Frame, not defined as a Touch Key, do not react to touch. Generally in PIN and Secure modes, there would be only one Touch Frame covering the whole monitor. An empty Touch Frame disables that part of the monitor.
- **Mouse area** – <br> an area outside of all Touch Frames in which touches behave like a mouse.
- Thus Data, PIN and Secure modes operate in a single Touch Frame or multiple Touch Frames. Mouse mode operates outside a Touch Frame, and is optional.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Note that there is a perceived risk in separating the drawing functionality from the touch functionality, but this type of risk is present in today's keyboard based systems. e.g. An application can draw on a monitor to prompt the user to enter a PIN and then enables the EPP for clear data entry. So the risk is no different than with an EPP – the application has to be trusted.

Depending upon the type of device, the application must then either inform the Service Provider as to the active key positions in the form of Touch Frames and Touch Keys using the DefineLayout command, or obtain them from the Service Provider using the GetLayout command. This collection is now referred to as a "Touch Keyboard definition".

The application then uses the normal PIN commands to enable the touch keyboard definition on the ETS device:

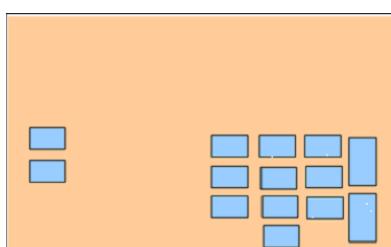
- PIN entry GetPin
- Clear data entry GetData
- Secure key entry SecureKeyEntry

These commands are referred to as "keyboard entry commands" throughout the remainder of this document.

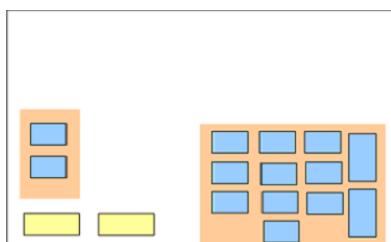
PCI compliance means that GetPin and SecureKeyEntry can only be used with a single Touch Frame that covers the entire monitor. i.e. Mouse mode cannot be mixed with either PIN or Secure mode. If a Touch Key (or areas) is defined for an key value and that key value is not subsequently specified as active in a GetPin, GetData or SecureKeyEntry command, then the Touch Key is made inactive.

Layouts defined with the DefineLayout command are persistent.

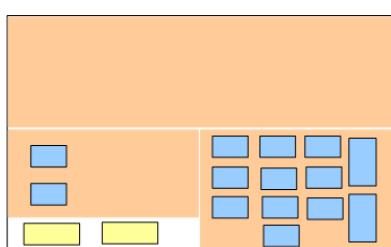
**Example 1** – this screen only uses Data mode – the entire screen is a Touch Frame. Mouse mode is not used.



**Example 2** – this shows a monitor with two Touch Frames and 14 Touch Keys. The space within the Touch Frames not defined by a Touch Key are inactive (do not respond to touch). All areas outside a Touch Frame operate in Mouse mode. This example shows two Mouse mode "keys". e.g. Windows "Button", HTML "BUTTON" or a custom control. Other touches in Mouse mode are normally dealt with by the application event engine. However, this can be restricted – see example 3.



**Example 3** – this screen uses Mouse and Data modes – Mouse mode is used only in a restricted area. The touch keyboard definition has 3 frames. Frame 1 has no Touch Keys. Frame 2 has 2 Touch Keys; Frame 3 has 12 Touch Keys.



## 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.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

**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	fk7	fkUnused
8	0	500	250	250	fk8	fkUnused
9	250	500	250	250	fk9	fkUnused
10	500	500	250	250	fkC	fkUnused
11	750	500	250	250	fkD	fkUnused
12	0	750	250	250	fkE	fkUnused
13	250	750	250	250	fkF	fkUnused
14	500	750	250	250		
15	750	750	250	250		

**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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

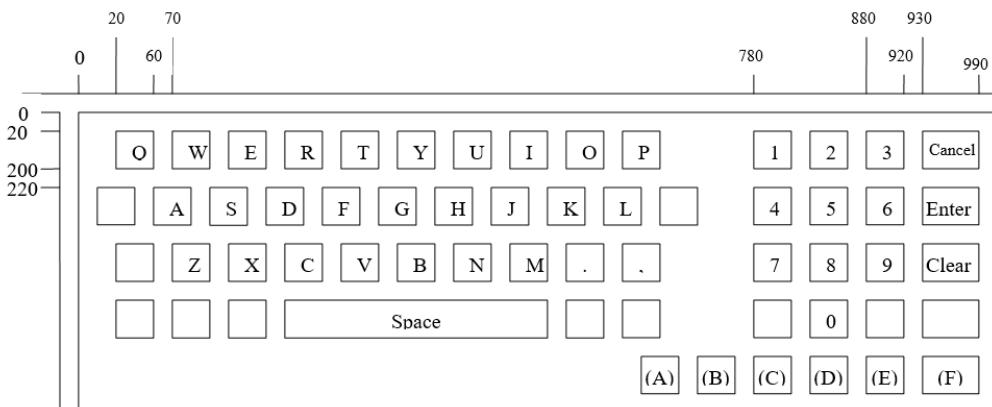
1 (B)	2 (C)	3 (D)	Clear
4 (E)	5 (F)	6 Cancel	
7 8 9 Enter			
0 (A)			
SHIFT			

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.

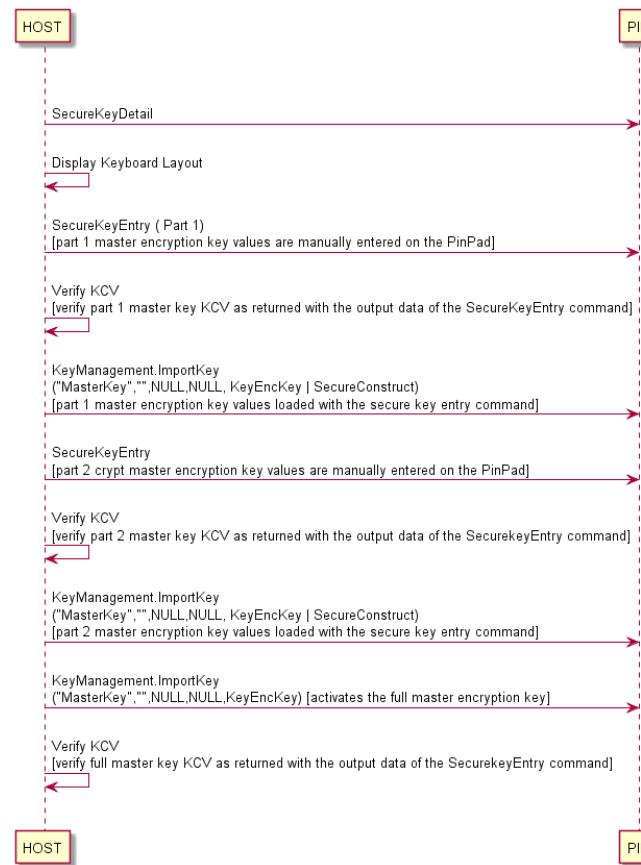
Index	xPos	yPos	xSize	ySize	fk	shiftfk
0	780	18	40	180	fk1	fkUnused
1	830	18	40	180	fk2	fkUnused
2	880	18	40	180	fk3	fkUnused
3	930	18	60	180	fkCancel	fkUnused
4	780	216	40	180	fk4	fkUnused
5	830	216	40	180	fk5	fkUnused
6	880	216	40	180	fk6	fkUnused
7	930	216	60	180	fkEnter	fkUnused
8	780	414	40	180	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

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Index	xPos	yPos	xSize	ySize	fk	shiftfk
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.



## Commands

### Keyboard.GetStatus

#### Description

This command returns several kinds of status information

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

**Message Payload**

Name	Type	Default	Description
common <b>(Required)</b>			Status information common to all XFS4IoT services.
common.device	string		Specifies the state of the device.
common.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.
common.guideLights	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.
common.guideLights.flashRate	string		Indicates the current flash rate of the guidelight.
common.guideLights.color	string		Indicates the current color of the guidelight.
common.guideLights.direction	string		Indicates the current direction of the guidelight.
common.devicePosition	string		Position of the device.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.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
common.antiFraudModule	string		Specifies the state of the anti-fraud module
cardReader.			
cardReader.timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
cardReader.			
cardReader.media	string		<p>Specifies the media state of the device as one of the following values. This status is independent of any media in the parking stations.</p> <p><b>notSupported</b>&lt;br&gt; Capability to report media position is not supported by the device (e.g. a typical swipe reader or contactless chip card reader).</p> <p><b>unknown</b>&lt;br&gt; The media state cannot be determined with the device in its current state (e.g. the value of device is noDevice, powerOff, offline or hwerror).</p> <p><b>present</b>&lt;br&gt; Media is present in the device, not in the entering position and not jammed. A card in a parking station is not considered to be present. On the latched dip device, this indicates that the card is present in the device and the card is unlatched.</p> <p><b>notPresent</b>&lt;br&gt; Media is not present in the device and not at the entering position.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device; operator intervention is required.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of a motorized device.</p> <p><b>latched</b>&lt;br&gt; Media is present and latched in a latched dip card unit. This means the card can be used for chip card dialog.</p>
cardReader.retainBin	string		Specifies the state of the retain bin.
cardReader.security	string		<p>Specifies the state of the security unit as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; No security module is available.</p> <p><b>notReady</b>&lt;br&gt; The security module is not ready to process cards or is inoperable.</p> <p><b>notPresent</b>&lt;br&gt; The security module is open and ready to process cards.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cardReader.numberCards	integer		<p>The number of cards retained; applicable only to motor driven card units, for non-motorized card units this value is zero. This value is persistent it is reset to zero by the resetCount command.</p>
cardReader.chipPower	string		<p>Specifies the state of the chip controlled by this service. Depending on the value of capabilities response, this can either be the chip on the currently inserted user card or the chip on a permanently connected chip card. The state of the chip is one of the following:</p> <ul style="list-style-type: none"> <li><b>notSupported</b>&lt;br&gt; Capability to report the state of the chip is not supported by the ID card unit device. This value is returned for contactless chip card readers.</li> <li><b>unknown</b>&lt;br&gt; The state of the chip cannot be determined with the device in its current state.</li> <li><b>online</b>&lt;br&gt; The chip is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).</li> <li><b>busy</b>&lt;br&gt; The chip is present, powered on, and busy (unable to process an Execute command at this time).</li> <li><b>poweredOff</b>&lt;br&gt; The chip is present, but powered off (i.e. not contacted).</li> <li><b>noDevice</b>&lt;br&gt; A card is currently present in the device, but has no chip.</li> <li><b>hwerror</b>&lt;br&gt; The chip is present, but inoperable due to a hardware error that prevents it from being used (e.g. MUTE, if there is an unresponsive card in the reader).</li> <li><b>noCard</b>&lt;br&gt; There is no card in the device.</li> </ul>
cashAcceptor	object		<p>Status information for XFS4IoT services implementing the CashAcceptor interface. This will be omitted if the CashAcceptor interface is not supported.</p>
cashAcceptor.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty.</li> <li>"full": The intermediate stacker is full. This may also be reported during a cash-in transaction where a limit specified by CashAcceptor.SetCashInLimit has been reached.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.stackerItems	string		<p>This field informs the application whether items on the intermediate stacker have been in customer access. Following values are possible:</p> <ul style="list-style-type: none"> <li>"customerAccess": Items on the intermediate stacker have been in customer access. If the device is a cash recycler then the items on the intermediate stacker may be there as a result of a previous cash-out operation.</li> <li>"noCustomerAccess": Items on the intermediate stacker have not been in customer access.</li> <li>"accessUnknown": It is not known if the items on the intermediate stacker have been in customer access.</li> <li>"noItems": There are no items on the intermediate stacker or the physical device has no intermediate stacker.</li> </ul>
cashAcceptor.banknoteReader	string		<p>Supplies the state of the banknote reader. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The banknote reader is in a good state.</li> <li>"inoperable": The banknote reader is inoperable.</li> <li>"unknown": Due to a hardware error or other condition, the state of the banknote reader cannot be determined.</li> <li>"notSupported": The physical device has no banknote reader.</li> </ul>
cashAcceptor.dropBox	boolean		<p>The drop box is an area within the CashAcceptor where items which have caused a problem during an operation are stored. This field specifies the status of the drop box. TRUE means that some items are stored in the drop box due to a cash-in transaction which caused a problem. FALSE indicates that the drop box is empty.</p>
cashAcceptor.positions	array		Array of structures for each position from which items can be accepted.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.position	string		<p>Supplies the input or output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>
cashAcceptor.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul> <p>The status of the input or output position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> </ul>
cashAcceptor.positions.positionStatus	string		<ul style="list-style-type: none"> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>
cashAcceptor.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous dispense operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>
cashAcceptor.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>
cashAcceptor.mixedMode	string		<p>Reports if Mixed Media mode is active. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notActive": Mixed Media transactions are not supported by the device or Mixed Media mode is not activated.</li> <li>"active": Mixed Media mode using the CashAcceptor and ItemProcessor interfaces is activated.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser	object		<p>Status information for XFS4IoT services implementing the CashDispenser interface. This will be omitted if the CashDispenser interface is not supported.</p>
cashDispenser.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. These bills are typically present on the intermediate stacker as a result of a retract operation or because a dispense has been performed without a subsequent present. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty. The items have not been in customer access.</li> <li>"notEmptyCustomer": The intermediate stacker is not empty. The items have been in customer access. If the device is a recycler then the items on the intermediate stacker may be there as a result of a previous cash-in operation.</li> <li>"notEmptyUnknown": The intermediate stacker is not empty. It is not known if the items have been in customer access.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>
cashDispenser.positions	array		<p>Array of structures for each position to which items can be dispensed or presented.</p>
cashDispenser.positions.position	string		<p>Supplies the output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"left": Left output position.</li> <li>"right": Right output position.</li> <li>"center": Center output position.</li> <li>"top": Top output position.</li> <li>"bottom": Bottom output position.</li> <li>"front": Front output position.</li> <li>"rear": Rear output position.</li> </ul>
cashDispenser.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.positionStatus	string		<p>Returns information regarding items which may be at the output position. If the device is a recycler it is possible that the output position will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>
cashDispenser.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>
cashDispenser.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>
cashManagement	object		<p>Status information for XFS4IoT services implementing the CashManagement interface. This will be omitted if the CashManagement interface is not supported.</p>
cashManagement.safeDoor	string		<p>Supplies the state of the safe door. Following values are possible:</p> <ul style="list-style-type: none"> <li>"doorNotSupported": Physical device has no safe door or safe door state reporting is not supported.</li> <li>"doorOpen": Safe door is open.</li> <li>"doorClosed": Safe door is closed.</li> <li>"doorUnknown": Due to a hardware error or other condition, the state of the safe door cannot be determined.</li> </ul>
cashManagement.dispenser	string		<p>Supplies the state of the logical cash units for dispensing. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": All cash units present are in a good state.</li> <li>"cashUnitState": One or more of the cash units is in a low, empty, inoperative or manipulated condition. Items can still be dispensed from at least one of the cash units.</li> <li>"cashUnitStop": Due to a cash unit failure dispensing is impossible. No items can be dispensed because all of the cash units are in an empty, inoperative or manipulated condition. This state may also occur when a reject/retract cash unit is present, or when an application lock is set on every cash unit which can be locked.</li> <li>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement.acceptor	string		<p>Supplies the state of the cash units for accepting cash. Following values are possible:</p> <p>"ok": All cash units present are in a good state.</p> <p>"cashUnitState": One or more of the cash units is in a high, full, inoperative or manipulated condition. Items can still be accepted into at least one of the cash units.</p> <p>"cashUnitStop": Due to a cash unit failure accepting is impossible. No items can be accepted because all of the cash units are in a full, inoperative or manipulated condition. This state may also occur when a retract cash unit is full or no retract cash unit is present, or when an application lock is set on every cash unit, or when Level 2/3 notes are to be automatically retained within cash units, but all of the designated cash units for storing them are full or inoperative.</p> <p>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</p>
pinPad			Status information for XFS4IoT services implementing the PinPad interface. This will be omitted if the PinPad interface is not supported.
crypto	object		Status information for XFS4IoT services implementing the Crypto interface. This will be omitted if the Crypto interface is not supported.
keyManagement	object		Status information for XFS4IoT services implementing the KeyManagement interface. This will be omitted if the KeyManagement interface is not supported.
keyboard	object		Status information for XFS4IoT services implementing the Keyboard interface. This will be omitted if the Keyboard interface is not supported.
textTerminal	object		Status information for XFS4IoT services implementing the TextTerminal interface. This will be omitted if the TextTerminal interface is not supported.
printer	object		Status information for XFS4IoT services implementing the Printer interface. This will be omitted if the Printer interface is not supported.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.media	string		<p>Specifies the state of the print media (i.e. receipt, statement, passbook, etc.) as one of the following values. This field does not apply to journal printers:</p> <p><b>notSupported</b>&lt;br&gt; The capability to report the state of the print media is not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; The state of the print media cannot be determined with the device in its current state.</p> <p><b>present</b>&lt;br&gt; Media is in the print position, on the stacker or on the transport (i.e. a passbook in the parking station) is not considered to be present). On devices with continuous paper supplies, this value is set when paper is under the print head. On devices with no supply or individual sheet supplies, this value is set when paper/media is successfully inserted/loaded.</p> <p><b>notPresent</b>&lt;br&gt; Media is not in the print position or on the stacker.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of the device.</p> <p><b>retracted</b>&lt;br&gt; Media was retracted during the last command which controlled media.</p>
printer.paper	object		<p>Specifies the state of paper supplies as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The paper supply is full.</p> <p><b>low</b>&lt;br&gt; The paper supply is low.</p> <p><b>out</b>&lt;br&gt; The paper supply is empty.</p> <p><b>jammed</b>&lt;br&gt; The paper supply is jammed.</p>
printer.paper.upper	string		The state of the upper paper supply.
printer.paper.lower	string		The state of the lower paper supply.
printer.paper.external	string		The state of the external paper supply.
printer.paper.aux	string		The state of the auxiliary paper supply.
printer.paper.aux2	string		The state of the second auxiliary paper supply.
printer.paper.park	string		The state of the parking station paper supply.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.toner	string		<p>Specifies the state of the toner or ink supply or the state of the ribbon as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of toner or ink supply or the ribbon cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The toner or ink supply is full or the ribbon is OK.</p> <p><b>low</b>&lt;br&gt; The toner or ink supply is low or the print contrast with a ribbon is weak.</p> <p><b>out</b>&lt;br&gt; The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.</p>
printer.ink	string		<p>Specifies the status of the stamping ink in the printer as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the stamping ink supply cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; Ink supply in device is full.</p> <p><b>low</b>&lt;br&gt; Ink supply in device is low.</p> <p><b>out</b>&lt;br&gt; Ink supply in device is empty.</p>
printer.lamp	string		<p>Specifies the status of the printer imaging lamp as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the imaging lamp cannot be determined with device in its current state.</p> <p><b>ok</b>&lt;br&gt; The lamp is OK.</p> <p><b>fading</b>&lt;br&gt; The lamp should be changed.</p> <p><b>inop</b>&lt;br&gt; The lamp is inoperative.</p>
printer.retractBins	array		<p>An array of bin state objects. If no retain bins are supported, the array will be empty.</p>
printer.retractBins.state	string		<p>Specifies the state of the printer retract bin as one of the following:</p> <p><b>ok</b>&lt;br&gt; The retract bin of the printer is in a healthy state.</p> <p><b>full</b>&lt;br&gt; The retract bin of the printer is full.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>high</b>&lt;br&gt; The retract bin of the printer is nearly full.</p> <p><b>missing</b>&lt;br&gt; The retract bin is missing.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printer.retractBins.count	integer		The number of media retracted to this bin. This value is persistent; it may be reset to zero by the Printer.ResetCount command.
printer.mediaOnStacker	integer		The number of media on stacker; applicable only to printers with stacking capability.
printer.paperType	object		Specifies the type of paper loaded as one of the following:  <b>unknown</b>   No paper is loaded, reporting of this paper type is not supported or the paper type cannot be determined.  <b>single</b>   The paper can be printed on only one side.  <b>dual</b>   The paper can be printed on both sides.
printer.paperType.upper	string		The upper paper supply paper type.
printer.paperType.lower	string		The lower paper supply paper type.
printer.paperType.external	string		The external paper supply paper type.
printer.paperType.aux	string		The auxilliary paper supply paper type.
printer.paperType.aux2	string		The second auxilliary paper supply paper type.
printer.paperType.park	string		The parking station paper supply paper type.
printer.blackMarkMode	string		Specifies the status of the black mark detection and associated functionality:  <b>notSupported</b>   Black mark detection is not supported.  <b>unknown</b>   The status of the black mark detection cannot be determined.  <b>on</b>   Black mark detection and associated functionality is switched on.  <b>off</b>   Black mark detection and associated functionality is switched off.
autoBeepMode <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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)

**Example Message (generated)**

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

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
payload : {
    "common": {
        "device": "online",
        "extra": [
            "string"
        ],
        "guideLights": [
            {}
        ],
        "devicePosition": "inposition",
        "powerSaveRecoveryTime": 0,
        "antiFraudModule": "notSupp"
    },
    "cardReader": {
        "timeout": "5000",
        "media": "notSupported",
        "retainBin": "notSupported",
        "security": "notSupported",
        "numberCards": 0,
        "chipPower": "notSupported"
    },
    "cashAcceptor": {
        "intermediateStacker": "empty",
        "stackerItems": "customerAccess",
        "banknoteReader": "customokerAccess",
        "dropBox": true,
        "positions": [
            {
                "position": "inLeft",
                "shutter": "closed",
                "positionStatus": "empty",
                "transport": "ok",
                "transportStatus": "empty",
                "jammedShutterPosition": "notSupported"
            }
        ],
        "mixedMode": "notActive"
    },
    "cashDispenser": {
        "intermediateStacker": "empty",
        "positions": [
            {
                "position": "left",
                "shutter": "closed",
                "positionStatus": "empty",
                "transport": "ok",
                "transportStatus": "empty",
                "jammedShutterPosition": "notSupported"
            }
        ]
    },
    "cashManagement": {
        "safeDoor": "doorNotSupported",
        "dispenser": "ok",
        "acceptor": "ok"
    },
    "pinPad": null,
    "crypto": {},
    "keyManagement": {},
    "keyboard": {},
    "textTerminal": {},
    "printer": {
        "media": "notSupported",
        "paper": {
            "upper": "notSupported",
            "lower": "notSupported",
            "external": "notSupported",
            "aux": "notSupported",
            "aux2": "notSupported",
            "park": "notSupported",
            "property1": "notSupported",
            "property2": "notSupported"
        },
        "toner": "notSupported",
        "ink": "notSupported",
        "lamp": "notSupported",
        "retractBins": [
            {
                "state": "unknown",
                "count": 0
            }
        ],
        "mediaOnStacker": 0,
        "paperType": [
            "upper": "unknown",
            "lower": "unknown"
        ]
    }
}
```

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
        "external": "unknown",
        "aux1": "unknown",
        "aux2": "unknown",
        "park": "unknown",
        "property1": "unknown",
        "property2": "unknown"
    },
    "blackMarkMode": "notSupported"
},
"autoBeepMode": "active"
}
```

### Event Messages

## Keyboard.GetFuncKeyDetail

### 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 PinEntry or DataEntry 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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	string		Mask for the fdk for which additional information is requested.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
funcMask <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		Specifies the code returned by this FDK, defined as one of the following values:
fdks.xPosition <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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

---

### Keyboard.GetSecureKeyDetail

---

#### 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 devices 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

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
keyEntryMode <b>(Required)</b>	string		Specifies the method to be used to enter the encryption key digits (including 'A' to 'F') during secure key entry.
funcKeyDetail <b>(Required)</b>	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 <b>(Required)</b>	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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
funcKeyDetail.funcMask.fkE	boolean	false	
funcKeyDetail.funcMask.fkF	boolean	false	
funcKeyDetail.funcMask.fkEnter	boolean	false	
funcKeyDetail.funcMask.fkCancel	boolean	false	
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.fdk	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.fdk.fdk	string		Specifies the code returned by this FDK, defined as one of the following values:
funcKeyDetail.fdk.xPosition <b>(Required)</b>	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.fdk.yPosition <b>(Required)</b>	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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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	
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 <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	integer		Specifies the maximum number of columns on the device (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 <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	integer		Specifies the maximum number of rows on the device (the rows are defined by the y 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.
hexKeys <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	array		Array to hexKeys describes the physical keys on the device, it does not include FDKs.
hexKeys.xPos <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; padding: 2px;">(Required)</span>	integer		Specifies the FK height expressed as a value between 1 and 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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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	
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)

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
"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
  }
}
```

```

        "fkK8": 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

---

### Keyboard.GetLayout

---

#### Description

This command allows an application to retrieve layout information for any 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 DataEntry command (2) PIN Entry mode which corresponds to the PinEntry command (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 <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
entryMode <b>(Required)</b>	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 DataEntry method.
entryMode.pin	boolean	false	Specifies that the layout be applied to the PinEntry 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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	integer		For ETS, specifies the width of the frame. For all other device types, this value is ignored
frames.ySize <b>(Required)</b>	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 <b>(Required)</b>	boolean	false	Specifies that the PIN device will randomly shift the layout in a horizontal direction
frames.floatAction.floatY <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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.fk	string		Specifies the FK code associated with the physical area in non-shifted mode.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
frames.fks.shiftFK	string		Specifies the FDK code associated with the physical key 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,
            "fk": "fk0",
            "shiftFK": "fk0"
          }
        ]
      }
    ]
  }
}
```

### Event Messages

## Keyboard.PinEntry

### Description

This function stores the pin entry via the pin pad device. 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 <b>(Required)</b>	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
type <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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
    }
  }
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
    "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,
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

```
        "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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

## Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
digits <small>(Required)</small>	integer		Specifies the number of PIN digits entered
completion <small>(Required)</small>	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 DataEntry

### **Example Message (generated)**

## Event Messages

- [Keyboard.KeyEvent](#)
  - [Keyboard.EnterDataEvent](#)
  - [Keyboard.LayoutEvent](#)

## Keyboard.DataEntry

### Description

All rights of exploitation in any form and by any means reserved worldwide for CEN national Members.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This function enables keyboard insecure mode and report entered key in clear text with solicited events. For PIN pad device, 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	integer		Specifies the maximum number of digits which can be returned to the application in the output parameter.
autoEnd <b>(Required)</b>	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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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",
    "maxLength": 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,
      "fkH": false,
      "fkL": false,
      "fkP": false,
      "fkR": false,
      "fkT": false,
      "fkU": false,
      "fkV": false,
      "fkW": false,
      "fkX": false,
      "fkY": false,
      "fkZ": 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,  
    "fkcancode": false  
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```

        "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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
keys <b>(Required)</b>	integer		Number of keys entered by the user
pinKeys <b>(Required)</b>	array		Array to the pinKey that contain the keys entered by the user
completion <b>(Required)</b>	string		Specifies the reason for completion of the entry

#### Example Message (generated)

```

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

```

### Event Messages

- [Keyboard.KeyEvent](#)
- [Keyboard.EnterDataEvent](#)
- [Keyboard.LayoutEvent](#)

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

---

**Keyboard.Reset**

---

**Description**

Sends a service reset to the Service Provider.

**Command Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

---

**Keyboard.SecureKeyEntry**

---

#### Description

This command allows a full length symmetric encryption key part to be entered directly into the device without being exposed outside of the device. 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 device 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 device 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 device 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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 <span style="border: 1px solid black; padding: 2px;">(Required)</span>	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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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	
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	(Required)	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  
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```

        "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
    },
    "terminatesKeys": {
        "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 <span style="border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="border: 1px solid black; padding: 2px;">(Required)</span>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
digits <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	integer		Specifies the number of key digits entered. Applications must ensure all required digits have been entered before trying to store the key.
completion <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 field it not set 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": {
    "completionCode": "success",
    "errorDescription": "string",
    "digits": 0,
    "completion": "auto",
    "kcv": "string"
  }
}
```

### Event Messages

- [Keyboard.KeyEvent](#)
- [Keyboard.EnterDataEvent](#)
- [Keyboard.LayoutEvent](#)

## Keyboard.KeypressBeep

### Description

This command is used to enable or disable the 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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #e0e0e0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

---

## Keyboard.DefineLayout

---

#### Description

This command allows an application to configure a layout for any 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 DataEntry command (3) PIN mode which corresponds to the PinEntry command (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 DataEntry, PinEntry, 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Message Type	Name	Type	Default	Description
	entryMode <small>(Required)</small>	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 DataEntry method.
	entryMode.pin	boolean	false	Specifies that the layout be applied to the PinEntry 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 <small>(Required)</small>	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 <small>(Required)</small>	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 <small>(Required)</small>	integer		For ETS, specifies the width of the frame. For all other device types, this value is ignored
	frames.ySize <small>(Required)</small>	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 <small>(Required)</small>	boolean	false	Specifies that the PIN device will randomly shift the layout in a horizontal direction
	frames.floatAction.floatY <small>(Required)</small>	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 <small>(Required)</small>	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 <small>(Required)</small>	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 <small>(Required)</small>	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 <small>(Required)</small>	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.fk	string		Specifies the FK code associated with the physical area in non-shifted mode.
	frames.fks.shiftFK	string		Specifies the FK code associated with the physical key in shifted mode.

Example Message (generated)

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
{
  "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,
            "fk": "fk0",
            "shiftFK": "fk0"
          }
        ]
      }
    ]
  }
}
```

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

##### Example Message (generated)

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

#### Event Messages

## Unsolicited Events

### Events

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

---

**Keyboard.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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload**

Name	Type	Default	Description
completion <b>(Required)</b>	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	
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	

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
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  
        }  
    }  
}
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

---

## Keyboard.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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Example Message (generated)

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

---

## Keyboard.LayoutEvent

---

### Description

This event sends the layout for a specific keyboard entry mode if the layout has changed since it was loaded (i.e. if a float action is being used).

### Message Header

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

### Message Payload

Name	Type	Default	Description
entryMode <b>(Required)</b>	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 DataEntry method.
entryMode.pin	boolean	false	Specifies that the layout be applied to the PinEntry 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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	integer		For ETS, specifies the width of the frame. For all other device types, this value is ignored
frames.ySize <b>(Required)</b>	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

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
frames.floatAction.floatX <b>(Required)</b>	boolean	false	Specifies that the PIN device will randomly shift the layout in a horizontal direction
frames.floatAction.floatY <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	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.fk	string		Specifies the FK code associated with the physical area in non-shifted mode.
frames.fks.shiftFK	string		Specifies the FK code associated with the physical key 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,
            "fk": "fk0",
            "shiftFK": "fk0"
          }
        ]
      }
    ]
  }
}
```

XFS4IoT  
All rights reserved

# XFS4IOT Sample Messaging for PinPad Draft

## 0.0.4

This section describes the general interface for the following functions:

- Administration of encryption devices
- PIN verification
- PIN block generation (encrypted PIN)
- PIN presentation to chipcard
- EMV 4.0 PIN blocks, EMV 4.0 public key loading, static and dynamic data verification

## Documentation

### 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.GetQueryPCIPTSDeviceId

---

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

**Message Payload**

Name	Type Default	Description
completionCode	string	success if the command was successful otherwise error
errorDescription	string	If not success, then this is optional vendor dependent information to provide additional information
manufacturerIdentifier	string	Returns an ASCII string containing the manufacturer identifier of the PIN device. This value is not set 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 not set if the model identifier is not available.
hardwareIdentifier	string	Returns an ASCII string containing the hardware identifier of the PIN device. This value is not set if the hardware identifier is not available.
firmwareIdentifier	string	Returns an ASCII string containing the firmware identifier of the PIN device. This value is not set if the firmware identifier is not available.
applicationIdentifier	string	Returns an ASCII string containing the application identifier of the PIN device. This value is not set if the application identifier is not available.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "manufacturerIdentifier": "string",
    "modelIdentifier": "string",
    "hardwareIdentifier": "string",
    "firmwareIdentifier": "string",
    "applicationIdentifier": "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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	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 this field is not set then no offset is used. The character must be in the ranges '0' to '9', 'a' to 'f' and 'A' to 'F'.
padding <b>(Required)</b>	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 <b>(Required)</b>	integer		Maximum number of PIN digits to be used for validation. This parameter corresponds to PINMINL in the IBM 3624 specification
valDigits <b>(Required)</b>	integer		Number of Validation digits from the validation data to be used for validation. This is the length of the validationData string.
noLeadingZero <b>(Required)</b>	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.

Name	Type	Default	Description
key <b>(Required)</b>	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 this field is not set, 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 <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
result <b>(Required)</b>	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": {  
    "completionCode": "success",  
    "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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
pan <b>(Required)</b>	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 <b>(Required)</b>	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 <b>(Required)</b>	integer		Number of digits of PVV.
key <b>(Required)</b>	string		Name of the validation key. The key referenced by key must have the function or pinLocal attribute
keyEncKey	string		If this field is not set, 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
result <b>(Required)</b>	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": {
    "completionCode": "success",
    "errorDescription": "string",
    "result": true
  }
}
```

### 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

Name	Type	Default	Description
name <b>(Required)</b>	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.
presentAlgorithm <b>(Required)</b>	string		Specifies the algorithm that is used for presentation. Possible values are: (see command Capabilites).
chipProtocol <b>(Required)</b>	string		Identifies the protocol that is used to communicate with the chip. Possible values are: (see command CardReader.Capabilities in the Identification Card Device Class Interface)
chipData <b>(Required)</b>	string		Points to the data to be sent to the chip formatted in base64.
algorithmData <b>(Required)</b>	object		Contains the data required for the specified presentation algorithm
algorithmData.pinPointer <b>(Required)</b>	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 <b>(Required)</b>	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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
chipProtocol <b>(Required)</b>	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 <b>(Required)</b>	string		The data responded from the chip.

### Example Message (generated)

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

## Event Messages

---

### Pinpad.Reset

---

#### Description

Sends a service reset to the Service Provider.

#### Command Message

---

### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "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	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		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
maintainPIN	boolean	false <b>(Required)</b>	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	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "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. Keyboard.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 Keyboard.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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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
customerData <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	string		<p>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 DIELBOLD 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).</p>
xorData <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	string		<p>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 IpsKeyEncKey is NULL) this parameter is ignored.</p>
padding <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	integer		<p>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.</p>
format <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	string		<p>Specifies the format of the PIN block. Possible values are: (see command Capabilities)</p>
key	string		<p>Specifies the key used to encrypt the formatted PIN for the first time, this field is not required 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 IpsKey must have the WFS_PIN_USEFUNCTION or UserPinRemote attribute. If this specifies an RSA key, RSA encryption will be performed</p>

Name	Type	Default	Description
secondEncKey	string		Specifies the key used to format the once encrypted formatted PIN, this field is not required if no second encryption required. The key referenced by IpsKeyEncKey must have the UseFunction or UsePinRemote attribute. If this specifies a double-length or triple-length key, triple DES encryption will be performed.
pinBlockAttributes <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>			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 <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the key usages supported by the PINBLOCK command.
pinBlockAttributes.algorithm <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the encryption algorithms supported by the PINBLOCK command as one of the following values
pinBlockAttributes.modeOfUse <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	string		Specifies the encryption modes supported by the PINBLOCK command as one of the following values
pinBlockAttributes.cryptoMethod <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	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": "o",
      "cryptoMethod": "ecb"
    }
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## 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 <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

##### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
customerData <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	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 DIELBOLD 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 <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	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, this field is not required 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

Name	Type	Default	Description
secondEncKey	string		Specifies the key used to format the once encrypted formatted PIN, this field is not required if no second encryption required. The key referenced by IpsKeyEncKey must have the UseFunction or UsePinRemote attribute. If this specifies a double-length or triple-length key, triple DES encryption will be performed.
pinBlockAttributes <span style="background-color: #cccccc; padding: 2px;">(Required)</span>			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 <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	string		Specifies the key usages supported by the PINBLOCK command.
pinBlockAttributes.algorithm <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	string		Specifies the encryption algorithms supported by the PINBLOCK command as one of the following values
pinBlockAttributes.modeOfUse <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	string		Specifies the encryption modes supported by the PINBLOCK command as one of the following values
pinBlockAttributes.cryptoMethod <span style="background-color: #cccccc; padding: 2px;">(Required)</span>	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"
    }
  }
}
```

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
pinBlock <b>(Required)</b>	string		The encrypted PIN block formatted in base64

### Example Message (generated)

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

### Event Messages

- [Pinpad.DUKPTKSNEvent](#)

---

## Unsolicited Events

### Pinpad.IllegalKeyAccessEvent

---

#### Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
keyName <b>(Required)</b>	string		Specifies the name of the key that caused the error.
errorCode <b>(Required)</b>	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"
  }
}
```

---

## Events

### **Pinpad.DUKPTKSNEvent**

---

**Description**

This event sends the DUKPT KSN of the key used in the command. The receiving TRSM uses this to derive the key from the BDK.

**Message Header**

Name	Type	Default	Description

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

### Message Payload

Name	Type	Default	Description
key <b>(Required)</b>	string		Specifies the name of the DUKPT Key derivation key.
ksn <b>(Required)</b>	string		structure that contains the KSN formatted in base64.

### Example Message (generated)

XFS4  
risk  
All rig

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

## XFS4IOT Sample Messaging for Printer Draft 0.0.1

This specification describes the functionality of the services provided by banking printers and scanning devices under XFS, focusing on the following areas:

- application programming for printing
- print document definition
- scanning images for devices such as check scanners

The XFS printer service class is implemented around a forms model which also standardizes the basic document definition.

## Documentation

### Banking Printer Types

---

The XFS printer service defines and supports five types of banking printers through a common interface:

- **Receipt Printer**

The receipt printer is used to print cut sheet documents. It may or may not require insert or eject operations, and often includes an operator identification device, e.g. Teller A and Teller B lights, for shared operation.

- **Journal Printer**

The journal is a continuous form device used to record a hardcopy audit trail of transactions, and for certain report printing requirements.

- **Passbook Printer**

The passbook device is physically and functionally the most complex printer. The XFS definition supports automatic positioning of the book, as well as read/write capability for an optional integrated magnetic stripe. The implementation also manages the book geometry - i.e. the margins and centerfolds - presenting the simplest possible application interface while delivering the full range of functionality.

Some passbook devices also support the dispensing of new passbooks from up to four passbook paper sources (upper, aux, aux2, lower). Some passbook devices may also be able to place a full passbook in a parking station, print the new passbook and return both to the customer. Passbooks can only be dispensed or moved from the parking station if there is no other media in the print position or in the entry/exit slot.

- **Document Printer**

Document printing is similar to receipt printing - a set of fields are positioned on one or more inserted sheets of paper - but the focus is on full-size forms. It should be noted that the XFS environment supports the printing of text and graphic fields from the application. The electronic printing of the form image (the template portion of the form which is usually pre-printed with dot-matrix style printers) may also be printed by the application.

- **Scanner Printer**

The scanner printer is a device incorporating both the capabilities to scan inserted documents and optionally to print on them. These devices may have more than one area where documents may be retained.

Additional hardware components, like scanners, stripe readers, OCR readers, and stamps, normally attached directly to the printer are also controlled through this interface. Additionally, the Printer and Scanning class interface can also be used for devices that are capable of scanning without necessarily printing. This includes devices such as Check Scanners.

The specification refers to the terms paper and media. When the term paper is used this refers to paper that is situated in a paper supply attached to the device. The term media is used for media that is inserted by the customer (e.g. check and other material that is scanned) or that is issued to the customer (e.g. a receipt or

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
statement). Receipt, document and passbook printers with white passbook dispensing capability have both. As soon as the paper gets printed it becomes media. Scanners only have media. The term media does not apply to journal printers. When paper is in the print position it is classified as media, on some printers that maintain paper under the print head there will always be both media and paper.

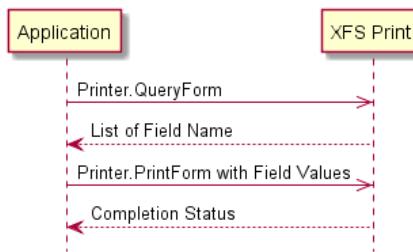
## Forms Model

---

The XFS printing class functionality is based on a “forms” model for printing. Banking documents are represented as a series of text and/or graphic fields output from the application and positioned on the document by the XFS printing system.

The form is an object which includes the positioning and presentation information for each of the fields in the document. The application selects a form and supplies only the field data and the control parameters to fully define the print document.

The form objects are owned and managed by the XFS printing service. To optimize maintainability of the system, the application can query the service for the list of fields required to print a given form. Through this mechanism, it is not necessary to duplicate the field contents of forms in application authoring data. The figure below outlines the printing process from the application's view.



The XFS implementation recognizes that the form object must be supported by job-specific data to fully address printing requirements. As an example, a form defining a passbook print line will need to have its origin defined externally in order to be reused for different passbook lines. These job specific parameters are supplied on the [Printer.PrintForm](#) command.

In some cases, the application wants to print a block of data without considering it as a series of separate fields. One example is a line of journal data, fully formatted by the application. This can be handled by defining a one field form, or by use of the [Printer.PrintRawData](#) command.

The document definition under XFS printing is standardized to provide portability across vendor implementations. The standard has been defined at the source language level for the document definition, allowing vendor differences at the runtime level to manage implementation specific dependencies, providing several areas where vendors can provide value-added extensions. As an example, a vendor providing a graphical form definition tool can produce the field definition object format directly. The XFS requirements for portability are:

- A vendor must be able to export print format in the standardized field definition source format for portability to other systems.
- A vendor must be able to import document formats produced on other systems in the standardized field definition source format.
- A vendor can extend the field definition source language, but any verbs included in the standard must be implemented strictly as defined by the standard. Import and export facilities must be tolerant of source language extensions, reporting but ignoring the exceptions.

## Command Overview

---

The basic operation of the print devices is managed using the two primary commands:

- [Printer.QueryForm](#)  
This command retrieves the form header information, and the list of fields.
- [Printer.PrintForm](#)  
This command includes as parameter data the name of the form to select and the required field data values.

## XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

This approach combines in the most efficient manner the four logical steps required to print a form:

- Selecting a document form object.
- Querying the service for the list of fields.
- Supplying the data for each field.
- Issuing the print command.

By using [Printer.QueryForm](#) command to retrieve the list of field names, it is possible for an application to assemble the required set of fields for a form before locking the service. This minimizes the time that each application request ties up the service. Using [Printer.QueryForm](#), it is also possible to query the attributes of a field. This command is generally not required for most applications.

The combination of form selection, field value presentation and the print action make it possible to express a complete print operation with [Printer.PrintForm](#) command. Where these multiple print functions represent a series of passbook lines (using the INDEX capability in the field definition), the [Printer.PrintForm](#) command provides support for management of the print line number. Note that if a form contains a tabular field (i.e. one with a non-zero INDEX value), and data is not supplied for some of the lines in the "table", then those lines are left blank.

For printers with the capability to read from a passbook (OCR, MICR and/or magnetic stripe), the data is read with the [Printer.ReadForm](#) command. The data is written using the [Printer.PrintForm](#) command. Since these devices are usable only for passbook operations, they are not defined as separate logical devices.

Finally, the [Printer.PrintRaw](#) command can be used to print a file that contains a complete print job in the native printer language. This file will have been created using the native Operating System API (e.g. Windows GDI).

## Form, Sub-Form, Field, Frame, Table and Media Definitions

---

This section outlines the format of the definitions of forms, the fields within them, optional tables and fields within the form, and the media on which they are printed.

### Definition Syntax

---

The syntactic rules for form, field and media definitions are as follows:

- **White space**  
space, tab
- **Line continuation**  
backslash (\)
- **Line termination**  
CR, LF, CR/LF; line termination ends a "keyword section" (a keyword and its value[s])
- **Keywords**  
must be all upper case
- **Names**  
(field/media/font names) any case; case is preserved;

Service Providers are case sensitive

- **Strings**  
all strings must be enclosed in double quote characters (");  
standard C escape sequences are allowed.
- **Comments**  
start with two forward slashes (//), end at line termination

Other Notes:

- The values of a keyword are separated by commas.
- If a keyword is present, all its values must be specified; default values are used only if the keyword is absent.
- Values that are character strings are marked with \* in the definitions below and must be quoted as specified above.
- The order of attributes within the forms is not mandatory and the attributes may be defined in any order.
- All forms can be represented using either ISO 646 (ANSI) or UNICODE character encoding. If the UNICODE representation is used, then all Names and Strings are restricted to an internal representation of ISO 646 (ANSI) characters. Only the INITIALVALUE and FORMAT keyword values can have double byte values

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
outside of the ISO 646 (ANSI) character set.

- If forms character encoding is UNICODE then, consistent with the UNICODE standard, the file prefix must be in LittleEndian (0xFFFFE) or BigEndian (0xFEFF) notation, such that UNICODE encoding is recognized.
- A form and its optional subforms that have multiple XFSFIELDS with the same fieldname are invalid. The *formInvalid* error will be returned if specified in the input to the command.
- A form that has multiple XFSSUBFORMS with the same subformname is invalid. The *formInvalid* error will be returned if specified in the input to the command.
- A form and its optional subforms that have multiple XFSFRAMES with the same framename are invalid. The *formInvalid* error will be returned if specified in the input to the command.

## Form and Media Measurements

---

The UNIT keyword sections of the form and media definitions specify the base horizontal and vertical resolution as follows:

- The *base* value specifies the base unit of measurement.
- The *x* and *y* values specify the horizontal and vertical resolution as fractions of the base value (e.g. an *x* value of 10 and a base value of MM means that the base horizontal resolution is 0.1 mm).

The base resolutions thus defined by the UNIT keyword section of the XFSFORM definition are used as the units of the form definition keyword sections:

- SIZE (*width* and *height* values)
- ALIGNMENT (*xoffset* and *yoffset* values)

and of the sub-form definition keyword sections:

- POSITION (*x* and *y* values)
- SIZE (*width* and *height* values)

and of the field definition keyword sections:

- POSITION (*x* and *y* values)
- SIZE (*width* and *height* values)
- INDEX (*xoffset* and *yoffset* values)

and of the frame definition keyword sections:

- POSITION (*x* and *y* values)
- SIZE (*width* and *height* values)
- REPEATONX (*xoffset* value)
- REPEATONY (*yoffset* value)

The base resolutions thus defined by the UNIT keyword section of the XFSMEDIA definition are used as the units of the media definition keyword sections:

- SIZE (*width* and *height* values)
- PRINTAREA (*x*, *y*, *width* and *height* values)
- RESTRICTED (*x*, *y*, *width* and *height* values)

NOTE: The origin for coordinate based systems is (0,0). The origin for row/column based systems can be (0,0) or (1,1) and must be configurable within the Service Provider.

## XFSFORM

### *form* Form Definition

---

#### BEGIN

(~~required~~) Attributes are not required in any mandatory order within a Form definition.

#### UNIT XFSFORM

*base*,

Base resolution unit for form definition:

MM

INCH

ROWCOLUMN

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

*x,*

Horizontal base unit fraction

*y*

Vertical base unit fraction

**(required)**

#### **SIZE**

*width,*

Width of form

*height*

Height of form

**(required)**

#### **ALIGNMENT**

*alignment,*

Alignment of the form on the physical media (TOPLEFT (default), TOPRIGHT, BOTTOMLEFT, BOTTOMRIGHT). This option allows the positioning of a form onto a physical page relative to any combination of the edges of the physical media, to support the variations in how devices sense the edge of page for positioning purposes.

*xoffset,*

Horizontal offset relative to the horizontal alignment specified by alignment. Always specified as a positive value (i.e. if aligned to the right side of the media, means offset the form to the left). (default = 0)

*yoffset*

Vertical offset relative to the vertical alignment specified by alignment. Always specified as a positive value (i.e. if aligned to the bottom of the media, means offset the form upward). (default = 0)

#### **ORIENTATION**

*type*

Orientation of form:

PORTRAIT (default)

LANDSCAPE

#### **SKEW**

*skewfactor*

Maximum skew factor in degrees (default = 0)

#### **VERSION**

*major,*

Major version number

*minor,*

Minor version number

*date\**,

Creation/modification date

*author\**

Author of the form

#### **CPI**

*cpi*

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
Characters per inch. This value specifies the default CPI within the form. When the ROWCOLUMN unit is used, the form CPI and LPI are used to calculate the position and size of all fields within a form, irrespective of the CPI and LPI of the fields themselves.

**LPI**

*lpi*

Lines per inch. This value specifies the default LPI within the form. When the ROWCOLUMN unit is used, the form CPI and LPI are used to calculate the position and size of all fields within a form, irrespective of the CPI and LPI of the fields themselves.

**POINTSIZE**

*pointsize*

This value specifies the default POINTSIZE within the form.

**COPYRIGHT**

*copyright\**

Copyright entry

**TITLE**

*title\**

Title of form

**COMMENT**

*comment\**

Comment section

**USERPROMPT**

*prompt\**

Prompt string for user interaction

**[XFSFIELD**

**BEGIN**

...

**END]**

*fieldname\**

One field definition (as defined in the next section) for each field in the form. The fieldname within a form and its optional subforms must be unique.

**[XFSFRAME**

**BEGIN**

...

**END]**

*framename\**

One frame definition (as defined in the next section) for each frame in the form. The framename within a form and its optional subforms must be unique.

**[XFSSUBFORM**

**BEGIN**

...

**END]**

*subformname\**

One subform definition (as defined in the next section) for each subform in the form. The subformname within a form must be unique.

**END**

**XFSSUBFORM**

**XFSSUBFORM**

*subformname*\*

**BEGIN**

**(required)**

**POSITION**

*x*,

Horizontal position (relative to left side of form)

*y* or *(y, z)*

Vertical position (relative to top of form). Format *(y, z)* is used to indicate vertical positioning relative to top of form when top of form is other than 1st page of form, where Z indicates page number (relative to 0) and Y indicates base resolution units relative to top of the form page number (as indicated by Z). Format *y* is used to indicate vertical positioning relative to top of the 1st form page.

**(required)**

**SIZE**

*width*,

Width of subform. Width must not exceed width of form.

*height*

Height of subform. Height must not exceed height of form.

**[XFSFIELD**

**BEGIN**

...

**END]**

*fieldname*\*

One field definition (as defined in the next section) for each field in the form. The *fieldname* within a form and its optional subforms must be unique.

**[XFSFRAME**

**BEGIN**

...

**END]**

*framename*\*

One frame definition (as defined in the next section) for each frame in the form. The *framename* within a form and its optional subforms must be unique.

**END**

The XFSSUBFORM definition provides a means to isolate a selected area of a form where the user may want to have a select group of fields, frames, and/or running headers and footers. All field and frame definitions within a subform are relative to the POSITION of the subform. A form definition with an imbedded subform will have a series of statements illustrated as follows:

```
XFSFORM
BEGIN
*
*
XFSSUBFORM
BEGIN
    XFSFIELD
    BEGIN
```

```
*  
*  
END  
XFSFIELD  
BEGIN  
*  
*  
END  
END  
END
```

## Field Definition

---

### **XFSFIELD**

#### **XFSFIELD**

*fieldname*\*

The *fieldname* within a form and its optional subforms must be unique.

#### **BEGIN**

##### **(required)**

#### **POSITION**

x,

Horizontal position (relative to left side of form/subform).

y or (y, z)

Vertical position (relative to top of form/subform). Format (y, z) is used to indicate vertical positioning relative to top of form/subform when top of form/subform is other than 1st page of form/subform, where z indicates page number (relative to 0) and y indicates base resolution units relative to top of the form/subform page number (as indicated by z). Format y is used to indicate vertical positioning relative to top of the 1st form/subform.

#### **FOLLOWS**

*fieldname*\*

Print this field directly following the field with the name *fieldname*; positioning information is ignored. See the description of [Printer.PrintForm](#). If **FOLLOWS** is omitted, then fields are printed in the order that they appear in the form definition.

#### **HEADER**

N

This field is either a form/subform header field. N represents a form/subform page number (relative to 0) the header field is to print within.

N-N

N-N represents a form/subform page number range the header field is to print within. Combinations of N and N-N may exist separated by commas.

ALL

ALL indicates that header field is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example, 0,2-4,6 indicates that the header field is to print on relative form/subform pages 0, 2, 3, 4, and 6.

#### **HEADER**

N

This field is either a form/subform header field. N represents a form/subform page number (relative to 0) the header field is to print within.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
**N-N**

N-N represents a form/subform page number range the header field is to print within. Combinations of N and N-N may exist separated by commas.

**ALL**

ALL indicates that header field is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example, 0,2-4,6 indicates that the header field is to print on relative form/subform pages 0, 2, 3, 4, and 6.

**FOOTER**

**N**

This field is either a form/subform footer field. N represents a form/subform page number (relative to 0) the footer field is to print within.

**N-N**

N-N represents a form/subform page number range the footer field is to print within. Combinations of N and N-N may exist separated by commas.

**ALL**

ALL indicates that footer field is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example, 0,2-4,6 indicates that the footer field is to print on relative form/subform pages 0, 2, 3, 4, and 6.

**SIDE**

*side*

Side of form where field is positioned:

FRONT (default)

BACK

**(required)**

**SIZE**

*width*,

Field width.

*height*

Field height.

**INDEX**

*repeatcount*

Count how often this field is repeated in the form, INDEX fields are fixed length (default is no INDEX field).

*xoffset*,

Horizontal offset for next field.

*yoffset*

Vertical offset for next field.

**TYPE**

*fieldtype*

Type of field:

TEXT (default)

MICR

OCR

MSF

BARCODE

GRAPHIC

## **SCALING**

*scalingtype*

Information on how to size the GRAPHIC within GRAPHIC field types:

BESTFIT (default): Scale to size indicated.

ASIS: Render at native size.

MAINTAINASPECT: scale as close as possible to size indicated while maintaining the aspect ratio and not losing graphic information.

## **BARCODE**

*hriposition*

Position of the HRI (Human Readable Interpretation) characters:

NONE (default)

ABOVE

BELLOW

BOTH

The type of barcode to print is defined in the FONT field.

## **COERCIVITY**

*coercivity*

Coercivity to be used for writing to the magnetic stripe of MSF field types:

AUTO (default): Coercivity is decided by the Service Provider or hardware.

LOW: Low coercivity.

HIGH: High coercivity.

## **CLASS**

*class*

Field class:

OPTIONAL (default)

STATIC

REQUIRED

## **ACCESS**

*access*

Access rights of field:

WRITE (default)

READ

READWRITE

## **OVERFLOW**

*overflow*

Action of field overflow:

TERMINATE (default)

TRUNCATE

BESTFIT (The Service Provider fits the data into the field as well as it can)

OVERWRITE (a contiguous write)

WORDWRAP

## **STYLE**

*style*

Display attributes as a combination, using the | operator, of the following:

NORMAL (default)

BOLD

ITALIC

UNDER (single underline)

DOUBLEUNDER (double underline)

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

DOUBLE (double width)  
TRIPLE (triple width)  
QUADRUPLE (quadruple width)  
STRIKETHROUGH  
ROTATE90 (rotate 90 degrees clockwise)  
ROTATE270 (rotate 270 degrees clockwise)  
UPSIDEDOWN (upside down)  
PROPORTIONAL (proportional spacing)  
DOUBLEHIGH  
TRIPLEHIGH  
QUADRUPELHIGH  
CONDENSED  
SUPERSCRIPT  
OVERSCORE  
LETTERQUALITY  
NEARLETTERQUALITY  
DOUBLESTRIKE  
OPAQUE (If omitted then the default attribute is transparent)

Some of these styles may be mutually exclusive or may combine to provide unexpected results.

#### **CASE**

*case*

Convert field contents to:

NOCHANGE (default)  
UPPER  
LOWER

#### **HORIZONTAL**

*justify*

Horizontal alignment of field contents:

LEFT (default)  
RIGHT  
CENTER  
JUSTIFY

#### **VERTICAL**

*justify*

Vertical alignment of field contents:

BOTTOM (default)  
CENTER  
TOP

#### **COLOR**

*color*

Color name:

BLACK (default)  
WHITE  
GRAY  
RED  
BLUE  
GREEN  
YELLOW

#### **RGBCOLOR**

*R, G, B*

Color in RGB 8 bits per color format:

R: Red portion of the RGB value 0-255.  
G: Green portion of the RGB value 0-255.  
B: Blue portion of the RGB value 0-255.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

RGBCOLOR overrides the COLOR attribute.

**FONT**

*fontname*\*

Font name: This attribute is interpreted by the Service Provider. In some cases, it may indicate printer resident fonts, and in others it may indicate the name of a downloadable font. For BARCODE fields it represents the barcode font name. In some cases, this pre-defines the following parameters:

**POINTSIZE**

*pointsize*

Point size. If unspecified, the point size defaults to the POINTSIZE defined for the form.

**CPI**

*cpi*

Characters per inch. If unspecified, the CPI defaults to the CPI defined for the form.

**LPI**

*lpi*

Lines per inch. If unspecified, the LPI defaults to the LPI defined for the form.

**FORMAT**

*formatstring*\*

This is an application defined input field describing how the application should format the data. This may be interpreted by the Service Provider.

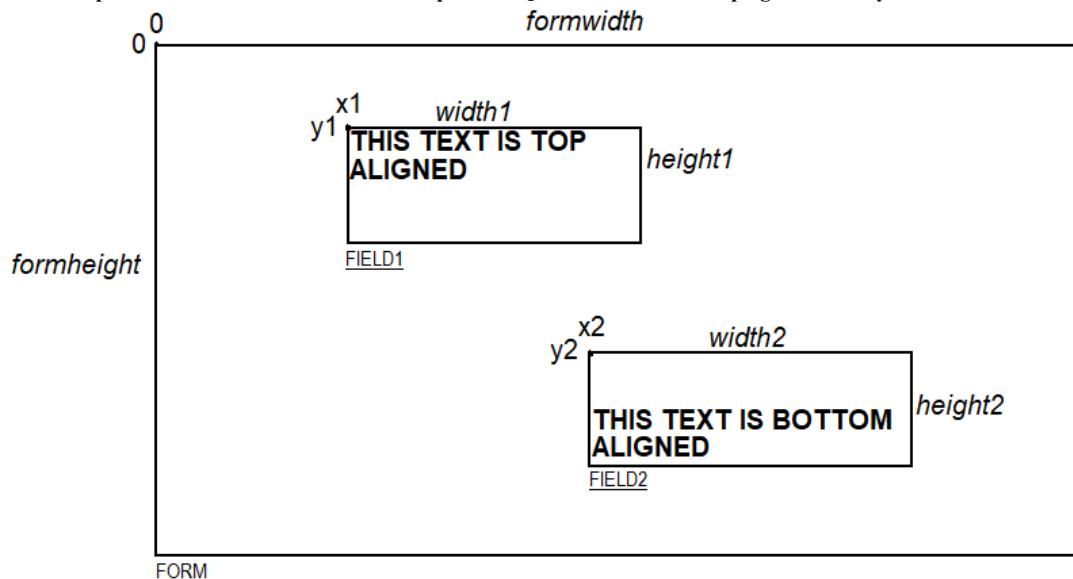
**INITIALVALUE**

*value*\*

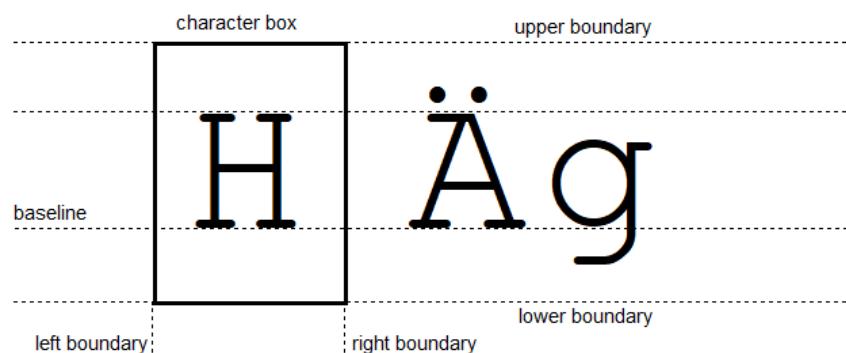
Initial value. For GRAPHIC type fields, this value may contain the filename of the graphic image. The type of this graphic will be determined by the file extension (e.g. BMP for Windows Bitmap). Graphic file name may be full or partial path. For example, "C:\BSVC\BSVCLOGO.BMP" illustrates use of full path name. A file name specification of "LOGO.BMP" illustrates partial path name. In this instance file is obtained from current directory. Graphic contents can be changed dynamically at run-time and the new content will be printed on the next print action.

**END**

The following diagrams illustrate the positioning and sizing of text fields on a form, and the vertical alignment of text within a field using **VERTICAL=TOP** and **VERTICAL=BOTTOM** values in the field definition.



Definition of the character drawing box:



When more than one line of text is to be printed in a field, and the definition includes **VERTICAL=BOTTOM**, the vertical position of the first line is calculated using the specified (or implied) **LPI** value.

## Frame Definition

---

**XFSFRAME**

**XFSGRAME**

*framename\**

**BEGIN**

**(required)**

**POSITION**

*X,*

Horizontal position of top left corner of the frame (relative to left side of form/subform).

*Y or (Y, Z)*

Vertical position of top left corner of frame (relative to top of form/subform). Format (Y, Z) is used to indicate vertical positioning of the top left corner of the frame relative to top of form/subform when top of form/subform is other than 1st page of form/subform, where Z indicates page number (relative to 0) and Y indicates base resolution units relative to top of the form/subform page number (as indicated by Z). Format Y is used to indicate vertical positioning of the left corner of frame

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
relative to top of the 1st form/subform.

## FRAMES

*fieldname*\*

Frames the field with the name *fieldname*, positioning and size information are ignored. The frame surrounds the complete field, not just the printed data. If the field is repeated, the frame surrounds the first and last fields that are printed.

## HEADER

*N*

This frame is either a form/subform header frame. *N* represents a form/subform page number (relative to 0) the header frame is to print within.

*N-N*

*N-N* represents a form/subform page number range the header frame is to print within. Combinations of *N* and *N-N* may exist separated by commas.

*ALL*

*ALL* indicates that header frame is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example, 0,2-4,6 indicates that the header frame is to print on relative form/subform pages 0, 2, 3, 4, and 6.

## FOOTER

*N*

This frame is either a form/subform footer frame. *N* represents a form/subform page number (relative to 0) the footer frame is to print within.

*N-N*

*N-N* represents a form/subform page number range the footer frame is to print within. Combinations of *N* and *N-N* may exist separated by commas.

*ALL*

*ALL* indicates that footer frame is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example, 0,2-4,6 indicates that the footer frame is to print on relative form/subform pages 0, 2, 3, 4, and 6.

## SIDE

*side*

Side of form where this frame is positioned:

FRONT (default)

BACK

**(required)**

## SIZE

*width,*

Frame width in base horizontal units for the form.

*height*

Frame height in base vertical units for the form.

## REPEATONX

*repeatcount,*

Count how often this frame is repeated horizontally in the form.

*xoffset*

Horizontal offset for next frame in base horizontal units.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
**REPEATONY**

*repeatcount*,

Count how often this frame is repeated vertically in the form.

*yoffset*

Vertical offset for next frame in base vertical units.

#### **TYPE**

*frametype*

Type of frame:

RECTANGLE (default)

ROUNDED\_CORNER

ELLIPSE

#### **CLASS**

*class*

Frame class:

STATIC (default)

OPTIONAL (The frame is printed only if its name appears in the list of field names given as parameter to the [Printer.PrintForm](#) command. In this case, the name of the frame must be different from all the names of the fields.)

#### **OVERFLOW**

*overflow*

Action on frame overflowing the form:

TERMINATE (default)

TRUNCATE

BESTFIT (the Service Provider fits the frame into the media as well as it can)

#### **STYLE**

*style*

Frame line attributes:

SINGLE\_THIN (default)

DOUBLE\_THIN

SINGLE\_THICK

DOUBLE\_THICK

DOTTED

#### **COLOR**

*color*

Color name for frame lines:

BLACK (default)

WHITE

GRAY

RED

BLUE

GREEN

YELLOW

#### **RGBCOLOR**

*R, G, B*

Color in RGB 8 bits per color format:

R: Red portion of the RGB value 0-255.

G: Green portion of the RGB value 0-255.

B: Blue portion of the RGB value 0-255.

RGBCOLOR overrides the COLOR attribute.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
**FILLCOLOR**

*color*

Color name for interior of frame:

BLACK  
WHITE (default)  
GRAY  
RED  
BLUE  
GREEN  
YELLOW

**RGBFILLCOLOR**

*R, G, B*

Color in RGB 8 bits per color format:

R: Red portion of the RGB value 0-255.  
G: Green portion of the RGB value 0-255.  
B: Blue portion of the RGB value 0-255.

RGBFILLCOLOR overrides the FILLCOLOR attribute.

**FILLSTYLE**

*style*

Style for filling the interior of frame:

NONE (default): No fill  
SOLID: Solid color  
BDIAGONAL: Downward hatch (left to right) at 45 degrees  
CROSS: Horizontal and vertical crosshatch  
DIAGCROSS: Crosshatch at 45 degrees  
FDIAGONAL: Upward hatch (left to right) at 45 degrees  
HORIZONTAL: Horizontal hatch  
VERTICAL: Vertical hatch

**SUBSTSIGN**

*substitute sign*

Character that is used as substitute sign when a character in a read field cannot be read

**TITLE**

*fieldname\**

Uses the field with the name as the title of the frame. Positioning information of the field is ignored.

**HORIZONTAL**

*justify*

Horizontal alignment of the frame title:

LEFT (default)  
CENTER  
RIGHT

**VERTICAL**

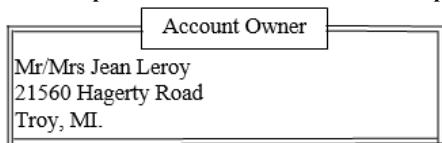
*justify*

Vertical alignment of the frame title:

TOP (default)  
BOTTOM

**END**

The **XFSFRAME** definition provides a means for framing a **XFSFIELD** text field. The basic concept of a **XFSFRAME** definition and corresponding **XFSFIELD** definition is illustrated as follows:



When the **XFSFRAME** frames a field, its positioning and size information are ignored. Instead, Service Providers should position the top left corner of the frame one horizontal base unit to the left and one vertical base unit to the top of the top left corner of the field. Similarly, Service Providers should size the frame so that its bottom right corner is one base unit below and to the right of the field. For instance, if the form units are **ROWCOLUMN**, and a **XFSFRAME** "A" is said to frame the **XFSFIELD** "B" which is positioned at row 1, column 1 with a size of 1 row and 20 columns, the frame will be drawn from row 0, column 0 to row 3, column 22.

The horizontal and vertical positioning of a frame title overrides the position of the named **XFSFIELD**. For instance, if a **XFSFRAME** "A" is said to have the **XFSFIELD** "B" as its title, with the default horizontal and vertical title justification, it is just as if **XFSFIELD** "B" had been positioned at the top left corner of the frame. Note that the **SIZE** information for the title field still is meaningful; it gives the starting and/or ending positions of the frame lines.

The **SIDE** attributes of the **XFSFRAME** and the **XFSFIELDS** it refers to must agree.

The width of the lines and the interval between the lines of doubled frames are vendor specific. Whether the lines are drawn using graphics printing or using pseudo-graphic is vendor specific. However, Service Providers are responsible for rendering intersecting frames.

Depending on the printer technology, framing of fields can substantially slow down the print process.

Support of framing by a Service Provider or the device it controls is not mandatory to be XFS compliant.

### Sample 1 - Simple Framing

---

The form:

```
XFSFORM "Multiple Balances"
BEGIN
UNIT INCH, 16, 16
SIZE 91, 64
VERSION 1, 0, "13/09/96", "XFS"
LANGUAGE 0x0409
XFSFIELD "Account Title"
BEGIN
    POSITION 15, 4
    SIZE 30, 4
    CLASS STATIC
    HORIZONTAL CENTER
    INITIALVALUE "Account"
END
XFSFIELD "Balance Title"
BEGIN
    POSITION 45, 4
    SIZE 30, 4
    CLASS STATIC
    HORIZONTAL CENTER
    INITIALVALUE "Balance"
END

XFSFIELD "Account"
BEGIN
    POSITION 15, 8
    SIZE 30, 4
    INDEX 10, 0, 3
END //Account"
XFSFIELD "Balance"
BEGIN
    POSITION 45, 8
```

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
SIZE 30, 4
INDEX 10, 0, 3
HORIZONTAL RIGHT
END //"Balance"
XFSFRAME "Account Title"
BEGIN
    POSITION 15, 4
    FRAMES "Account Title"
    SIZE 30, 4
    STYLE DOUBLE_THIN
END
XFSFRAME "Balance Title"
BEGIN
    POSITION 45, 4
    FRAMES "Balance Title"
    SIZE 30, 4
    STYLE DOUBLE_THIN
END
XFSFRAME "Account"
BEGIN
    POSITION 15, 8
    FRAMES "Account"
    SIZE 30, 34
    STYLE DOUBLE_THIN
END
XFSFRAME "Balance"
BEGIN
    POSITION 45, 8
    FRAMES "Balance"
    SIZE 30, 34
    STYLE DOUBLE_THIN
END
END
```

When printed with the following field list:

```
Account[0]=0123456789123001
Account[1]=0123456789123002
Account[2]=0123456789123003
Balance[0]=$17465.12
Balance[1]=$2458.23
Balance[2]=$6542.78
```

Will print:

Account	Balance
012345678912300	\$17465.12
1	
012345678912300	\$2458.23
2	
012345678912300	\$6542.78
3	

**Sample 2 - Framing With Title**

---

The form:

```
XFSFORM "Bank Details"
BEGIN
UNIT INCH, 16, 16
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
SIZE 121, 64
VERSION 1, 0, "13/09/96", "XFS Editor"
LANGUAGE 0x0409
XFSFIELD "Owner Frame Title"
BEGIN
    POSITION 24, 9

    SIZE 27, 3
    CLASS STATIC
    HORIZONTAL CENTER
    VERTICAL CENTER

    INITIALVALUE "Account Owner"
END
XFSFIELD "Owner"
BEGIN
    POSITION 20, 11
    SIZE 35, 9
    CLASS REQUIRED
    VERTICAL TOP
END // "Owner"
XFSFRAME "Owner Frame"
BEGIN
    POSITION 19, 10
    FRAMES "Owner"
    SIZE 37, 11
    TITLE "Owner Frame Title"
    HORIZONTAL CENTER
END
END
```

When printed with the following field list:

```
Owner = Mr./Mrs. Jean Leroy
      21560 Hagerty Road
      Troy, MI.
```

Will print:

____	Account Owner	____
<p>Mr./Mrs. Jean Leroy 21560 Hagerty Road Troy, MI.</p>		

### Sample 3 - Framing With Filled Interior

---

The form:

```
XFSFORM "Bank Details"
BEGIN
UNIT INCH, 16, 16
SIZE 121, 64
VERSION 1, 0, "13/09/96", "XFS Editor"
LANGUAGE 0x0409
XFSFIELD "Owner"
BEGIN
    POSITION 20, 11
    SIZE 35, 9
    CLASS REQUIRED
```

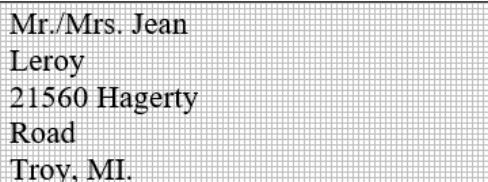
**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

```
    VERTICAL TOP
END
XFSFRAME "Owner Frame"
BEGIN
    POSITION 19, 10
    FRAMES "Owner"
    SIZE 37, 11
    FILLCOLOR GRAY
    FILLSTYLE CROSS
END
END
```

When printed with the following field list:

```
Owner = Mr./Mrs. Jean Leroy
        21560 Hagerty Road
        Troy, MI.
```

Will print:



```
Mr./Mrs. Jean
Leroy
21560 Hagerty
Road
Troy, MI.
```

---

**Sample 4 - Repeated Framing**

---

The form:

```
XFSFORM "Smart Account Number"
BEGIN
UNIT INCH, 16, 16
SIZE 121, 64
VERSION 1, 0, "13/09/96", "XFS Editor"
LANGUAGE 0x0409
XFSFIELD "Account Number"
BEGIN
    POSITION 20, 8
    SIZE 4, 4
    INDEX 12, 4, 0
    HORIZONTAL CENTER
    VERTICAL CENTER
END
XFSFRAME "A/N Frame"
BEGIN
    POSITION 20, 8
    SIZE 4, 4
    REPEATONX 12, 4
    END
END
```

When printed with the following field list:

```
Account Number[0]=0
Account Number[1]=1
Account Number[2]=2
Account Number[3]=3
Account Number[4]=4
Account Number[5]=5
Account Number[6]=6
```

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Account Number[7]=7  
Account Number[8]=8  
Account Number[9]=9  
Account Number[10]=0  
Account Number[11]=1

Will print:

0	1	2	3	4	5	6	7	8	9	0	1
---	---	---	---	---	---	---	---	---	---	---	---

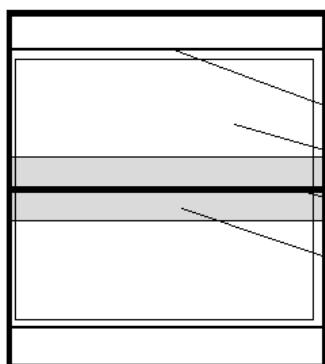
## Media Definition

---

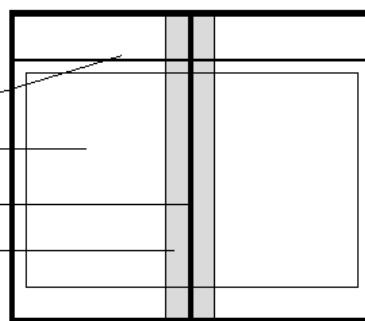
The media definition determines those characteristics that result from the combination of a particular media type together with a particular vendor's printer. The aim is to make it easy to move forms between different vendors' printers which might have different constraints on how they handle a specific media type. It is the Service Provider's responsibility to ensure that the form definition does not specify the printing of any fields that conflict with the media definition. An example of such a conflict might be that the form definition asks for a field to be printed in an area that the media definition defines as an unprintable area.

The media definition is also intended to provide the capability of defining media types that are specific to the financial industry. An example is a passbook as shown below.

### Passbook with horizontal fold



### Passbook with vertical fold



**XFSMEDIA**

**XFSMEDIA**

*medianame\**

**BEGIN**

**TYPE**

*type*

Predefined media types are:

GENERIC (default)

MULTIPART

PASSBOOK

**SOURCE**

*source*

Paper source:

ANY (default)

UPPER

LOWER

EXTERNAL (envelope tray or single sheet feed tray)

AUX

AUX2

PARK

**(required)**

**UNIT**

*base*,

Base resolution unit for form definition:

MM

INCH

ROWCOLUMN

*x*,

Horizontal base unit fraction

*y*

Vertical base unit fraction

**(required)**

**SIZE**

*width*,

Width of physical media

*height*

Height of physical media (0 = unlimited, i.e. roll paper)

**PRINTAREA**

*x*,

Printable area relative to top left corner of physical media (default = physical size of media)

*y*,

*width*,

*height*

**RESTRICTED**

*x*,

Restricted area relative to top left corner of physical media (default = no restricted area)

*y*

*width*,

*height*

**FOLD**

*fold*,

Type of passbook:

HORIZONTAL (default)

VERTICAL

**STAGGERING**

*staggering*

Staggering of passbook from top (default = 0)

**PAGE**

*count*

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
Number of pages in passbook (default = 0)

**LINES**

*count*

Number of printable lines (default = 0)

**END**

## **XFS Form/Media Definition Files in Multi-Vendor Environments**

---

Although for most Service Providers directory location and extension of XFS form/media definition files are configurable through the registry, the capabilities of Service Providers and or actual hardware may vary. Therefore, the following considerations should be taken into account when applications use XFS form definition files with the purpose of running in a multi-vendor environment:

- Physical print area dimensions of printers are not identical.
- Graphic printout may not be supported, which may limit the use of the FONT, CPI and LPI keywords.
- Some printers may have a resolution of dots/mm rather than dots/inch, which may result in printouts with a specific CPI/LPI font resolution to be slightly off size.
- Some form/media definition keywords may not be supported due to limitations of the hardware or software.

## **Command and Event Flows during Single and Multi-Page / Wad Printing**

---

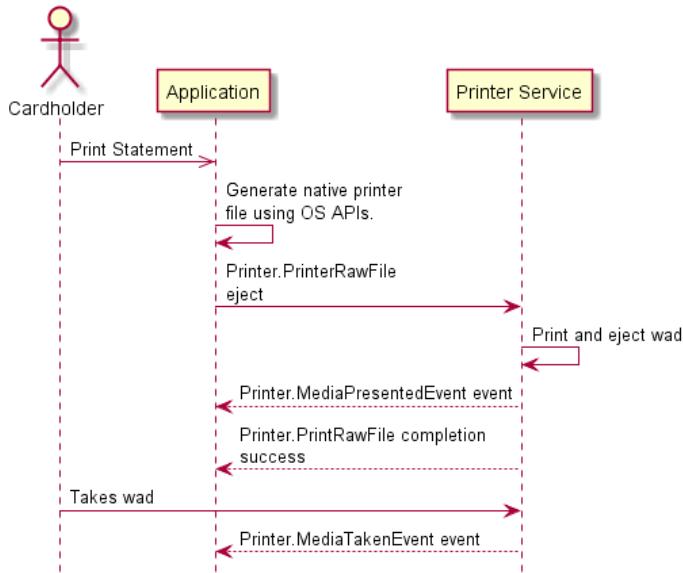
It is possible to print a number of pages or bunches of pages (wads) through the XFS Service Provider. The following sections describe how this is achieved.

### **Single Page / Single Wad Printing With Immediate Media Control**

---

This diagram illustrates the command and event flows in a successful print command (i.e. [Printer.PrintRawFile](#), [Printer.PrintForm](#) and [Printer.PrintRawData](#)) where the following conditions apply:

- A single page or single wad of pages is presented.
- The mediaPresented Capability flag is true (indicates that the [Printer.MediaPresentedEvent](#) event can be generated).
- The mediaControl flag in the command data is set to eject. The [Printer.PrintRawFile](#) command is used as an example.

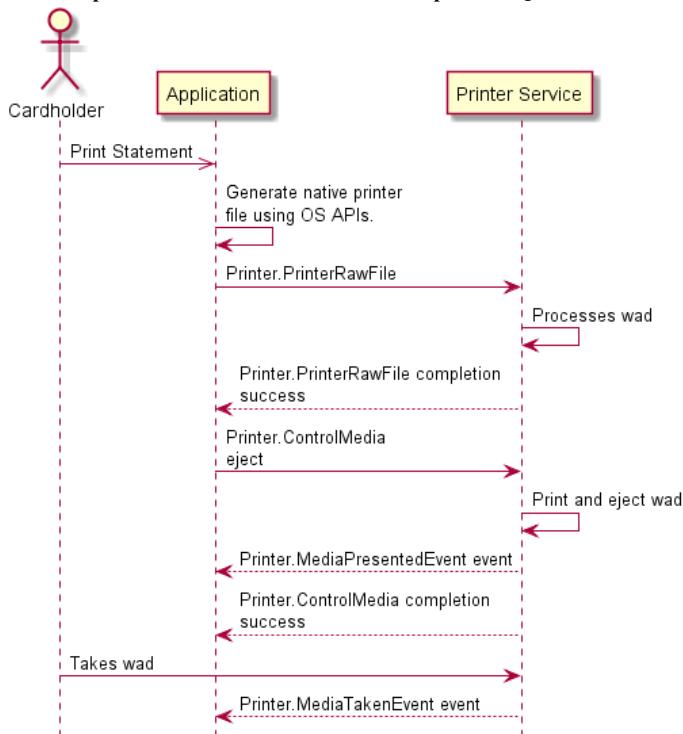


### Single Page / Single Wad Printing With Separate Media Control

---

This diagram illustrates the command and event flows in a successful print command (i.e. [Printer.PrintRawFile](#), [Printer.PrintForm](#) and [Printer.PrintRawData](#)) where the following conditions apply:

- A single page or single wad of pages is presented.
- The mediaPresented Capability flag is true (indicates that the [Printer.MediaPresentedEvent](#) event can be generated).
- The mediaControl flag is not included in the command data. The [Printer.PrintRawFile](#) command is used as an example.
- The media is presented to the user through a [Printer.ControlMedia](#) command, with the mediaControl parameter set to eject.



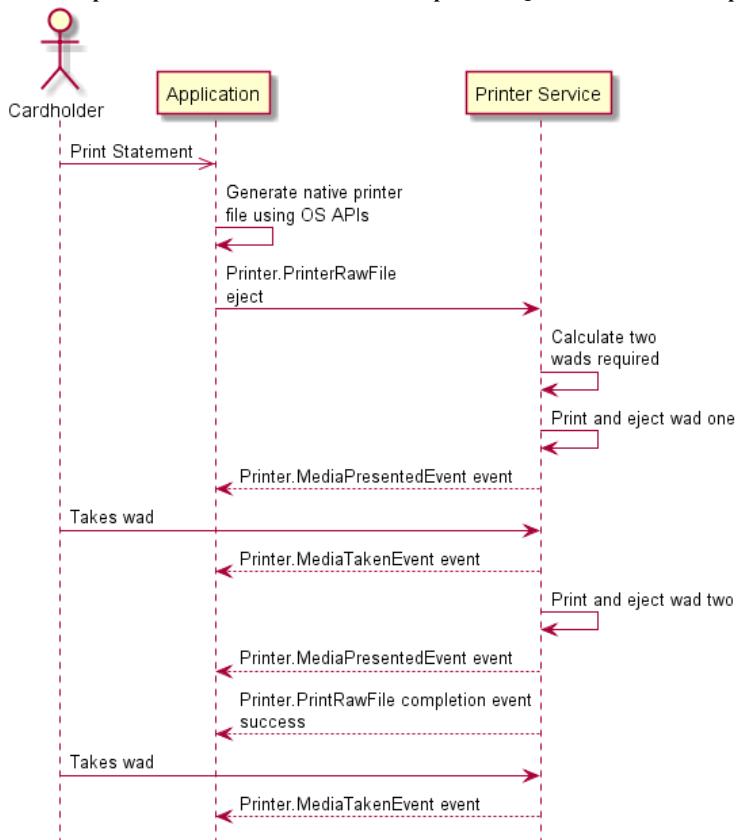
### Multi Page / Multi Wad Printing With Immediate Media Control

---

This diagram illustrates a successful [Printer.PrintRawFile](#) command where multiple page/wads are presented (and the [mediaPresented](#) capability indicates that the [Printer.MediaPresentedEvent](#) can be generated). In addition, the previous page/wad must be removed before subsequent pages/wads can be printed.

The diagram illustrates the command and event flows in a successful print command where the following conditions apply:

- Multiple pages or multiple wads of pages are presented.
- The [mediaPresented](#) capability is true (indicates that the [Printer.MediaPresentedEvent](#) event can be generated).
- The [mediaControl](#) flag in the command data is set to [eject](#).
- The previous page/wad must be removed before subsequent pages/wads can be presented.

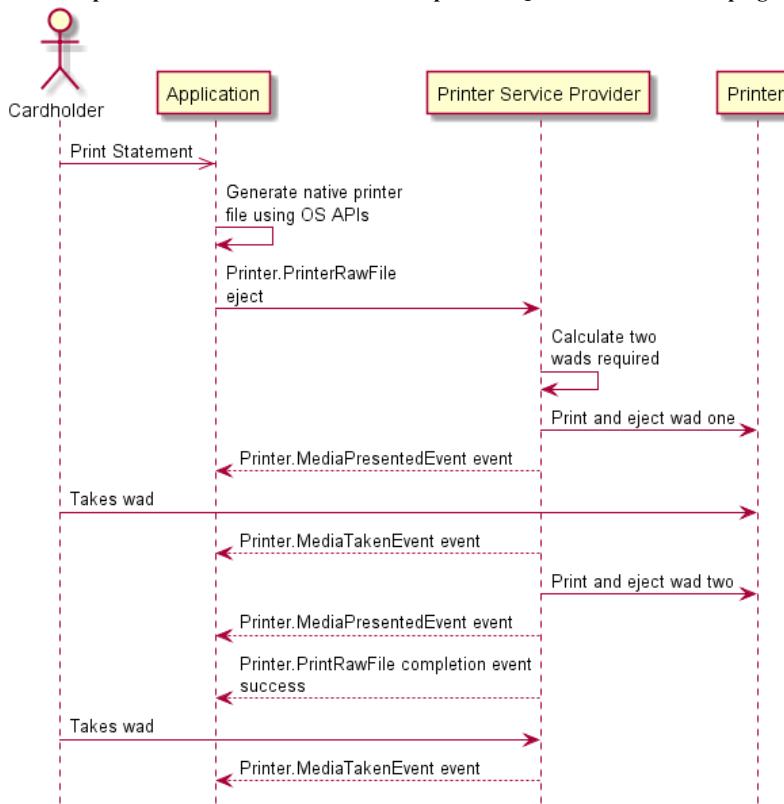


### Multi Page / Multi Wad Printing With Separate Media Control

This diagram illustrates a successful `Printer.PrintRawFile` command where multiple page / wads are presented (and the `mediaPresented` capability indicates that the `Printer.MediaPresentedEvent` can be generated). In addition, the previous page/wad must be removed before subsequent pages/wads can be printed.

The diagram illustrates the command and event flows in a successful print command where the following conditions apply:

- Multiple pages or multiple wads of pages are presented.
- The `mediaPresented` capability is true (indicates that the `Printer.MediaPresentedEvent` event can be generated).
- The `mediaControl` flag command data is omitted.
- The media is presented to the user through a `Printer.ControlMedia` command, with the `mediaControl` property set to `eject`.
- The previous page/wad must be removed before subsequent pages/wads can be presented.



## Commands

### Printer.GetFormList

---

#### Description

This command is used to retrieve the list of forms available on the device.

#### Command Message

##### Message Header

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

##### Message Payload

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
formList	array (string)		The list of form names.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "formList": [
      "string"
    ]
  }
}
```

## Event Messages

---

### Printer.GetMediaList

---

#### Description

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
This command is used to retrieve the list of media definitions available on the device.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
mediaList	array (string)		The list of media names.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "mediaList": [
      "string"
    ]
  }
}
```

## Event Messages

---

### Printer.GetQueryForm

---

#### Description

This command is used to retrieve details of the definition of a specified form.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
formName	string		The form name for which to retrieve details.

#### Example Message (generated)

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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
formName	string		Specifies the name of the form.
base	string		Specifies the base unit of measurement of the form as one of the following: <b>inch</b> The base unit is inches. <b>mm</b> The base unit is millimeters. <b>rowColumn</b> The base unit is rows and columns.
unitX	integer		Specifies the horizontal resolution of the base units as a fraction of the <i>base</i> value. For example, a value of 16 applied to the base unit <i>inch</i> means that the base horizontal resolution is 1/16 inch.
unitY	integer		Specifies the vertical resolution of the base units as a fraction of the <i>base</i> value. For example, a value of 10 applied to the base unit <i>mm</i> means that the base vertical resolution is 0.1 mm.
width	integer		Specifies the width of the form in terms of the base horizontal resolution.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
height	integer		<p>Specifies the height of the form in terms of the base vertical resolution.</p>
alignment	string		<p>Specifies the relative alignment of the form on the media and can be one of the following values:</p> <p><b>topLeft</b> The form is aligned relative to the top and left edges of the media.</p> <p><b>topRight</b> The form is aligned relative to the top and right edges of the media.</p> <p><b>bottomLeft</b> The form is aligned relative to the bottom and left edges of the media.</p> <p><b>bottomRight</b> The form is aligned relative to the bottom and right edges of the media.</p>
orientation	string		<p>Specifies the orientation of the form as one of the following values:</p> <p><b>portrait</b> The orientation of the form is portrait.</p> <p><b>landscape</b> The orientation of the form is landscape.</p>
offsetX	integer		<p>Specifies the horizontal offset of the position of the top-left corner of the form, relative to the left or right edge specified by <i>alignment</i>. This value is specified in terms of the base horizontal resolution and is always positive.</p>
offsetY	integer		<p>Specifies the vertical offset of the position of the top-left corner of the form, relative to the top or bottom edge specified by <i>alignment</i>. This value is specified in terms of the base vertical resolution and is always positive.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
versionMajor	integer		Specifies the major version of the form. If the version is not specified in the form, then zero is returned.
versionMinor	integer		Specifies the minor version of the form. If the version is not specified in the form, then zero is returned.
userPrompt	string		The user prompt string. This will be omitted if the form does not define a value for the user prompt.
fields	array (string)		The field names.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "formName": "string",
    "base": "inch",
    "unitX": 0,
    "unity": 0,
    "width": 0,
    "height": 0,
    "alignment": "topLeft",
    "orientation": "portrait",
    "offsetX": 0,
    "offsetY": 0,
    "versionMajor": 0,
    "versionMinor": 0,
    "userPrompt": "string",
    "fields": [
      "string"
    ]
  }
}
```

**Event Messages**

---

## Printer.GetQueryMedia

---

### Description

This command is used to retrieve details of the definition of a specified media.

### Command Message

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

**Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
mediaName	string		The media name for which to retrieve details.

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
mediaType	string		<p>Specifies the type of media as one of the following:</p> <p><b>generic</b> The media is a generic media, i.e. a single sheet.</p> <p><b>passbook</b> The media is a passbook media.</p> <p><b>multipart</b> The media is a multi part media.</p>
base	string		<p>Specifies the base unit of measurement of the form and can be one of the following values:</p> <p><b>inch</b> The base unit is inches.</p> <p><b>mm</b> The base unit is millimeters.</p> <p><b>rowcolumn</b> The base unit is rows and columns.</p>
unitX	integer		<p>Specifies the horizontal resolution of the base units as a fraction of the <i>base</i> value. For example, a value of 16 applied to the base unit <i>inch</i> means that the base horizontal resolution is 1/16th inch.</p>
unitY	integer		<p>Specifies the vertical resolution of the base units as a fraction of the <i>base</i> value. For example, a value of 10 applied to the base unit <i>mm</i> means that the base vertical resolution is 0.1 mm.</p>
sizeWidth	integer		<p>Specifies the width of the media in terms of the base horizontal resolution.</p>
sizeHeight	integer		<p>Specifies the height of the media in terms of the base vertical resolution.</p>
pageCount	integer		<p>Specifies the number of pages in a media of type <i>passbook</i>.</p>
lineCount	integer		<p>Specifies the number of lines on a page for a media of type <i>passbook</i>.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
printAreaX	integer		Specifies the horizontal offset of the printable area relative to the top left corner of the media in terms of the base horizontal resolution.
printAreaY	integer		Specifies the vertical offset of the printable area relative to the top left corner of the media in terms of the base vertical resolution.
printAreaWidth	integer		Specifies the printable area width of the media in terms of the base horizontal resolution.
printAreaHeight	integer		Specifies the printable area height of the media in terms of the base vertical resolution.
restrictedAreaX	integer		Specifies the horizontal offset of the restricted area relative to the top left corner of the media in terms of the base horizontal resolution.
restrictedAreaY	integer		Specifies the vertical offset of the restricted area relative to the top left corner of the media in terms of the base vertical resolution.
restrictedAreaWidth	integer		Specifies the restricted area width of the media in terms of the base horizontal resolution.
restrictedAreaHeight	integer		Specifies the restricted area height of the media in terms of the base vertical resolution.
stagger	integer		Specifies the staggering from the top in terms of the base vertical resolution for a media of type <i>passbook</i> .
foldType	string		<p>Specified the type of fold for a media of type <i>passbook</i> as one of the following:</p> <ul style="list-style-type: none"> <li><b>none</b> Passbook has no fold.</li> <li><b>horizontal</b> Passbook has a horizontal fold.</li> <li><b>vertical</b> Passbook has a vertical fold.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
paperSources	object		Specifies the Paper sources to use when printing forms using this media as a combination of the following flags
<input type="checkbox"/> any	boolean		Use any paper source.
<input type="checkbox"/> upper	boolean		Use the only or the upper paper source.
<input type="checkbox"/> lower	boolean		Use the lower paper source.
<input type="checkbox"/> external	boolean		Use the external paper source.
<input type="checkbox"/> aux	boolean		Use the auxiliary paper source.
<input type="checkbox"/> aux2	boolean		Use the second auxiliary paper source.
<input type="checkbox"/> park	boolean		Use the parking station.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "mediaType": "generic",
    "base": "inch",
    "unitX": 0,
    "unitY": 0,
    "sizeWidth": 0,
    "sizeHeight": 0,
    "pageCount": 0,
    "lineCount": 0,
    "printAreaX": 0,
    "printAreaY": 0,
    "printAreaWidth": 0,
    "printAreaHeight": 0,
    "restrictedAreaX": 0,
    "restrictedAreaY": 0,
    "restrictedAreaWidth": 0,
    "restrictedAreaHeight": 0,
    "stagger": 0,
    "foldType": "none",
    "paperSources": {
      "any": true,
      "upper": true,
      "lower": true,
      "external": true,
      "aux": true,
      "aux2": true,
      "park": true
    }
  }
}
```

## Event Messages

---

### Printer.GetQueryField

---

#### Description

This command is used to retrieve details of the definition of a single or all fields on a specified form.

#### Command Message

##### Message Header

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

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

**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.
formName	string		The form name.
fieldName	string		The name of the field about which to retrieve details. If omitted, then details are retrieved for all fields on the form.

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
fields	object		Details of the field(s) requested. For each object, the key is the field name.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
<input type="checkbox"/> indexCount	integer		<p>Specifies the number of entries for an index field. A value of zero indicates that this field is not an index field. Index fields are typically used to present information in a tabular fashion.</p>
<input type="checkbox"/> type	string		<p>Specifies the type of field as one of the following:</p> <ul style="list-style-type: none"> <li><b>text</b> The field is a text field.</li> <li><b>micr</b> The field is a Magnetic Ink Character Recognition field.</li> <li><b>ocr</b> The field is an Optical Character Recognition field.</li> <li><b>msf</b> The field is a Magnetic Stripe Facility field.</li> <li><b>barcode</b> The field is a barcode field.</li> <li><b>graphic</b> The field is a Graphic field.</li> <li><b>pageMark</b> The field is a Page Mark field.</li> </ul>
<input type="checkbox"/> class	string		<p>Specifies the class of the field as one of the following:</p> <ul style="list-style-type: none"> <li><b>static</b> The field data cannot be set by the application.</li> <li><b>optional</b> The field data can be set by the application.</li> <li><b>required</b> The field data must be set by the application.</li> </ul>
<input type="checkbox"/> access	string		<p>Specifies the field access as one of the following:</p> <ul style="list-style-type: none"> <li><b>read</b> The field is used for input.</li> <li><b>write</b> The field is used for output.</li> <li><b>readWrite</b> The field is used for both input and output.</li> </ul>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
<input type="checkbox"/> overflow	string		<p>Specifies how an overflow of field data should be handled as one of the following:</p> <p><b>terminate</b> Return an error and terminate printing of the form.</p> <p><b>truncate</b> Truncate the field data to fit in the field.</p> <p><b>bestFit</b> Fit the text in the field.</p> <p><b>overwrite</b> Print the field data beyond the extents of the field boundary.</p> <p><b>wordWrap</b> If the field can hold more than one line the text is wrapped around. Wrapping is performed, where possible, by splitting the line on a space character or a hyphen character or any other character which is used to join two words together.</p>
<input type="checkbox"/> initialValue	string		The initial value of the field. When the form is printed (using <a href="#">Printer.PrintForm</a> ), this value will be used if another value is not provided. This value will be omitted if the parameter is not specified in the field definition.
<input type="checkbox"/> format	string		Format string as defined in the form for this field. This value will be omitted if the parameter is not specified in the field definition.
<input type="checkbox"/> coercivity	string		<p>Specifies the coercivity to be used for writing the magnetic stripe as one of the following:</p> <p><b>auto</b> The coercivity is decided by the Service Provider or the hardware.</p> <p><b>low</b> A low coercivity is to be used for writing the magnetic stripe.</p> <p><b>high</b> A high coercivity is to be used for writing the magnetic stripe.</p>

**Example Message (generated)**

---

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "fields": {
      "property1": {
        "indexCount": 0,
        "type": "text",
        "class": "static",
        "access": "read",
        "overflow": "terminate",
        "initialValue": "string",
        "format": "string",
        "coercivity": "auto"
      },
      "property2": {
        "indexCount": 0,
        "type": "text",
        "class": "static",
        "access": "read",
        "overflow": "terminate",
        "initialValue": "string",
        "format": "string",
        "coercivity": "auto"
      }
    }
  }
}
```

## Event Messages

---

### Printer.GetCodelineMapping

---

#### Description

This command is used to retrieve the byte code mapping for the special banking symbols defined for image processing (e.g. check processing). This mapping must be reported as there is no standard for the fonts defined below.

#### Command Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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.
codelineFormat	string		<p>Specifies the code-line format that the mapping for the special characters is required for. This field can be one of the following values:</p> <p><b>cmc7</b> Report the CMC7 mapping.</p> <p><b>e13b</b> Report the E13B mapping.</p>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "codelineFormat": "cmc7"
  }
}
```

**Completion Message**

**Message Header**

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

**Message Payload**

Name

---

Type

Default

Description

completionCode

string

success if the command was successful otherwise error

errorDescription

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
string

If not success, then this is optional vendor dependent information to provide additional information

codelineFormat

string

Specifies the code-line format that is being reported.

charMapping

string

Defines the mapping of the font specific symbols to byte values. These byte values are used to represent the font specific characters when the code line is read through the Printer.ReadImage command. The font specific meaning of each index is defined in the following tables.

#### **E13B**

Index

Symbol

Meaning

0

■

Transit

1

■

Amount

2

■■

On Us

3

■■■

Dash

4

N/A

Reject / Unreadable

#### **CMC7**

Index

Symbol

Meaning

0

■■

S1 - Start of Bank Account

1

■■■

S2 - Start of the Amount field

!!!

S3 - Terminate Routing

3

!!!

S4 - Unused

4

!!!

S5 - Transit / Routing

5

N/A

Reject / Unreadable

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "codelineFormat": "cmc7",
    "charMapping": "string"
  }
}
```

## Event Messages

### Printer.ControlMedia

#### Description

This command is used to control media.

If an eject operation is specified, it completes when the media is moved to the exit slot. A service event is generated when the media has been taken by the user (only if the [mediaTaken](#) capability is true).

#### Command Message

##### Message Header

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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.  
**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.
mediaControl	object		<p>Specifies the manner in which the media should be handled, as a combination of the following flags:</p> <p>It is not possible to combine the flags eject, retract, park, expel and ejectToTransport with each other otherwise the command completes with <i>errInvalidData</i>.</p>
			<p>It is not possible to combine the flag clearBuffer with any other flags, otherwise the command completes with <i>errInvalidData</i>.</p>
			<p>An application should be aware that the sequence of the actions is not guaranteed if more than one flag is specified in this parameter.</p>
<input type="checkbox"/> eject	boolean		<p>Flush any data to the printer that has not yet been printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands, then eject the media.</p>
<input type="checkbox"/> perforate	boolean		Flush data as per eject, then perforate the media.
<input type="checkbox"/> cut	boolean		Flush data as per eject, then cut the media. For printers which have the ability to stack multiple cut sheets and deliver them as a single bundle to the customer, cut causes the media to be stacked and eject causes the bundle to be moved to the exit slot.
<input type="checkbox"/> skip	boolean		Flush data as per eject, then skip the media to mark.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
<input type="checkbox"/> flush	boolean		Flush any data to the printer that has not yet been physically printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands. This will synchronize the application with the device to ensure that all data has been physically printed.
<input type="checkbox"/> retract	boolean		Flush data as per flush, then retract the media to retract bin number one. For devices with more than one bin the command <a href="#">Printer.RetractMedia</a> should be used if the media should be retracted to another bin than bin number one.
<input type="checkbox"/> stack	boolean		Flush data as per flush, then move the media item on the internal stacker.
<input type="checkbox"/> partialCut	boolean		Flush the data as per flush, then partially cut the media.
<input type="checkbox"/> alarm	boolean		Cause the printer to ring a bell, beep, or otherwise sound an audible alarm.
<input type="checkbox"/> forward	boolean		Flush the data as per flush, then turn one page forward.
<input type="checkbox"/> backward	boolean		Flush the data as per flush, then turn one page backward.
<input type="checkbox"/> turnMedia	boolean		Flush the data as per flush, then turn inserted media.
<input type="checkbox"/> stamp	boolean		Flush the data as per flush, then stamp on inserted media.
<input type="checkbox"/> park	boolean		Park the media in the parking station.
<input type="checkbox"/> expel	boolean		Flush the data as per flush, then throw the media out of the exit slot.
<input type="checkbox"/> ejectToTransport	boolean		Flush the data as per flush, then move the media to a position on the transport just behind the exit slot.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
<input type="checkbox"/> rotate180	boolean		Flush the data as per flush, then rotate media 180 degrees in the printing plane.
<input type="checkbox"/> clearBuffer	boolean		Clear any data that has not yet been physically printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "mediaControl": {
      "eject": true,
      "perforate": true,
      "cut": true,
      "skip": true,
      "flush": true,
      "retract": true,
      "stack": true,
      "partialCut": true,
      "alarm": true,
      "forward": true,
      "backward": true,
      "turnMedia": true,
      "stamp": true,
      "park": true,
      "expel": true,
      "ejectToTransport": true,
      "rotate180": true,
      "clearBuffer": true
    }
  }
}
```

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

## Event Messages

- [Printer.RetractBinThresholdEvent](#)
- [Printer.MediaTakenEvent](#)
- [Printer.PaperThresholdEvent](#)
- [Printer.TonerThresholdEvent](#)
- [Printer.InkThresholdEvent](#)
- [Printer.MediaPresentedEvent](#)
- [Printer.MediaAutoRetractedEvent](#)

## Printer.PrintForm

### Description

This command is used to print a form by merging the supplied variable field data with the defined form and field data specified in the form. If no media is present, the device waits, for the period of time specified by the *timeout* parameter, for media to be inserted from the external paper source.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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.
formName	string		The form name.
mediaName	string		The media name. If no media definition applies, this should be empty or omitted.
alignment	string		<p>Specifies the alignment of the form on the physical media, as one of the following values:</p> <p><b>formDefinition</b> Use the alignment specified in the form definition.</p> <p><b>topLeft</b> Align form to top left of physical media.</p> <p><b>topRight</b> Align form to top right of physical media.</p> <p><b>bottomLeft</b> Align form to bottom left of physical media.</p> <p><b>bottomRight</b> Align form to bottom right of physical media.</p>
offsetX	integer		<p>Specifies the horizontal offset of the form, relative to the horizontal alignment specified in <i>alignment</i>, in horizontal resolution units (from form definition); always a positive number (i.e. if aligned to the right side of the media, means offset the form to the left). A value of <i>formDefinition</i> indicates that the <i>xoffset</i> value from the form definition should be used.</p>
offsetY	integer		<p>Specifies the vertical offset of the form, relative to the vertical alignment specified in <i>alignment</i>, in vertical resolution units (from form definition); always a positive number (i.e. if aligned to the bottom of the media, means offset the form upward). A value of <i>formDefinition</i> indicates that the <i>yoffset</i> value from the form definition should be used.</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
resolution	string		<p>Specifies the resolution in which to print the form. Possible values are:</p> <ul style="list-style-type: none"> <li><b>low</b> Print form with low resolution.</li> <li><b>medium</b> Print form with medium resolution.</li> <li><b>high</b> Print form with high resolution.</li> <li><b>veryHigh</b> Print form with very high resolution.</li> </ul>
mediaControl	object		<p>Specifies the manner in which the media should be handled after the printing is done, as a combination of the following flags. If no flags are set, it means do none of these actions, as when printing multiple forms on a single page. When no flags are set and the device does not support the flush capability, the data will be printed immediately. If the device supports flush, the data may be buffered and the <a href="#">Printer.ControlMedia</a> command should be used to synchronize the application with the device to ensure that all data has been physically printed. The clearBuffer flag is not applicable to this command. If set, the command will fail with error <i>invalidData</i>.</p>
<input type="checkbox"/> eject	boolean		<p>Flush any data to the printer that has not yet been printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands, then eject the media.</p>
<input type="checkbox"/> perforate	boolean		<p>Flush data as per eject, then perforate the media.</p>
<input type="checkbox"/> cut	boolean		<p>Flush data as per eject, then cut the media. For printers which have the ability to stack multiple cut sheets and deliver them as a single bundle to the customer, cut causes the media to be stacked and eject causes the bundle to be moved to the exit slot.</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
<input type="checkbox"/> skip	boolean		Flush data as per eject, then skip the media to mark.
<input type="checkbox"/> flush	boolean		Flush any data to the printer that has not yet been physically printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands. This will synchronize the application with the device to ensure that all data has been physically printed.
<input type="checkbox"/> retract	boolean		Flush data as per flush, then retract the media to retract bin number one. For devices with more than one bin the command <a href="#">Printer.RetractMedia</a> should be used if the media should be retracted to another bin than bin number one.
<input type="checkbox"/> stack	boolean		Flush data as per flush, then move the media item on the internal stacker.
<input type="checkbox"/> partialCut	boolean		Flush the data as per flush, then partially cut the media.
<input type="checkbox"/> alarm	boolean		Cause the printer to ring a bell, beep, or otherwise sound an audible alarm.
<input type="checkbox"/> forward	boolean		Flush the data as per flush, then turn one page forward.
<input type="checkbox"/> backward	boolean		Flush the data as per flush, then turn one page backward.
<input type="checkbox"/> turnMedia	boolean		Flush the data as per flush, then turn inserted media.
<input type="checkbox"/> stamp	boolean		Flush the data as per flush, then stamp on inserted media.
<input type="checkbox"/> park	boolean		Park the media in the parking station.
<input type="checkbox"/> expel	boolean		Flush the data as per flush, then throw the media out of the exit slot.
<input type="checkbox"/> ejectToTransport	boolean		Flush the data as per flush, then move the media to a position on the transport just behind the exit slot.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
<input type="checkbox"/> rotate180	boolean		Flush the data as per flush, then rotate media 180 degrees in the printing plane.
<input type="checkbox"/> clearBuffer	boolean		Clear any data that has not yet been physically printed from previous <a href="#">Pinter.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands.
fields	object		An object containing one or more key/value pairs where the key is a field name and the value is the field value. If the field is an index field, the key must be specified as <b>fieldname[index]</b> where index specifies the zero-based element of the index field. The field names and values can contain UNICODE if supported by the service.
paperSource	string		<p>Specifies the Paper source to use when printing this form. When the value is zero, then the paper source is determined from the media definition. This parameter is ignored if there is already paper in the print position. Possible values are:</p> <p><b>any</b> Any paper source can be used; it is determined by the service.</p> <p><b>upper</b> Use the only paper source or the upper paper source, if there is more than one paper supply.</p> <p><b>lower</b> Use the lower paper source.</p> <p><b>external</b> Use the external paper source (such as envelope tray or single sheet feed).</p> <p><b>aux</b> Use the auxiliary paper source.</p> <p><b>aux2</b> Use the second auxiliary paper source.</p> <p><b>park</b> Use the parking station.</p>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "formName": "string",
    "mediaName": "string",
    "alignment": "formDefinition",
    "offsetX": 0,
    "offsetY": 0,
    "resolution": "low",
    "mediaControl": {
      "eject": true,
      "perforate": true,
      "cut": true,
      "skip": true,
      "flush": true,
      "retract": true,
      "stack": true,
      "partialCut": true,
      "alarm": true,
      "forward": true,
      "backward": true,
      "turnMedia": true,
      "stamp": true,
      "park": true,
      "expel": true,
      "ejectToTransport": true,
      "rotate180": true,
      "clearBuffer": true
    },
    "fields": {
      "property1": "string",
      "property2": "string"
    },
    "paperSource": "any"
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

- [Printer.NoMediaEvent](#)
- [Printer.MediaInsertedEvent](#)
- [Printer.FieldErrorEvent](#)
- [Printer.FieldWarningEvent](#)
- [Printer.RetractBinThresholdEvent](#)
- [Printer.MediaTakenEvent](#)
- [Printer.PaperThresholdEvent](#)
- [Printer.TonerThresholdEvent](#)
- [Printer.InkThresholdEvent](#)
- [Printer.MediaPresentedEvent](#)
- [Printer.MediaRejectedEvent](#)
- [Printer.MediaAutoRetractedEvent](#)

**Printer.ReadForm****Description**

This command is used to read data from input fields on the specified form. These input fields may consist of MICR, OCR, MSF, BARCODE, or PAGEMARK input fields. These input fields may also consist of TEXT fields for purposes of detecting available passbook print lines with passbook printers supporting such capability. If no media is present, the device waits, for the timeout specified, for media to be inserted.

**Command Message****Message Header**

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

**Message Payload**

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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.
formName	string		The name of the form.
fieldNames	array (string)		The field names from which to read input data. If this is omitted or empty, all input fields on the form will be read.
mediaName	string		The media name. If omitted or empty, no media definition applies.
mediaControl	object		Specifies the manner in which the media should be handled after the reading was done and can be a combination of the following flags. The clearBuffer flag is not applicable to this command.
<input type="checkbox"/> eject	boolean		Flush any data to the printer that has not yet been printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands, then eject the media.
<input type="checkbox"/> perforate	boolean		Flush data as per eject, then perforate the media.
<input type="checkbox"/> cut	boolean		Flush data as per eject, then cut the media. For printers which have the ability to stack multiple cut sheets and deliver them as a single bundle to the customer, cut causes the media to be stacked and eject causes the bundle to be moved to the exit slot.
<input type="checkbox"/> skip	boolean		Flush data as per eject, then skip the media to mark.
<input type="checkbox"/> flush	boolean		Flush any data to the printer that has not yet been physically printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands. This will synchronize the application with the device to ensure that all data has been physically printed.
<input type="checkbox"/> retract	boolean		Flush data as per flush, then retract the media to retract bin number one. For devices with more than one bin the command <a href="#">Printer.RetractMedia</a> should be used if the media should be retracted to another bin than bin number one.
<input type="checkbox"/> stack	boolean		Flush data as per flush, then move the media item on the internal stacker.
<input type="checkbox"/> partialCut	boolean		Flush the data as per flush, then partially cut the media.
<input type="checkbox"/> alarm	boolean		Cause the printer to ring a bell, beep, or otherwise sound an audible alarm.
<input type="checkbox"/> forward	boolean		Flush the data as per flush, then turn one page forward.
<input type="checkbox"/> backward	boolean		Flush the data as per flush, then turn one page backward.
<input type="checkbox"/> turnMedia	boolean		Flush the data as per flush, then turn inserted media.
<input type="checkbox"/> stamp	boolean		Flush the data as per flush, then stamp on inserted media.
<input type="checkbox"/> park	boolean		Park the media in the parking station.
<input type="checkbox"/> expel	boolean		Flush the data as per flush, then throw the media out of the exit slot.
<input type="checkbox"/> ejectToTransport	boolean		Flush the data as per flush, then move the media to a position on the transport just behind the exit slot.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
<input type="checkbox"/> rotate180	boolean		Flush the data as per flush, then rotate media 180 degrees in the printing plane.
<input type="checkbox"/> clearBuffer	boolean		Clear any data that has not yet been physically printed from previous <a href="#">Pinter.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "formName": "string",
    "fieldNames": [
      "string"
    ],
    "mediaName": "string",
    "mediaControl": {
      "eject": true,
      "perforate": true,
      "cut": true,
      "skip": true,
      "flush": true,
      "retract": true,
      "stack": true,
      "partialCut": true,
      "alarm": true,
      "forward": true,
      "backward": true,
      "turnMedia": true,
      "stamp": true,
      "park": true,
      "expel": true,
      "ejectToTransport": true,
      "rotate180": true,
      "clearBuffer": true
    }
  }
}
```

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
fields	object		An object containing one or more key/value pairs where the key is a field name and the value is the field value. If the field is an index field, the key must be specified as <b>fieldname[index]</b> where index specifies the zero-based element of the index field. The field names and values can contain UNICODE if supported by the service.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "fields": {
      "property1": "string",
      "property2": "string"
    }
  }
}
```

**Event Messages**

- [Printer.NoMediaEvent](#)
- [Printer.MediaInsertedEvent](#)
- [Printer.FieldErrorEvent](#)
- [Printer.FieldWarningEvent](#)
- [Printer.RetractBinThresholdEvent](#)
- [Printer.MediaTakenEvent](#)
- [Printer.InkThresholdEvent](#)
- [Printer.LampThresholdEvent](#)
- [Printer.MediaRejectedEvent](#)

**Printer.RawData****Description**

This command is used to send raw data (a byte string of device dependent data) to the physical device.

**Command Message****Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
name <b>(Required)</b>	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.
inputData	string		<p>Specifies that input data from the device is expected in response to sending the raw data (i.e. the data contains a command requesting data). Possible values are:</p> <p><b>no</b> No input data is expected.</p> <p><b>yes</b> Input data is expected.</p>
data	string		BASE64 encoded device dependent data to be sent to the device.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "inputData": "no",
    "data": "JUI0ODE1ODgxMDAyODYxODk2X11BVEVTL0VVR0VORSBKT0hOICAgICAgICAgXjM3ODI5ODIxMDAw MTIzNDU2Nzg5Pw==\n"
  }
}
```

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
data	string		BASE64 encoded device dependent data received from the device.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "data": "JUI0ODE1ODgxMDAyODYxODk2X11BVEVTL0VVR0VORSBKT0hOICAgICAgICAgXjM3ODI5ODIxMDAw MTIzNDU2Nzg5Pw==\n"
  }
}
```

## Event Messages

- [Printer\\_RetractionBinThresholdEvent](#)
- [Printer\\_MediaTakenEvent](#)
- [Printer\\_PaperThresholdEvent](#)
- [Printer\\_TonerThresholdEvent](#)
- [Printer\\_MediaPresentedEvent](#)
- [Printer\\_MediaAutoRetractedEvent](#)

## Printer.MediaExtents

### Description

This command is used to get the extents of the media inserted in the physical device. The input parameter specifies the base unit and fractions in which the media extent values will be returned. If no media is present, the device waits, for the timeout specified, for media to be inserted.

### Command Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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.
base	string		<p>Specifies the base unit of measurement of the media and can be one of the following values:</p> <p><b>inches</b> The base unit is inches.</p> <p><b>mm</b> The base unit is millimeters.</p> <p><b>rowColumn</b> The base unit is rows and columns.</p>
unitX	integer		<p>Specifies the horizontal resolution of the base units as a fraction of the base value. For example, a value of 16 applied to the base unit, inches, means that the base horizontal resolution is 1/16.</p>
unitY	integer		<p>Specifies the vertical resolution of the base units as a fraction of the base value. For example, a value of 10 applied to the base unit, mm, means that the base vertical resolution is 0.1 mm.</p>

**Example Message (generated)**

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

**Completion Message**

**Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
sizeX	integer		Specifies the width of the media in terms of the base horizontal resolution.
sizeY	integer		Specifies the height of the media in terms of the base vertical resolution.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "String",
    "sizeX": 0,
    "sizeY": 0
  }
}
```

**Event Messages**

- [Printer.NoMediaEvent](#)
- [Printer.MediaInsertedEvent](#)
- [Printer.MediaRejectedEvent](#)
- [Printer.MediaTakenEvent](#)

**Printer.ResetCount****Description**

This function resets the present value for number of media items retracted to zero. The function is possible only for printers with retract capability.

The number of media items retracted is controlled by the service and can be requested before resetting using the info Printer.Status command.

**Command Message****Message Header**

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
binNumber	integer		Specifies the height of the media in terms of the base vertical resolution.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

## Event Messages

- [Printer.RetractBinThresholdEvent](#)
- 

## Printer.ReadImage

---

### Description

This function returns image data from the current media. If no media is present, the device waits for the timeout specified, for media to be inserted.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <span>(Required)</span>	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 <span>(Required)</span>	string		The message type, either command, response, event or completion.
name <span>(Required)</span>	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.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
frontImageType	string		<p>Specifies the format of the front image returned by this command as one of the following. If omitted, no front image is returned.</p> <p><b>tif</b> The returned image is in TIF 6.0 format.</p> <p><b>wmf</b> The returned image is in WMF (Windows Metafile) format.</p> <p><b>bmp</b> The returned image is in BMP format.</p> <p><b>jpg</b> The returned image is in JPG format.</p>
backImageType	string		<p>Specifies the format of the back image returned by this command as one of the following. If omitted, no back image is returned.</p> <p><b>tif</b> The returned image is in TIF 6.0 format.</p> <p><b>wmf</b> The returned image is in WMF (Windows Metafile) format.</p> <p><b>bmp</b> The returned image is in BMP format.</p> <p><b>jpg</b> The returned image is in JPG format.</p>
frontImageColorFormat	string		<p>Specifies the color format of the requested front image as one of the following:</p> <p><b>binary</b> The scanned images has to be returned in binary (image contains two colors, usually the colors black and white).</p> <p><b>grayscale</b> The scanned images has to be returned in gray scale (image contains multiple gray colors).</p> <p><b>fullcolor</b> The scanned images has to be returned in full color (image contains colors like red, green, blue etc.).</p>

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
backImageColorFormat	string		<p>Specifies the color format of the requested back image as one of the following:</p> <p><b>binary</b> The scanned images has to be returned in binary (image contains two colors, usually the colors black and white).</p> <p><b>grayscale</b> The scanned images has to be returned in gray scale (image contains multiple gray colors).</p> <p><b>fullcolor</b> The scanned images has to be returned in full color (image contains colors like red, green, blue etc.).</p>
codelineFormat	string		<p>Specifies the code line (MICR data) format, as one of the following flags (zero if source not selected):</p> <p><b>cmc7</b> Read CMC7 code line.</p> <p><b>e13b</b> Read E13B code line.</p> <p><b>ocr</b> Read code line using OCR.</p>
imageSource	object		<p>Specifies the source as a combination of the following flags:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> front boolean The front image of the document is requested.</li> <li><input type="checkbox"/> back boolean The back image of the document is requested.</li> <li><input type="checkbox"/> codeline boolean The code line of the document is requested.</li> </ul>
frontImageFile	string		<p>File specifying where to store the front image, e.g. "C:\Temp\FrontImage.bmp". If omitted or empty, the front image data will be returned in the output parameter. This value must not contain UNICODE characters.</p> <p>To reduce the size of data sent between the client and service, it is recommended to use of this parameter.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
backImageFile	string		<p>File specifying where to store the back image, e.g. "C:\Temp\BackImage.bmp". If omitted or empty, the back image data will be returned in the output parameter. This value must not contain UNICODE characters.</p> <p>To reduce the size of data sent between the client and service, it is recommended to use of this parameter.</p>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "frontImageType": "tif",
    "backImageType": "tif",
    "frontImageColorFormat": "binary",
    "backImageColorFormat": "binary",
    "codelineFormat": "cmc7",
    "imageSource": {
      "front": true,
      "back": true,
      "codeline": true
    },
    "frontImageFile": "string",
    "backImageFile": "string"
  }
}
```

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
images	object		The status and data for each of the requested images.
<input type="checkbox"/> front	object		The front image status and data.
status	string		Status of data source. Possible values are:  <b>ok</b> The data is OK.  <b>notSupp</b> The data source is not supported.  <b>missing</b> The data source is missing, for example, the Service is unable to get the code line.
data	string		If the image source is front or back and the image data has not been stored to the hard disk (file name not provided), this will contain the Base64 encoded image. If the image source is codeline, this contains characters in the ASCII range. If the code line was read using the OCR-A font then the ASCII codes will conform to Figure E1 in ANSI X3.17-1981. If the code line was read using the OCR-B font then the ASCII codes will conform to Figure C2 in ANSI X3.49-1975. In both these cases unrecognized characters will be reported as the REJECT code, 0x1A. The E13B and CMC7 fonts use the ASCII equivalents for the standard characters and use the byte values as reported by the Printer.CodelineMapping command for the symbols that are unique to MICR fonts.
<input type="checkbox"/> back	object		The back image status and data.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
<input type="checkbox"/> status	string		<p>Status of data source. Possible values are:</p> <p><b>ok</b> The data is OK.</p> <p><b>notSupp</b> The data source is not supported.</p> <p><b>missing</b> The data source is missing, for example, the Service is unable to get the code line.</p>
<input type="checkbox"/> data	string		If the image source is front or back and the image data has not been stored to the hard disk (file name not provided), this will contain the Base64 encoded image. If the image source is codeline, this contains characters in the ASCII range. If the code line was read using the OCR-A font then the ASCII codes will conform to Figure E1 in ANSI X3.17-1981. If the code line was read using the OCR-B font then the ASCII codes will conform to Figure C2 in ANSI X3.49-1975. In both these cases unrecognized characters will be reported as the REJECT code, 0x1A. The E13B and CMCT fonts use the ASCII equivalents for the standard characters and use the byte values as reported by the Printer.CodelineMapping command for the symbols that are unique to MICR fonts.
<input type="checkbox"/> codeline	object		The codeline status and data.
<input type="checkbox"/> status	string		<p>Status of data source. Possible values are:</p> <p><b>ok</b> The data is OK.</p> <p><b>notSupp</b> The data source is not supported.</p> <p><b>missing</b> The data source is missing, for example, the Service is unable to get the code line.</p>

Name	Type	Default	Description
<input type="checkbox"/> data	string		If the image source is front or back and the image data has not been stored to the hard disk (file name not provided), this will contain the Base64 encoded image. If the image source is codeline, this contains characters in the ASCII range. If the code line was read using the OCR-A font then the ASCII codes will conform to Figure E1 in ANSI X3.17-1981. If the code line was read using the OCR-B font then the ASCII codes will conform to Figure C2 in ANSI X3.49-1975. In both these cases unrecognized characters will be reported as the REJECT code, 0x1A. The E13B and CMC7 fonts use the ASCII equivalents for the standard characters and use the byte values as reported by the Printer.CodelineMapping command for the symbols that are unique to MICR fonts.

#### Example Message (generated)

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

#### Event Messages

- [Printer.NoMediaEvent](#)
- [Printer.MediaInsertedEvent](#)
- [Printer.MediaTakenEvent](#)
- [Printer.LampThresholdEvent](#)

- [Printer.MediaRejectedEvent](#)
- [Printer.MediaAutoRetractedEvent](#)

---

## Printer.Reset

---

### Description

This command is used by the application to perform a hardware reset which will attempt to return the device to a known good state.

The device will attempt to retract or eject any items found anywhere within the device. This may not always be possible because of hardware problems. The [Printer.MediaDetectedEvent](#) event will inform the application where items were actually moved to.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <small>(Required)</small>	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 <small>(Required)</small>	string		The message type, either command, response, event or completion.
name <small>(Required)</small>	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.
mediaControl	string		<p>Specifies the manner in which the media should be handled, as one of the following:</p> <p><b>eject</b> Eject the media.</p> <p><b>retract</b> Retract the media to retract bin number specified.</p> <p><b>expel</b> Throw the media out of the exit slot.</p>
retractBinNumber	integer		Number of the retract bin the media is retracted to. This number has to be between one and the <a href="#">number of bins</a> supported by this device. It is only relevant if <i>mediaControl</i> is <i>retract</i> .

---

#### Example Message (generated)

---

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [Printer.MediaDetectedEvent](#)
- [Printer.RetractBinThresholdEvent](#)
- [Printer.MediaAutoRetractedEvent](#)
- [Printer.MediaPresentedEvent](#)

## Printer.RetractMedia

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

### Description

The media is removed from its present position (media inserted into device, media entering, unknown position) and stored in one of the retract bins. An event is sent if the storage capacity of the specified retract bin is reached. If the bin is already full and the command cannot be executed, an error is returned and the media remains in its present position.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
binNumber	integer		This number has to be between one and the number of bins supported by this device. If omitted, the media will be retracted to the transport. After it has been retracted to the transport, in a subsequent operation the media can be ejected again, or retracted to one of the retract bins.

#### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
binNumber	integer		The number of the retract bin where the media has actually been deposited.

**Example Message (generated)**

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

**Event Messages**

- [Printer.RetракtBinThresholdEvent](#)

## Printer.DispensePaper

**Description**

This command is used to move paper (which can also be a new passbook) from a paper source into the print position.

**Command Message**

**Message Header**

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
			The paper source to dispense from. It can be one of the following:
		<b>any</b>	Any paper source can be used; it is determined by the service.
		<b>upper</b>	Use the only paper source or the upper paper source, if there is more than one paper supply.
		<b>lower</b>	Use the lower paper source.
		<b>internal</b>	Use the external paper.
		<b>aux</b>	Use the auxiliary paper source.
		<b>aux2</b>	Use the second auxiliary paper source.
		<b>park</b>	Use the parking station paper source.
paperSource	string		

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

- [Printer.PaperThresholdEvent](#)
- [Printer.MediaPresentedEvent](#)
- [Printer.MediaAutoRetractedEvent](#)

## Printer.PrintRawFile

### Description

This command is used to print a file that contains a complete print job in the native printer language. The creation and content of this file are both Operating System and printer specific and outwith the scope of this specification. If no media is present, the device waits, for the timeout specified, for media to be inserted from the external paper source. This command must not complete until all pages have been presented to the customer.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
fileName	string		This is the full path and file name of the file to be printed. This value cannot contain UNICODE characters.
mediaControl	object		Specifies the manner in which the media should be handled after each page is printed, as a combination of the following flags. If no flags are set, no actions will be performed, as when printing multiple pages on a single media item. Note that the clearBuffer flag is not applicable to this command and will be ignored.
<input type="checkbox"/> eject	boolean		Flush any data to the printer that has not yet been printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands, then eject the media.
<input type="checkbox"/> perforate	boolean		Flush data as per eject, then perforate the media.
<input type="checkbox"/> cut	boolean		Flush data as per eject, then cut the media. For printers which have the ability to stack multiple cut sheets and deliver them as a single bundle to the customer, cut causes the media to be stacked and eject causes the bundle to be moved to the exit slot.
<input type="checkbox"/> skip	boolean		Flush data as per eject, then skip the media to mark.
<input type="checkbox"/> flush	boolean		Flush any data to the printer that has not yet been physically printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands. This will synchronize the application with the device to ensure that all data has been physically printed.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
<input type="checkbox"/> retract	boolean		Flush data as per flush, then retract the media to retract bin number one. For devices with more than one bin the command <a href="#">Printer.RetractMedia</a> should be used if the media should be retracted to another bin than bin number one.
<input type="checkbox"/> stack	boolean		Flush data as per flush, then move the media item on the internal stacker.
<input type="checkbox"/> partialCut	boolean		Flush the data as per flush, then partially cut the media.
<input type="checkbox"/> alarm	boolean		Cause the printer to ring a bell, beep, or otherwise sound an audible alarm.
<input type="checkbox"/> forward	boolean		Flush the data as per flush, then turn one page forward.
<input type="checkbox"/> backward	boolean		Flush the data as per flush, then turn one page backward.
<input type="checkbox"/> turnMedia	boolean		Flush the data as per flush, then turn inserted media.
<input type="checkbox"/> stamp	boolean		Flush the data as per flush, then stamp on inserted media.
<input type="checkbox"/> park	boolean		Park the media in the parking station.
<input type="checkbox"/> expel	boolean		Flush the data as per flush, then throw the media out of the exit slot.
<input type="checkbox"/> ejectToTransport	boolean		Flush the data as per flush, then move the media to a position on the transport just behind the exit slot.
<input type="checkbox"/> rotate180	boolean		Flush the data as per flush, then rotate media 180 degrees in the printing plane.
<input type="checkbox"/> clearBuffer	boolean		Clear any data that has not yet been physically printed from previous <a href="#">Printer.PrintForm</a> or <a href="#">Printer.PrintRawFile</a> commands.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
paperSource	string		<p>Specifies the paper source to use when printing. If omitted, the Service Provider will determine the paper source that will be used. This parameter is ignored if there is already paper in the print position. Possible values are:</p> <ul style="list-style-type: none"> <li><b>*any</b> Any paper source can be used; it is determined by the service.</li> <li><b>*upper</b> Use the only paper source or the upper paper source, if there is more than one paper supply.</li> <li><b>lower</b> Use the lower paper source.</li> <li><b>external</b> Use the external paper source (such as envelope tray or single sheet feed).</li> <li><b>aux</b> Use the auxiliary paper source.</li> <li><b>aux2</b> Use the second auxiliary paper source.</li> <li><b>park</b> Use the parking station.</li> </ul>

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "fileName": "string",
    "mediaControl": {
      "eject": true,
      "perforate": true,
      "cut": true,
      "skip": true,
      "flush": true,
      "retract": true,
      "stack": true,
      "partialCut": true,
      "alarm": true,
      "forward": true,
      "backward": true,
      "turnMedia": true,
      "stamp": true,
      "park": true,
      "expel": true,
      "ejectToTransport": true,
      "rotate180": true,
      "clearBuffer": true
    },
    "paperSource": "any"
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [Printer.NoMediaEvent](#)
- [Printer.MediaInsertedEvent](#)
- [Printer.MediaPresentedEvent](#)
- [Printer.MediaTakenEvent](#)
- [Printer.PaperThresholdEvent](#)
- [Printer.TonerThresholdEvent](#)
- [Printer.RetractBinThresholdEvent](#)
- [Printer.InkThresholdEvent](#)
- [Printer.MediaRejectedEvent](#)
- [Printer.MediaAutoRetractedEvent](#)

## Printer.LoadDefinition

### Description

This command is used to load a form (including sub-forms and frames) or media definition into the list of available forms. Once a form or media definition has been loaded through this command it can be used by any of the other form/media definition processing commands. Forms and media definitions loaded through this command are persistent. When a form or media definition is loaded a [Printer.DefinitionLoadedEvent](#) event is generated to inform applications that a form or media definition has been added or replaced.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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.
fileName	string		This is the full path and file name of the file to be loaded. This value cannot contain UNICODE characters. The file contains the form (including sub-forms and frames) or media definition in text format as described in <a href="#">Form, Sub-Form, Field, Frame, Table and Media Definitions</a> . Only one form or media definition can be defined in the file.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
overwrite	boolean		Specifies if an existing form or media definition with the same name is to be replaced. If this flag is true then an existing form or media definition with the same name will be replaced, unless the command fails with an error, where the definition will remain unchanged. If this flag is false this command will fail with an error if the form or media definition already exists.

**Example Message (generated)**

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

**Completion Message**

**Message Header**

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

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

- [Printer.DefinitionLoadedEvent](#)

## Printer.SupplyReplenish

---

### Description

After the supplies have been replenished, this command is used to indicate that one or more supplies have been replenished and are expected to be in a healthy state.

Hardware that cannot detect the level of a supply and reports on the supply's status using metrics (or some other means), must assume the supply has been fully replenished after this command is issued. The appropriate threshold event must be broadcast.

Hardware that can detect the level of a supply must update its status based on its sensors, generate a threshold event if appropriate, and succeed the command even if the supply has not been replenished. If it has already detected the level and reported the threshold before this command was issued, the command must succeed and no threshold event is required.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <span>(Required)</span>	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 <span>(Required)</span>	string		The message type, either command, response, event or completion.
name <span>(Required)</span>	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.
upper	boolean		The only paper supply or the upper paper supply was replenished.
lower	boolean		The lower paper supply was replenished.
aux	boolean		The auxiliary paper supply was replenished.
aux2	boolean		The second auxiliary paper supply was replenished.
toner	boolean		The toner supply was replenished.
ink	boolean		The ink supply was replenished.
lamp	boolean		The imaging lamp was replaced.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "upper": true,
    "lower": true,
    "aux": true,
    "aux2": true,
    "toner": true,
    "ink": true,
    "lamp": true
  }
}
```

## Completion Message

### Message Header

Name	Type	Default	Description
requestId <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.
name <span style="background-color: #ccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

- [Printer.PaperThresholdEvent](#)
- [Printer.TonerThresholdEvent](#)
- [Printer.InkThresholdEvent](#)
- [Printer.LampThresholdEvent](#)

## Description

This command can turn the pages of a passbook inserted in the printer by a specified number of pages in a specified direction and it can close the passbook. The [controlPassbook](#) field returned by [Common.Capabilities](#) specifies which functionality is supported. This command flushes the data before the pages are turned or the passbook is closed.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
action	string		<p>Specifies the direction of the page turn as one of the following values:</p> <p><b>forward</b> Turns forward the pages of the passbook.</p> <p><b>backward</b> Turns backward the pages of the passbook.</p> <p><b>closeForward</b> Close the passbook forward.</p> <p><b>closeBackward</b> Close the passbook backward.</p>
count	integer		Specifies the number of pages to be turned. In the case where <i>action</i> is closeForward or closeBackward, this field will be ignored.

### Example Message (generated)

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

### Completion Message

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

---

## Printer.SetBlackMarkMode

---

#### Description

This command switches the black mark detection mode and associated functionality on or off. The black mark detection mode is persistent. If the selected mode is already active this command will complete with success.

---

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
blackMarkMode	string		<p>Specifies the desired black mark detection mode as one of the following:</p> <ul style="list-style-type: none"> <li><b>on</b> Turns the black mark detection and associated functionality on.</li> <li><b>off</b> Turns the black mark detection and associated functionality off.</li> </ul>

### Example Message (generated)

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

## Completion Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

**Message Payload**

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

**Example Message (generated)**

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

**Event Messages**

## Unsolicited Events

### Printer.MediaTakenEvent

**Description**

This event is sent when the media is taken from the exit slot following the completion of a successful eject operation or following a [Printer.MediaRejectedEvent](#). For devices that do not physically move media, this event may also be generated when the media is taken from the device.

**Message Header**

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

**Example Message (generated)**

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

## Printer.MediaInsertedUnsolicitedEvent

### Description

This event specifies that the physical media has been inserted into the device without any read or print execute commands being executed. This event is only generated when media is entered in an unsolicited manner.

### Message Header

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

### Example Message (generated)

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

## Printer.MediaPresentedUnsolicitedEvent

### Description

This event is used to indicate when media has been presented to the customer for removal as a result of a print operation through some non XFS interface.

### Message Header

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

Message	Type	Default	Description
Name	Type	Default	Description
wadIndex	integer		Specifies the index (starting from one) of the presented wad, where a Wad is a bunch of one or more pages presented as a bunch.
totalWads	integer		Specifies the total number of wads in the print job, zero if the total number of wads is not known.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "wadIndex": 1,
    "totalWads": 0
  }
}
```

**Printer.MediaDetectedEvent****Description**

This event is generated when a media is detected in the device during a reset operation.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

Name	Type	Default	Description
position	string		<p>Specifies the media position after the reset operation, as one of the following values:</p> <ul style="list-style-type: none"> <li><b>retracted</b> The media was retracted during the reset operation.</li> <li><b>present</b> The media is in the print position or on the stacker.</li> <li><b>entering</b> The media is in the exit slot.</li> <li><b>jammed</b> The media is jammed in the device.</li> <li><b>unknown</b> The media is in an unknown position.</li> <li><b>expelled</b> The media was expelled during the reset operation.</li> </ul>
retractBinNumber	integer		<p>Number of the retract bin the media was retracted to. This number has to be between one and the <a href="#">number of bins</a> supported by this device. It is only relevant if <i>position</i> is <i>retracted</i>.</p>

#### Example Message (generated)

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

## Printer.RetractBinStatusEvent

### Description

This event specifies that the status of the retract bin holding the retracted media has changed.

### Message Header

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

#### Message Payload

Name	Type	Default	Description
binNumber	integer		Number of the retract bin for which the status has changed.
state	string		<p>Specifies the current state of the retract bin as one of the following values:</p> <p><b>inserted</b> The retract bin has been inserted.</p> <p><b>removed</b> The retract bin has been removed.</p>

#### Example Message (generated)

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

## Printer.DefinitionLoadedEvent

#### Description

This event is used to indicate when a form or media definition has successfully been loaded via the Printer.LoadDefinition command.

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
name	string		Specifies the name of the form or media just loaded.
type	string		Specifies the type of definition loaded. This field can be one of the following values:  <b>form</b> The form identified by <i>name</i> has been loaded.  <b>media</b> The media identified by <i>name</i> has been loaded.

#### Example Message (generated)

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

## Printer.MediaAutoRetractedEvent

#### Description

This event indicates when media has been automatically retracted by the device. Support for this event is indicated when [autoRetractPeriod](#) is non-zero. The event can be generated as the result of any command that presents media to the customer.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
retractResult	string	<b>ok</b> The media was retracted successfully.  <b>jammed</b> The media is jammed.	Specifies the result of the automatic retraction, as one of the following values:
binNumber	integer		Number of the retract bin the media was retracted to or zero if the media is retracted to the transport. This number has to be between zero and the number of bins supported by this device. This value is also zero if <i>retractResult</i> is <i>jammed</i> .

**Example Message (generated)**

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

## Printer.RetractBinThresholdEvent

### Description

This event specifies that the status of the retract bin holding the retracted media has changed.

### Message Header

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

### Message Payload

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
binNumber	integer		Number of the retract bin for which the status has changed.
state	string		<p>Specifies the current state of the retract bin as one of the following:</p> <p><b>ok</b> The retract bin of the printer is in a good state.</p> <p><b>full</b> The retract bin of the printer is full.</p> <p><b>high</b> The retract bin of the printer is high.</p>

**Example Message (generated)**

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

## Printer.PaperThresholdEvent

### Description

This user event is used to specify that the state of the paper reached a threshold. There is no threshold defined for the parking station as this can contain only one paper item.

### Message Header

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

### Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
			Specifies the paper source as one of the following: <b>upper</b> The only paper source or the upper paper source, if there is more than one paper supply.
		<b>lower</b>	The lower paper source.
		<b>external</b>	The external paper source (such as envelope tray or single sheet feed).
		<b>aux</b>	The auxiliary paper source.
		<b>aux2</b>	The second auxiliary paper source.
			Specifies the current state of the paper source as one of the following:
		<b>full</b>	The paper in the paper source is in a good state.
		<b>low</b>	The paper in the paper source is low.
		<b>out</b>	The paper in the paper source is out.
<b>paperSource</b>	string		
<b>threshold</b>	string		

#### Example Message (generated)

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

---

## Printer.TonerThresholdEvent

---

### Description

This user event is used to specify that the state of the toner (or ink) reached a threshold.

---

### Message Header

---

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

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

#### Message Payload

Name	Type	Default	Description
state	string		<p>Specifies the current state of the toner (or ink) as one of the following:</p> <ul style="list-style-type: none"> <li><b>full</b> The toner (or ink) in the printer is in a good state.</li> <li><b>low</b> The toner (or ink) in the printer is low.</li> <li><b>out</b> The toner (or ink) in the printer is out.</li> </ul>

#### Example Message (generated)

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

## Printer.LampThresholdEvent

#### Description

This user event is used to specify that the state of the imaging lamp reached a threshold.

#### Message Header

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

Message Payload Default	Description		
Name	Type	Default	Description
state	string		<p>Specifies the current state of the imaging lamp as one of the following values:</p> <ul style="list-style-type: none"> <li><b>ok</b> The imaging lamp is in a good state.</li> <li><b>fading</b> The imaging lamp is fading and should be changed.</li> <li><b>inop</b> The imaging lamp is inoperative.</li> </ul>

**Example Message (generated)**

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

**Printer.InkThresholdEvent****Description**

This user event is used to specify that the state of the stamping ink reached a threshold.

**Message Header**

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

**Message Payload**

Name	Type	Default	Description

Name	Type	Default	Description
state	string		<p>Specifies the current state of the stamping ink as one of the following:</p> <ul style="list-style-type: none"> <li><b>full</b> The stamping ink in the printer is in a good state.</li> <li><b>low</b> The stamping ink in the printer is low.</li> <li><b>out</b> The stamping ink in the printer is out.</li> </ul>

#### Example Message (generated)

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

## Events

### Printer.MediaPresentedEvent

#### Description

This event is used to indicate when media has been presented to the customer for removal.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
wadIndex	integer		Specifies the index (starting from one) of the presented wad, where a Wad is a bunch of one or more pages presented as a bunch.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
totalWads	integer		Specifies the total number of wads in the print job, zero if the total number of wads is not known.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "wadIndex": 1,
    "totalWads": 0
  }
}
```

## Printer.NoMediaEvent

### Description

This event specifies that the physical media must be inserted into the device in order for the execute command to proceed.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
userPrompt	string		The user prompt from the form definition. This will be omitted if either a form does not define a value for the user prompt or the event is being generated as the result of a command that does not use forms. The application may use the this in any manner it sees fit, for example it might display the string to the operator, along with a message that the media should be inserted.

**Example Message (generated)**

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

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Printer.MedialInsertedEvent

---

### Description

This event specifies that the physical media has been inserted into the device.

The application may use this event to, for example, remove a message box from the screen telling the user to insert media.

### Message Header

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

### Example Message (generated)

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

---

## Printer.FieldErrorEvent

---

### Description

This event specifies that a fatal error has occurred while processing a field.

### Message Header

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

### Message Payload

Name	Type	Default	Description
formName	string		The form name.
fieldName	string		The field name.

Name	Type	Default	Description
			Specifies the type of failure as one of the following:
			<b>required</b> The specified field must be supplied by the application.
			<b>staticOverwrite</b> The specified field is static and thus cannot be overwritten by the application.
			<b>overflow</b> The value supplied for the specified fields is too long.
			<b>notFound</b> The specified field does not exist.
			<b>notRead</b> The specified field is not an input field.
			<b>notWrite</b> An attempt was made to write to an input field.
			<b>hwerror</b> The specified field uses special hardware (e.g. OCR, Low/High coercivity, etc) and an error occurred.
			<b>notSupported</b> The form field type is not supported with device.
			<b>graphic</b> The specified graphic image could not be printed.
failure	string		

#### Example Message (generated)

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

## Printer.FieldWarningEvent

### Description

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**  
This event specifies that a non-fatal error has occurred while processing a field.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
formName	string		The form name.
fieldName	string		The field name.
			Specifies the type of failure as one of the following:
			<b>required</b> The specified field must be supplied by the application.
			<b>staticOverwrite</b> The specified field is static and thus cannot be overwritten by the application.
			<b>overflow</b> The value supplied for the specified fields is too long.
			<b>notFound</b> The specified field does not exist.
			<b>notRead</b> The specified field is not an input field.
			<b>notWrite</b> An attempt was made to write to an input field.
			<b>hwerror</b> The specified field uses special hardware (e.g. OCR, Low/High coercivity, etc) and an error occurred.
			<b>notSupported</b> The form field type is not supported with device.
			<b>graphic</b> The specified graphic image could not be printed.
failure	string		

#### Example Message (generated)

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

## Printer.MediaRejectedEvent

### Description

This event is generated as a result of physical media that is rejected whenever an attempt is made to insert media into the physical device. Rejection of the media will cause the command currently executing to complete with an error, at which point the media should be removed.

The application may use this event to (for example) display a message box on the screen indicating why the media was rejected, and telling the user to remove and reinsert the media.

### Message Header

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

### Message Payload

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
			Specifies the reason for rejecting the media as one of the following values:
		<b>short</b>	The rejected media was too short.
		<b>long</b>	The rejected media was too long.
		<b>multiple</b>	The media was rejected due to insertion of multiple documents.
		<b>align</b>	The media could not be aligned and was rejected.
reason	string	<b>moveToAlign</b>	The media could not be transported to the align area and was rejected.
		<b>shutter</b>	The media was rejected due to the shutter failing to close.
		<b>escrow</b>	The media was rejected due to problems transporting media to the escrow position.
		<b>thick</b>	The rejected media was too thick.
		<b>other</b>	The media was rejected due to a reason other than those listed above.

**Example Message (generated)**

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

---

XFS4IoT specification - Preview version 0.2. Next preview - Dec 2020. Note: work-in-progress. Use at your own risk  
All rights of exploitation in any form and by any means reserved worldwide for CEN national Members.

## XFS4IOT Sample Messaging for Text Terminal Unit Draft 0.0.4

This section describes the functions provided by a generic Text Terminal Unit () service. A Text Terminal Unit is a text i/o device, which applies both to ATM operator panels and to displays incorporated in devices such as pads and printers. This service allows for the following categories of functions:

- Forms oriented input and output
- Direct display output
- Keyboard input
- LED settings and control

All position indexes are zero based, where column zero, row zero is the top-leftmost position. If the device has no shift key, the ReadForm and Read commands will return only upper case letters. If the device has a shift key, these commands return upper and lower case letters as governed by the user's use of the shift key.

### Documentation

## Form and Field Definitions

---

This section outlines the format of the definitions of forms, the fields within them, and the media on which they are printed.

### Definition Syntax

---

The syntactic rules for form, field and media definitions are as follows:

White space	space, tab.
Line continuation	backslash (\).
Line termination	CR, LF, CR/LF; line termination ends a "keyword section" (a keyword and its value[s]).
Keywords	must be all upper case.
Names	(field/media/font names) any case; case is preserved;
d; Service Providers are case sensitive.	
Strings	all strings must be enclosed in double quote characters ("); standard C escape sequences are allowed.
Comments	start with two forward slashes (//); end at line termination.

Other notes:

- If a keyword is present, all its values must be specified; default values are

#### XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

used only if the keyword is absent.

- Values that are character strings are marked with asterisks in the definition s below, and must be quoted as specified above.
- Fields are processed in the sequence they are defined in the form.
- The order of attributes within a form is not mandatory; the attributes may be defined in any order.
- All forms can be represented using either ISO 646 (ANSI) or unicode character encoding. If the unicode representation is used then all Names and Strings are restricted to an internal representation of ISO 646 (ANSI) characters. Only the initialValue keyword values can have double byte values outside of the ISO 646 (ANSI) character set.
- If forms character encoding is unicode then, consistent with the unicode stan dard, the file prefix must be in Little Endian (xFFFE) or Big Endian (xFEFF) no tation, such that unicode encoding is recognized.
- In the form definition file, where characters are expressed using standard C hexadecimal escape sequences, the high order byte is defined first. For exampl e, "\x0041" would represent the character 'A'. This is independent of the enco ding format of the form definition file

### Form/media definition files in multi-vendor environments

Although for most Service Providers directory location and extension of form/media definition files are configurable through the registry, the capabilities of Service Providers and or actual hardware may vary. Therefore the following considerations should be taken into account when applications use form definition files with the purpose of running in a multi-vendor environment:

- Physical display area dimensions may vary from one text terminal to another.
- Just-in-time form loading may not be supported by all Service Providers, whic h makes it impossible to create dynamic form files just before displaying them (which in return means that only the display data of the forms can be changed, not the -layout data such as field positions).
- Some form/media definition keywords may not be supported due to limitations o f the hardware or software.

### Form Definition

<b>FORM</b>	<i>formname*</i>
<b>BEGIN</b>	
(required) <b>SIZE</b>	<i>width</i> <i>height</i> Width of form Height of form
	<i>major,</i>
	<i>minor,</i> Major version number (default 0)
<b>VERSION</b>	<i>date*,</i> Minor version number (default 0) Creation/modification date Author of form <i>author*</i>
	Language used in this form - a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits)
(required) <b>LANGUAGE</b>	<i>languageID</i> language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID)
<b>COPYRIGHT</b>	<i>copyright*</i> Copyright entry
<b>TITLE</b>	<i>title*</i> Title of form
<b>COMMENT</b>	<i>comment*</i> Comment section
[ <b>FIELD</b>	<i>fieldname*</i> One field definition (as defined in the next section) for each field in the form
<b>BEGIN . . .</b>	
<b>END ]</b>	

```
FORM          formname*  
END
```

### Field Definition

---

<b>FIELD</b>	<i>fieldname</i> *	
<b>BEGIN</b>		
<b>LANGUAGE</b>	<i>languageID</i>	Language used for this field. See Form definition for detailed description. If unspecified defaults to form definition Language specification.
(required) <b>POSITION</b>	<i>x, y</i>	Horizontal position (relative to left side of form) Vertical position (relative to top of form) The initial left upper position is referenced as (0,0)
(required) <b>SIZE</b>	<i>width, height</i>	Field width Field height
<b>TYPE</b>	<i>fieldtype</i>	Type of field: Text (default) Invisible Password (contents is echoed with '**') Graphic (ignored for ReadForm commands) Information on how to size the Graphic within the field: BestFit (default) scale to size indicated ASIS render at native size
<b>SCALING</b>	<i>scalingtype</i>	Maintain aspect scale as close as possible to size indicated while maintaining the aspect ratio and not losing Graphic information. SCALING is only relevant for Graphics field types
<b>CLASS</b>	<i>class</i>	Field class: Optional (default) Static Required
<b>KEYS</b>	<i>keys</i>	Accepted input key types: Numeric Hexadecimal Alphanumeric This is an optional field where the default value is vendor dependent.
<b>ACCESS</b>	<i>access</i>	Access rights of field: Write (default) Read ReadWrite
<b>OVERFLOW</b>	<i>overflow</i>	Action on field overflow. Terminate (default) Truncate OverWrite
<b>STYLE</b>	<i>style</i>	Display attributes as a combination of the following, ORed together using the "
<b>HORIZONTAL</b>	<i>justify</i>	Horizontal alignment of field contents: Left (default) Right Center
<b>FORMAT</b>	<i>formatstring</i> *	This is an application defined input field describing how the application should format the data. This may be interpreted by the Service Provider.
<b>INITIALVALUE</b>	<i>value</i> *	Initial value. For Graphic type fields, this value will contain the filename of the Graphic image. The type of this Graphic will be determined by the file extension (e.g. BMP for Windows Bitmap). The Graphic file name must contain the full path. For example "C:\XFS\BSVCLOGO.BMP" illustrates the use of the full path name

```
END
```

## Commands

### TextTerminal.GetStatus

---

#### Description

This command reports the full range of information available, including the information that is provided by the Service Provider.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

## Command Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
common <b>(Required)</b>			Status information common to all XFS4IoT services.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
common.device	string		Specifies the state of the device.
common.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.
common.guideLights	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.
common.guideLights.flashRate	string		Indicates the current flash rate of the guidelight.
common.guideLights.color	string		Indicates the current color of the guidelight.
common.guideLights.direction	string		Indicates the current direction of the guidelight.
common.devicePosition	string		Position of the device.
common.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
common.antiFraudModule	string		Specifies the state of the anti-fraud module
cardReader.			
cardReader.timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
cardReader.			

Name	Type	Default	Description
cardReader.media	string		<p>Specifies the media state of the device as one of the following values. This status is independent of any media in the parking stations.</p> <p><b>notSupported</b>&lt;br&gt; Capability to report media position is not supported by the device (e.g. a typical swipe reader or contactless chip card reader).</p> <p><b>unknown</b>&lt;br&gt; The media state cannot be determined with the device in its current state (e.g. the value of device is noDevice, powerOff, offline or hwerror).</p> <p><b>present</b>&lt;br&gt; Media is present in the device, not in the entering position and not jammed. A card in a parking station is not considered to be present. On the latched dip device, this indicates that the card is present in the device and the card is unlatched.</p> <p><b>notPresent</b>&lt;br&gt; Media is not present in the device and not at the entering position.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device; operator intervention is required.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of a motorized device.</p> <p><b>latched</b>&lt;br&gt; Media is present and latched in a latched dip card unit. This means the card can be used for chip card dialog.</p>
cardReader.retainBin	string		Specifies the state of the retain bin.

Name	Type	Default	Description
cardReader.security	string		<p>Specifies the state of the security unit as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; No security module is available.</p> <p><b>notReady</b>&lt;br&gt; The security module is not ready to process cards or is inoperable.</p> <p><b>notPresent</b>&lt;br&gt; The security module is open and ready to process cards.</p>
cardReader.numberCards	integer		<p>The number of cards retained; applicable only to motor driven card units, for non-motorized card units this value is zero. This value is persistent it is reset to zero by the resetCount command.</p>

Name	Type	Default	Description
cardReader.chipPower	string		<p>Specifies the state of the chip controlled by this service. Depending on the value of capabilities response, this can either be the chip on the currently inserted user card or the chip on a permanently connected chip card. The state of the chip is one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability to report the state of the chip is not supported by the ID card unit device. This value is returned for contactless chip card readers.</p> <p><b>unknown</b>&lt;br&gt; The state of the chip cannot be determined with the device in its current state.</p> <p><b>online</b>&lt;br&gt; The chip is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).</p> <p><b>busy</b>&lt;br&gt; The chip is present, powered on, and busy (unable to process an Execute command at this time).</p> <p><b>poweredOff</b>&lt;br&gt; The chip is present, but powered off (i.e. not contacted).</p> <p><b>noDevice</b>&lt;br&gt; A card is currently present in the device, but has no chip.</p> <p><b>hwerror</b>&lt;br&gt; The chip is present, but inoperable due to a hardware error that prevents it from being used (e.g. MUTE, if there is an unresponsive card in the reader).</p> <p><b>noCard</b>&lt;br&gt; There is no card in the device.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor	object		<p>Status information for XFS4IoT services implementing the CashAcceptor interface. This will be omitted if the CashAcceptor interface is not supported.</p>
cashAcceptor.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty.</li> <li>"full": The intermediate stacker is full. This may also be reported during a cash-in transaction where a limit specified by CashAcceptor.SetCashInLimit has been reached.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>

Name	Type	Default	Description
cashAcceptor.stackerItems	string		<p>This field informs the application whether items on the intermediate stacker have been in customer access. Following values are possible:</p> <ul style="list-style-type: none"> <li>"customerAccess": Items on the intermediate stacker have been in customer access. If the device is a cash recycler then the items on the intermediate stacker may be there as a result of a previous cash-out operation.</li> <li>"noCustomerAccess": Items on the intermediate stacker have not been in customer access.</li> <li>"accessUnknown": It is not known if the items on the intermediate stacker have been in customer access.</li> <li>"noltems": There are no items on the intermediate stacker or the physical device has no intermediate stacker.</li> </ul> <p>Supplies the state of the banknote reader. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The banknote reader is in a good state.</li> <li>"inoperable": The banknote reader is inoperable.</li> <li>"unknown": Due to a hardware error or other condition, the state of the banknote reader cannot be determined.</li> <li>"notSupported": The physical device has no banknote reader.</li> </ul>
cashAcceptor.banknoteReader	string		

Name	Type	Default	Description
cashAcceptor.dropBox	boolean		The drop box is an area within the CashAcceptor where items which have caused a problem during an operation are stored. This field specifies the status of the drop box. TRUE means that some items are stored in the drop box due to a cash-in transaction which caused a problem. FALSE indicates that the drop box is empty.
cashAcceptor.positions	array		<p>Array of structures for each position from which items can be accepted.</p> <p>Supplies the input or output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"inLeft": Left input position.</li> <li>"inRight": Right input position.</li> <li>"inCenter": Center input position.</li> <li>"inTop": Top input position.</li> <li>"inBottom": Bottom input position.</li> <li>"inFront": Front input position.</li> <li>"inRear": Rear input position.</li> <li>"outLeft": Left output position.</li> <li>"outRight": Right output position.</li> <li>"outCenter": Center output position.</li> <li>"outTop": Top output position.</li> <li>"outBottom": Bottom output position.</li> <li>"outFront": Front output position.</li> <li>"outRear": Rear output position.</li> </ul>
cashAcceptor.positions.position	string		

Name	Type	Default	Description
cashAcceptor.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul> <p>The status of the input or output position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>
cashAcceptor.positions.positionStatus	string		

Name	Type	Default	Description
cashAcceptor.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>
cashAcceptor.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous dispense operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashAcceptor.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>
cashAcceptor.mixedMode	string		<p>Reports if Mixed Media mode is active. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notActive": Mixed Media transactions are not supported by the device or Mixed Media mode is not activated.</li> <li>"active": Mixed Media mode using the CashAcceptor and ItemProcessor interfaces is activated.</li> </ul>
cashDispenser	object		<p>Status information for XFS4IoT services implementing the CashDispenser interface. This will be omitted if the CashDispenser interface is not supported.</p>

Name	Type	Default	Description
cashDispenser.intermediateStacker	string		<p>Supplies the state of the intermediate stacker. These bills are typically present on the intermediate stacker as a result of a retract operation or because a dispense has been performed without a subsequent present.</p> <p>Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The intermediate stacker is empty.</li> <li>"notEmpty": The intermediate stacker is not empty. The items have not been in customer access.</li> <li>"notEmptyCustomer": The intermediate stacker is not empty. The items have been in customer access. If the device is a recycler then the items on the intermediate stacker may be there as a result of a previous cash-in operation.</li> <li>"notEmptyUnknown": The intermediate stacker is not empty. It is not known if the items have been in customer access.</li> <li>"unknown": Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</li> <li>"notSupported": The physical device has no intermediate stacker.</li> </ul>
cashDispenser.positions	array		Array of structures for each position to which items can be dispensed or presented.

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashDispenser.positions.position	string		<p>Supplies the output position as one of the following values:</p> <ul style="list-style-type: none"> <li>"left": Left output position.</li> <li>"right": Right output position.</li> <li>"center": Center output position.</li> <li>"top": Top output position.</li> <li>"bottom": Bottom output position.</li> <li>"front": Front output position.</li> <li>"rear": Rear output position.</li> </ul>
cashDispenser.positions.shutter	string		<p>Supplies the state of the shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"closed": The shutter is operational and is closed.</li> <li>"open": The shutter is operational and is open.</li> <li>"jammed": The shutter is jammed and is not operational. The field jammedShutterPosition provides the positional state of the shutter.</li> <li>"unknown": Due to a hardware error or other condition, the state of the shutter cannot be determined.</li> <li>"notSupported": The physical device has no shutter or shutter state reporting is not supported.</li> </ul>

Name	Type	Default	Description
cashDispenser.positions.positionStatus	string		<p>Returns information regarding items which may be at the output position. If the device is a recycler it is possible that the output position will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The output position is empty.</li> <li>"notEmpty": The output position is not empty.</li> <li>"unknown": Due to a hardware error or other condition, the state of the output position cannot be determined.</li> <li>"notSupported": The device is not capable of reporting whether or not items are at the output position.</li> </ul>
cashDispenser.positions.transport	string		<p>Supplies the state of the transport mechanism. The transport is defined as any area leading to or from the position. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": The transport is in a good state.</li> <li>"inoperative": The transport is inoperative due to a hardware failure or media jam.</li> <li>"unknown": Due to a hardware error or other condition the state of the transport cannot be determined.</li> <li>"notSupported": The physical device has no transport or transport state reporting is not supported.</li> </ul>

Name	Type	Default	Description
cashDispenser.positions.transportStatus	string		<p>Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous cash-in operation. Following values are possible:</p> <ul style="list-style-type: none"> <li>"empty": The transport is empty.</li> <li>"notEmpty": The transport is not empty.</li> <li>"notEmptyCustomer": Items which a customer has had access to are on the transport.</li> <li>"notEmptyUnknown": Due to a hardware error or other condition it is not known whether there are items on the transport.</li> <li>"notSupported": The device is not capable of reporting whether items are on the transport.</li> </ul>
cashDispenser.positions.jammedShutterPosition	string		<p>Returns information regarding the position of the jammed shutter. Following values are possible:</p> <ul style="list-style-type: none"> <li>"notSupported": The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.</li> <li>"notJammed": The shutter is not jammed.</li> <li>"open": The shutter is jammed, but fully open.</li> <li>"partiallyOpen": The shutter is jammed, but partially open.</li> <li>"closed": The shutter is jammed, but fully closed.</li> <li>"unknown": The position of the shutter is unknown.</li> </ul>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement	object		<p>Status information for XFS4IoT services implementing the CashManagement interface. This will be omitted if the CashManagement interface is not supported.</p> <p>Supplies the state of the safe door. Following values are possible:</p> <ul style="list-style-type: none"> <li>"doorNotSupported": Physical device has no safe door or safe door state reporting is not supported.</li> <li>"doorOpen": Safe door is open.</li> <li>"doorClosed": Safe door is closed.</li> <li>"doorUnknown": Due to a hardware error or other condition, the state of the safe door cannot be determined.</li> </ul>
cashManagement.safeDoor	string		

Name	Type	Default	Description
cashManagement.dispenser	string		<p>Supplies the state of the logical cash units for dispensing. Following values are possible:</p> <p>"ok": All cash units present are in a good state.</p> <p>"cashUnitState": One or more of the cash units is in a low, empty, inoperative or manipulated condition. Items can still be dispensed from at least one of the cash units.</p> <p>"cashUnitStop": Due to a cash unit failure dispensing is impossible. No items can be dispensed because all of the cash units are in an empty, inoperative or manipulated condition. This state may also occur when a reject/retract cash unit is full or no reject/retract cash unit is present, or when an application lock is set on every cash unit which can be locked.</p> <p>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</p>

**XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.**

Name	Type	Default	Description
cashManagement.acceptor	string		<p>Supplies the state of the cash units for accepting cash. Following values are possible:</p> <ul style="list-style-type: none"> <li>"ok": All cash units present are in a good state.</li> <li>"cashUnitState": One or more of the cash units is in a high, full, inoperative or manipulated condition. Items can still be accepted into at least one of the cash units.</li> <li>"cashUnitStop": Due to a cash unit failure accepting is impossible. No items can be accepted because all of the cash units are in a full, inoperative or manipulated condition. This state may also occur when a retract cash unit is full or no retract cash unit is present, or when an application lock is set on every cash unit, or when Level 2/3 notes are to be automatically retained within cash units, but all of the designated cash units for storing them are full or inoperative.</li> <li>"cashUnitUnknown": Due to a hardware error or other condition, the state of the cash units cannot be determined.</li> </ul>
pinPad			Status information for XFS4IoT services implementing the PinPad interface. This will be omitted if the PinPad interface is not supported.
crypto	object		Status information for XFS4IoT services implementing the Crypto interface. This will be omitted if the Crypto interface is not supported.

Name	Type	Default	Description
keyManagement	object		Status information for XFS4IoT services implementing the KeyManagement interface. This will be omitted if the KeyManagement interface is not supported.
keyboard	object		Status information for XFS4IoT services implementing the Keyboard interface. This will be omitted if the Keyboard interface is not supported.
textTerminal	object		Status information for XFS4IoT services implementing the TextTerminal interface. This will be omitted if the TextTerminal interface is not supported.
printer	object		Status information for XFS4IoT services implementing the Printer interface. This will be omitted if the Printer interface is not supported.

Name	Type	Default	Description
printer.media	string		<p>Specifies the state of the print media (i.e. receipt, statement, passbook, etc.) as one of the following values. This field does not apply to journal printers:</p> <p><b>notSupported</b>&lt;br&gt; The capability to report the state of the print media is not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; The state of the print media cannot be determined with the device in its current state.</p> <p><b>present</b>&lt;br&gt; Media is in the print position, on the stacker or on the transport (i.e. a passbook in the parking station is not considered to be present). On devices with continuous paper supplies, this value is set when paper is under the print head. On devices with no supply or individual sheet supplies, this value is set when paper/media is successfully inserted-loaded.</p> <p><b>notPresent</b>&lt;br&gt; Media is not in the print position or on the stacker.</p> <p><b>jammed</b>&lt;br&gt; Media is jammed in the device.</p> <p><b>entering</b>&lt;br&gt; Media is at the entry/exit slot of the device.</p> <p><b>retracted</b>&lt;br&gt; Media was retracted during the last command which controlled media.</p>

Name	Type	Default	Description
printer.paper	object		<p>Specifies the state of paper supplies as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by the device.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The paper supply is full.</p> <p><b>low</b>&lt;br&gt; The paper supply is low.</p> <p><b>out</b>&lt;br&gt; The paper supply is empty.</p> <p><b>jammed</b>&lt;br&gt; The paper supply is jammed.</p>
printer.paper.upper	string		The state of the upper paper supply.
printer.paper.lower	string		The state of the lower paper supply.
printer.paper.external	string		The state of the external paper supply.
printer.paper.aux	string		The state of the auxiliary paper supply.
printer.paper.aux2	string		The state of the second auxiliary paper supply.
printer.paper.park	string		The state of the parking station paper supply.

Name	Type	Default	Description
printer.toner	string		<p>Specifies the state of the toner or ink supply or the state of the ribbon as one of the following:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of toner or ink supply or the ribbon cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; The toner or ink supply is full or the ribbon is OK.</p> <p><b>low</b>&lt;br&gt; The toner or ink supply is low or the print contrast with a ribbon is weak.</p> <p><b>out</b>&lt;br&gt; The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.</p>
printer.ink	string		<p>Specifies the status of the stamping ink in the printer as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the stamping ink supply cannot be determined with device in its current state.</p> <p><b>full</b>&lt;br&gt; Ink supply in device is full.</p> <p><b>low</b>&lt;br&gt; Ink supply in device is low.</p> <p><b>out</b>&lt;br&gt; Ink supply in device is empty.</p>

Name	Type	Default	Description
printer.lamp	string		<p>Specifies the status of the printer imaging lamp as one of the following values:</p> <p><b>notSupported</b>&lt;br&gt; Capability not supported by device.</p> <p><b>unknown</b>&lt;br&gt; Status of the imaging lamp cannot be determined with device in its current state.</p> <p><b>ok</b>&lt;br&gt; The lamp is OK.</p> <p><b>fading</b>&lt;br&gt; The lamp should be changed.</p> <p><b>inop</b>&lt;br&gt; The lamp is inoperative.</p>
printer.retractBins	array		<p>An array of bin state objects. If no retain bins are supported, the array will be empty.</p> <p>Specifies the state of the printer retract bin as one of the following:</p> <p><b>ok</b>&lt;br&gt; The retract bin of the printer is in a healthy state.</p> <p><b>full</b>&lt;br&gt; The retract bin of the printer is full.</p> <p><b>unknown</b>&lt;br&gt; Status cannot be determined with device in its current state.</p> <p><b>high</b>&lt;br&gt; The retract bin of the printer is nearly full.</p> <p><b>missing</b>&lt;br&gt; The retract bin is missing.</p>
printer.retractBins.count	integer		<p>The number of media retracted to this bin. This value is persistent; it may be reset to zero by the Printer.ResetCount command.</p>
printer.mediaOnStacker	integer		<p>The number of media on stacker; applicable only to printers with stacking capability.</p>

Name	Type	Default	Description
printer.paperType	object		<p>Specifies the type of paper loaded as one of the following:</p> <p><b>unknown</b>&lt;br&gt; No paper is loaded, reporting of this paper type is not supported or the paper type cannot be determined.</p> <p><b>single</b>&lt;br&gt; The paper can be printed on only one side.</p> <p><b>dual</b>&lt;br&gt; The paper can be printed on both sides.</p>
printer.paperType.upper	string		The upper paper supply paper type.
printer.paperType.lower	string		The lower paper supply paper type.
printer.paperType.external	string		The external paper supply paper type.
printer.paperType.aux	string		The auxilillary paper supply paper type.
printer.paperType.aux2	string		The second auxilillary paper supply paper type.
printer.paperType.park	string		The parking station paper supply paper type.
printer.blackMarkMode	string		<p>Specifies the status of the black mark detection and associated functionality:</p> <p><b>notSupported</b>&lt;br&gt; Black mark detection is not supported.</p> <p><b>unknown</b>&lt;br&gt; The status of the black mark detection cannot be determined.</p> <p><b>on</b>&lt;br&gt; Black mark detection and associated functionality is switched on.</p> <p><b>off</b>&lt;br&gt; Black mark detection and associated functionality is switched off.</p>

Name	Type	Default	Description
keyboard <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	string		Specifies the state of the keyboard.
keyLock <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	string		Specifies the state of the keyboard lock.
displaySizeX <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	integer		Specifies the horizontal size of the display of the text terminal unit.
displaySizeY <span style="background-color: #f0f0f0; border: 1px solid black; padding: 2px;">(Required)</span>	integer		Specifies the vertical size of the display of the text terminal unit.
leds	array		Specifies array that specifies the state of each LED. Specifies the state of the na, off or a combination of the following flags consisting of one type B, and optionally one type C
leds.na	boolean		The Status is not available. Type A
leds.off	boolean		The LED is turned off. Type A
leds.slowFlash	boolean		The LED is blinking. Type B
leds.mediumFlash	boolean		The LED is blinking medium frequency. Type B
leds.quickFlash	boolean		The LED is blinking quickly. Type B
leds.continuous	boolean		The LED is turned on continuous(steady). Type B
leds.red	boolean		The LED is red. Type C
leds.green	boolean		The LED is green. Type C
leds.yellow	boolean		The LED is yellow. Type C
leds.blue	boolean		The LED is blue. Type C
leds.cyan	boolean		The LED is cyan. Type C

Name	Type	Default	Description
leds.magenta	boolean		The LED is magenta. Type C
leds.white	boolean		The LED is white. Type C

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "common": {
      "device": "online",
      "extra": [
        "string"
      ],
      "guideLights": [
        {}
      ],
      "devicePosition": "inposition",
      "powerSaveRecoveryTime": 0,
      "antiFraudModule": "notSupp"
    },
    "cardReader": {
      "timeout": "5000",
      "media": "notSupported",
      "retainBin": "notSupported",
      "security": "notSupported",
      "numberCards": 0,
      "chipPower": "notSupported"
    },
    "cashAcceptor": {
      "intermediateStacker": "empty",
      "stackerItems": "customerAccess",
      "banknoteReader": "customokerAccess",
      "dropBox": true,
      "positions": [
        {
          "position": "inLeft",
          "shutter": "closed",
          "positionStatus": "empty",
          "transport": "ok",
          "transportStatus": "empty",
          "jammedShutterPosition": "notSupported"
        }
      ],
      "mixedMode": "notActive"
    },
    "cashDispenser": {
      "intermediateStacker": "empty",
      "positions": [
        {
          "position": "left",
          "shutter": "closed",
          "positionStatus": "empty",
          "transport": "ok",
          "transportStatus": "empty"
        }
      ]
    }
  }
}
```

```
        "shutter": "closed",
        "positionStatus": "empty",
        "transport": "ok",
        "transportStatus": "empty",
        "jammedShutterPosition": "notSupported"
    }
]
},
"cashManagement": {
    "safeDoor": "doorNotSupported",
    "dispenser": "ok",
    "acceptor": "ok"
},
"pinPad": null,
"crypto": {},
"keyManagement": {},
"keyboard": "on",
"textTerminal": {},
"printer": {
    "media": "notSupported",
    "paper": {
        "upper": "notSupported",
        "lower": "notSupported",
        "external": "notSupported",
        "aux": "notSupported",
        "aux2": "notSupported",
        "park": "notSupported",
        "property1": "notSupported",
        "property2": "notSupported"
    },
    "toner": "notSupported",
    "ink": "notSupported",
    "lamp": "notSupported",
    "retractBins": [
        {
            "state": "unknown",
            "count": 0
        }
    ],
    "mediaOnStacker": 0,
    "paperType": {
        "upper": "unknown",
        "lower": "unknown",
        "external": "unknown",
        "aux": "unknown",
        "aux2": "unknown",
        "park": "unknown",
        "property1": "unknown",
        "property2": "unknown"
    },
    "blackMarkMode": "notSupported"
},
"keyLock": "on",
"displaySizeX": 0,
"displaySizeY": 0,
"leds": [
{
    "na": true,
    "off": true,
    "on": true
}
]
```

```
        "slowFlash": true,
        "mediumFlash": true,
        "quickFlash": true,
        "continuous": true,
        "red": true,
        "green": true,
        "yellow": true,
        "blue": true,
        "cyan": true,
        "magenta": true,
        "white": true
    }
]
}
}
```

## Event Messages

---

## TextTerminal.GetFormList

---

### Description

This command is used to retrieve the list of forms available on the device.

### Command Message

#### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
formList <b>(Required)</b>	array		Array of the form names.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "formList": [
      "string"
    ]
  }
}
```

## Event Messages

---

## TextTerminal.GetQueryForm

### Description

This command is used to retrieve details of the definition of a specified form.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <small>(Required)</small>	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 <small>(Required)</small>	string		The message type, either command, response, event or completion.
name <small>(Required)</small>	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.
formName <small>(Required)</small>	string		Contains the form name on which to retrieve details.

#### Example Message (generated)

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

### Completion Message

#### Message Header

Name	Type	Default	Description

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
formName <b>(Required)</b>	object		Specifies the null-terminated name of the form.
width <b>(Required)</b>	string		Specifies the width of the form in columns.
height <b>(Required)</b>	string		Specifies the height of the form in rows.
versionMajor <b>(Required)</b>	string		Specifies the major version. If version is not specified in the form then zero is returned.
versionMinor <b>(Required)</b>	string		Specifies the minor version. If the version is not specified in the form then zero is returned.
charSupport <b>(Required)</b>	string		A single flag indicating whether the form is encoded in ascii or unicode.
fields <b>(Required)</b>	array		Object to a list of the field names.
languageId <b>(Required)</b>	string		Specifies the language identifier for the form.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "formName": {},
    "width": "string",
    "height": "string",
    "versionMajor": "string",
    "versionMinor": "string",
    "charSupport": "ascii",
    "fields": [
      "string"
    ],
    "languageId": "string"
  }
}
```

## Event Messages

---

## TextTerminal.GetQueryField

---

### Description

This command is used to retrieve details of the definition of a single or all fields on a specified form.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
formName <b>(Required)</b>	string		Specifies the form name
fieldName	string		Specifies the name of the field about which to retrieve details. If this value is not set, then retrieve details for all fields on the form.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
fields <b>(Required)</b>	array		Array of Fields.
fields.fieldName	string		Specifies the field name.
fields.type	string		Specifies the type of field.
fields.class	string		Specifies the class of the field.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
fields.access	string		Specifies whether the field is to be used for input, output or both.
fields.access.read	string		The Field is used for input from the physical device.
fields.access.write	string		The Field is used for output to the physical device.
fields.overflow	string		Specifies how an overflow of field data should be handled.
fields.format	string		Format string as defined in the form for this field.
fields.languageId	string		Specifies the language identifier for the field.

#### Example Message (generated)

```
{  
  "headers": {  
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",  
    "type": "command",  
    "name": "string"  
  },  
  "payload": {  
    "completionCode": "success",  
    "errorDescription": "string",  
    "fields": [  
      {  
        "fieldName": "string",  
        "type": "text",  
        "class": "static",  
        "access": "string",  
        "overflow": "terminate",  
        "format": "string",  
        "languageId": "string"  
      }  
    ]  
  }  
}
```

## Event Messages

---

### TextTerminal.GetKeyDetail

---

#### Description

This command returns information about the Keys (buttons) supported by the device. This command should be issued to determine which Keys are available.

#### Command Message

##### Message Header

---

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

#### Message Payload

Name	Type	Default	Description
keys	string		String which holds the printable characters (numeric and alphanumeric keys) on the Text Terminal Unit, e.g. "0123456789ABCabc" if those text terminal input keys are present. This field is not set if no keys of this type are present on the device.

Name	Type	Default	Description
commandKeys	array		Array of command keys on the Text Terminal Unit.
commandKeys.ckEnter	boolean	false	
commandKeys.ckCancel	boolean	false	
commandKeys.ckClear	boolean	false	
commandKeys.ckBackspace	boolean	false	
commandKeys.ckHelp	boolean	false	
commandKeys.ck00	boolean	false	
commandKeys.ck000	boolean	false	
commandKeys.ckArrowUp	boolean	false	
commandKeys.ckArrowDown	boolean	false	
commandKeys.ckArrowLeft	boolean	false	
commandKeys.ckArrowRight	boolean	false	
commandKeys.ckOEM1	boolean	false	
commandKeys.ckOEM2	boolean	false	
commandKeys.ckOEM3	boolean	false	
commandKeys.ckOEM4	boolean	false	
commandKeys.ckOEM5	boolean	false	
commandKeys.ckOEM6	boolean	false	
commandKeys.ckOEM7	boolean	false	
commandKeys.ckOEM8	boolean	false	
commandKeys.ckOEM9	boolean	false	
commandKeys.ckOEM10	boolean	false	
commandKeys.ckOEM11	boolean	false	
commandKeys.ckOEM12	boolean	false	
commandKeys.ckFDK01	boolean	false	
commandKeys.ckFDK02	boolean	false	
commandKeys.ckFDK03	boolean	false	
commandKeys.ckFDK04	boolean	false	
commandKeys.ckFDK05	boolean	false	
commandKeys.ckFDK06	boolean	false	
commandKeys.ckFDK07	boolean	false	
commandKeys.ckFDK08	boolean	false	
commandKeys.ckFDK09	boolean	false	
commandKeys.ckFDK10	boolean	false	
commandKeys.ckFDK11	boolean	false	
commandKeys.ckFDK12	boolean	false	
commandKeys.ckFDK13	boolean	false	
commandKeys.ckFDK14	boolean	false	
commandKeys.ckFDK15	boolean	false	
commandKeys.ckFDK16	boolean	false	
commandKeys.ckFDK17	boolean	false	
commandKeys.ckFDK18	boolean	false	
commandKeys.ckFDK19	boolean	false	
commandKeys.ckFDK20	boolean	false	
commandKeys.ckFDK21	boolean	false	
commandKeys.ckFDK22	boolean	false	
commandKeys.ckFDK23	boolean	false	
commandKeys.ckFDK24	boolean	false	
commandKeys.ckFDK25	boolean	false	
commandKeys.ckFDK26	boolean	false	
commandKeys.ckFDK27	boolean	false	

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default
commandKeys.ckFDK28	boolean	false
commandKeys.ckFDK29	boolean	false
commandKeys.ckFDK30	boolean	false
commandKeys.ckFDK31	boolean	false
commandKeys.ckFDK32	boolean	false

#### Example Message (generated)

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

## Event Messages

---

## TextTerminal.Beep

---

### Description

This command is used to beep at the text terminal unit.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
beep <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>			Specifies whether the beeper should be turned on or off.
beep.off	boolean		The beeper is turned off.
beep.keyPress	boolean		The beeper sounds a key click signal.
beep.exclamation	boolean		The beeper sounds an exclamation signal.
beep.warning	boolean		The beeper sounds a warning signal.
beep.error	boolean		The beeper sounds a error signal.
beep.critical	boolean		The beeper sounds a critical error signal.
beep.continuous	boolean		The beeper sound is turned on continuously.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "beep": {
      "off": true,
      "keyPress": true,
      "exclamation": true,
      "warning": true,
      "error": true,
      "critical": true,
      "continuous": true
    }
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
requestId <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	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 <span style="background-color: #cccccc; border: 1px solid black; padding: 2px;">(Required)</span>	string		The message type, either command, response, event or completion.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
name	(Required) string		The original message name, for example "CardReader.Status"

#### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

### Event Messages

---

## TextTerminal.ClearScreen

---

#### Description

This command clears the specified area of the text terminal unit screen. The cursor is positioned to the upper left corner of the cleared area.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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	string	(Required)	The message type, either command, response, event or completion.
name	string	(Required)	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.
positionX <b>(Required)</b>	integer		Specifies the horizontal position of the area to be cleared.
positionY <b>(Required)</b>	integer		Specifies the vertical position of the area to be cleared.
width <b>(Required)</b>	integer		Specifies the width position of the area to be cleared.
height <b>(Required)</b>	integer		Specifies the height position of the area to be cleared.

#### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.  
Example Message (generated)

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

## Event Messages

---

### TextTerminal.DispLight

---

#### Description

This command is used to switch the lighting of the text terminal unit on or off.

#### Command Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
mode <b>(Required)</b>	boolean		Specifies whether the lighting of the text terminal unit is switched on (TRUE) or off (FALSE).

### Example Message (generated)

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

## Event Messages

## TextTerminal.SetLed

---

### Description

This command is used to set the status of the LEDs.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
led <b>(Required)</b>	integer		Specifies the index array as reported in Capabilities) of the LED to set as one of the values defined within the capabilities section.
command <b>(Required)</b>	object		Specifies off type A or a combination of the following flags consisting of one type B, and optionally one type C. If no value of type C is specified then the default color is used. The Service Provider determines which color is used as the default color.
command.off	boolean		The LED is turned off. Type A
command.slowFlash	boolean		The LED is set to flash slowly. Type B
command.mediumFlash	boolean		The LED is set to flash medium frequency. Type B
command.quickFlash	boolean		The LED is set to flash quickly. Type B
command.continuous	boolean		The LED is turned on continuously(steady). Type B
command.red	boolean		The LED color is set to red. Type C

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

Name	Type	Default	Description
command.green	boolean		The LED color is set to green. Type C
command.yellow	boolean		The LED color is set to yellow. Type C
command.blue	boolean		The LED color is set to blue. Type C
command.cyan	boolean		The LED color is set to cyan. Type C
command.magenta	boolean		The LED color is set to magenta. Type C
command.white	boolean		The LED is set to white. Type C

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "led": 0,
    "command": {
      "off": true,
      "slowFlash": true,
      "mediumFlash": true,
      "quickFlash": true,
      "continuous": true,
      "red": true,
      "green": true,
      "yellow": true,
      "blue": true,
      "cyan": true,
      "magenta": true,
      "white": true
    }
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.

Name	Type	Default	Description
name <b>(Required)</b>	string		The original message name, for example "CardReader.Status"

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

### TextTerminal.SetResolution

---

#### Description

This command is used to set the resolution of the display. The screen is cleared and the cursor is positioned at the upper left position.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
resolution <b>(Required)</b>	object		Specifies the horizontal size of the display of the text terminal unit.
resolution.sizeXinteger			Specifies the horizontal size of the display of the text terminal unit (the number of columns that can be displayed).
resolution.sizeYinteger			Specifies the vertical size of the display of the text terminal unit (the number of rows that can be displayed).

#### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

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

## Event Messages

---

### TextTerminal.WriteForm

---

#### Description

This command is used to display a form by merging the supplied variable field data with the defined form and field data specified in the form.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		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.
formName	string <b>(Required)</b>		Form name.
clearScreen	boolean <b>(Required)</b>		Specifies whether the screen is cleared before displaying the form (TRUE) or not (FALSE).

Name	Type	Default	Description
fields <b>(Required)</b>	array		Specifies "<fieldName>=<fieldValue>" string. e.g. Field1=123. The <fieldValue> stands for a string containing all the printable characters (numeric and alphanumeric) to display on the text terminal unit key pad for this field.

#### Example Message (generated)

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

#### Completion Message

##### Message Header

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

##### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

#### Example Message (generated)

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

## Event Messages

---

### TextTerminal.ReadForm

---

#### Description

This command is used to read data from input fields on the specified form.

#### Command Message

##### Message Header

Name	Type	Default	Description
requestId	string <b>(Required)</b>		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	string <b>(Required)</b>		The message type, either command, response, event or completion.
name	string <b>(Required)</b>		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.
formName	string <b>(Required)</b>		Specifies the null-terminated name of the form
fieldNames	array		Specifies the field names from which to read input data. The fields are edited by the user in the order that the fields are specified within this parameter. If fieldNames value is not set, then data is read from all input fields on the form in the order they appear in the form file (independent of the field screen position).

##### Example Message (generated)

---

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information
fields <b>(Required)</b>	array		Specifies "<fieldName>=<fieldValue>" string. e.g. Field1=123. The <fieldValue> stands for a string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad for this field.

### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completionCode": "success",
    "errorDescription": "string",
    "fields": [
      "string"
    ]
  }
}
```

## Event Messages

---

## TextTerminal.Write

---

### Description

This command displays the specified text on the display of the text terminal unit. The specified text may include the control characters CR (Carriage Return) and LF (Line Feed). The control characters can be included in the text as CR, or LF, or CR LF, or LF CR and all combinations will perform the function of relocating the cursor position to the left hand side of the display on the next line down. If the text will overwrite the display area then the display will scroll.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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 <b>(Required)</b>	string		Specifies whether the position of the output is absolute or relative to the current cursor position.

Name	Type	Default	Description
posX <b>(Required)</b>	integer		If mode is set to absolute, this specifies the absolute horizontal position. If mode is set to relative this specifies a horizontal offset relative to the current cursor position as a zero (0) based value.
posY <b>(Required)</b>	integer		If mode is set to absolute, this specifies the absolute vertical position. If mode is set to relative this specifies a vertical offset relative to the current cursor position as a zero (0) based value.
textAttr <b>(Required)</b>	object		Specifies the text attributes used for displaying the text. If none of the following attribute flags are selected then the text will be displayed as normal text.
textAttr.underline boolean			The displayed text will be unlined.
textAttr.inverted boolean			The displayed text will be inverted.
textAttr.flash boolean			The displayed text will be flashing.
text <b>(Required)</b> string			Specifies the text that will be displayed.

#### Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "mode": "relative",
    "posX": 0,
    "posY": 0,
    "textAttr": {
      "underline": true,
      "inverted": true,
      "flash": true
    },
    "text": "string"
  }
}
```

#### Completion Message

##### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b> 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.

XFS4IoT specification - Preview version 0.3. Next preview - Q1 2021. Note: work-in-progress. Use at your own risk.

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

## TextTerminal.Read

---

### Description

This command activates the keyboard of the text terminal unit for input of the specified number of characters. Depending on the specified flush mode the input buffer is cleared. During this command, pressing an active key results in a Key event containing the key details. On completion of the command (when the maximum number of keys have been pressed or a terminator key is pressed), the entered string, as interpreted by the Service Provider, is returned. The Service Provider takes command keys into account when interpreting the data.

### Command Message

#### Message Header

Name	Type	Default	Description
requestId	string	(Required)	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
numOfChars <b>(Required)</b>	integer		Specifies the number of printable characters (numeric and alphanumeric keys) that will be read from the text terminal unit key pad. All command keys like ckEnter, ckFDK01 will not be counted.
mode <b>(Required)</b>	string		Specifies where the cursor is positioned for the read operation.
posX <b>(Required)</b>	integer		If mode is set to absolute, this specifies the absolute horizontal position. If mode is set to relative this specifies a horizontal offset relative to the current cursor position as a zero (0) based value.
posY <b>(Required)</b>	integer		If mode is set to absolute, this specifies the absolute vertical position. If mode is set to relative this specifies a vertical offset relative to the current cursor position as a zero (0) based value.
echoMode <b>(Required)</b>	string		Specifies how the user input is echoed to the screen.
echoAttr <b>(Required)</b>	object		Specifies the text attributes with which the user input is echoed to the screen. If none of the following attribute flags are selected then the text will be displayed as normal text.
echoAttr.underline	boolean		The displayed text will be underlined.
echoAttr.inverted	boolean		The displayed text will be inverted.
echoAttr.flash	boolean		The displayed text will be flashing.
echo <b>(Required)</b>	boolean		Specifies whether the cursor is visible(TRUE) or invisible(FALSE).
flush <b>(Required)</b>	boolean		Specifies whether the keyboard input buffer is cleared before allowing for user input(TRUE) or not (FALSE).

Name	Type	Default	Description
autoEnd <span style="background-color: #f0f0f0; padding: 2px;">(Required)</span>	boolean		Specifies whether the command input is automatically ended by Service Provider if the maximum number of printable characters as specified with numOfChars is entered.
activeKeys	string		String which specifies the numeric and alphanumeric keys on the Text Terminal Unit, e.g. "12ABab", to be active during the execution of the command. Devices having a shift key interpret this parameter differently from those that do not have a shift key. For devices having a shift key, specifying only the upper case of a particular letter enables both upper and lower case of that key, but the device converts lower case letters to upper case in the output parameter. To enable both upper and lower case keys, and have both upper and lower case letters returned, specify both the upper and lower case of the letter (e.g. "12AaBb"). For devices not having a shift key, specifying either the upper case only (e.g. "12AB"), or specifying both the upper and lower case of a particular letter (e.g. "12AaBb"), enables that key and causes the device to return the upper case of the letter in the output parameter. For both types of device, specifying only lower case letters (e.g. "12ab") produces a key invalid error. This parameter is a NULL if no keys of this type are active keys. activeKeys and activeUnicodeKeys are mutually exclusive, so activeKeys field must not be set if activeUnicodeKeys field is not set.
activeCommandKeys	array		Array specifying the command keys which are active during the execution of the command. The array is terminated with a zero value and this array is not set if no keys of this type are active keys.
terminateCommandKeys	array		Array specifying the command keys which must terminate the execution of the command. The array is terminated with a zero value and this array is not set if no keys of this type are terminate keys.

**Example Message (generated)**

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "numOfChars": 0,
    "mode": "relative",
    "posX": 0,
    "posY": 0,
    "echoMode": "text",
    "echoAttr": {
      "underline": true,
      "inverted": true,
      "flash": true
    },
    "echo": true,
    "flush": true,
    "autoEnd": true,
    "activeKeys": "string",
    "activeCommandKeys": [
      "string"
    ],
    "terminateCommandKeys": [
      "string"
    ]
  }
}
```

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

Name	Type	Default	Description
input <b>(Required)</b>	string		Specifies a zero terminated string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad.

#### Example Message (generated)

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

## Event Messages

---

### TextTerminal.Reset

---

#### Description

Sends a service reset to the Service Provider. This command clears the screen, clears the keyboard buffer, sets the default resolution and sets the cursor position to the upper left.

#### Command Message

##### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Event Messages

---

## TextTerminal.DefineKeys

---

## Description

This command defines the keys that will be active during the next ReadForm command. The configured set will be active until the next ReadForm command ends, at which point the default values are restored.

## Command Message

### Message Header

Name	Type	Default	Description
requestId <b>(Required)</b>	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 <b>(Required)</b>	string		The message type, either command, response, event or completion.
name <b>(Required)</b>	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.
activeKeys <b>(Required)</b>	string		String which specifies the alphanumeric keys on the Text Terminal Unit, e.g. "12ABab", to be active during the execution of the next ReadForm command. Devices having a shift key interpret this parameter differently from those that do not have a shift key. For devices having a shift key, specifying only the upper case of a particular letter enables both upper and lower case of that key, but the device converts lower case letters to upper case in the output parameter. To enable both upper and lower case keys, and have both upper and lower case letters returned, specify both the upper and lower case of the letter (e.g. "12AaBb"). For devices not having a shift key, specifying either the upper case only (e.g. "12AB"), or specifying both the upper and lower case of a particular letter (e.g. "12AaBb"), enables that key and causes the device to return the uppercase of the letter in the output parameter. For both types of device, specifying only lower case letters (e.g. "12ab") produces a key invalid error.
activeCommandKeys	array		Array specifying the command keys which are active during the execution of the next ReadForm command.
terminateCommandKeys	array		Array specifying the command keys which must terminate the execution of the next ReadForm command.

### Example Message (generated)

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

## Completion Message

### Message Header

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

### Message Payload

Name	Type	Default	Description
completionCode	string		success if the command was successful otherwise error
errorDescription	string		If not success, then this is optional vendor dependent information to provide additional information

### Example Message (generated)

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

## Unsolicited Events

### TextTerminal.FieldErrorEvent

---

#### Description

This event specifies that a fatal error has occurred while processing a field.

#### Message Header

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

#### Message Payload

Name	Type	Default	Description
formName <b>(Required)</b>	string		Specifies the form name.
fieldName <b>(Required)</b>	string		Specifies the field name.
failure <b>(Required)</b>	string		Specifies the type of failure.

#### Example Message (generated)

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

## TextTerminal.FieldWarningEvent

---

### Description

This event is used to specify that a non-fatal error has occurred while processing a field.

### Message Header

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

### Example Message (generated)

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

---

## TextTerminal.KeyEvent

---

### Description

This event specifies that any active key has been pressed at the TTU during the Read command. In addition to giving the application more details about individual key presses this information may also be used if the device has no internal display unit and the application has to manage the display of the entered digits.

### Message Header

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

---

### Message Payload

---

Name	Type	Default	Description
key <b>(Required)</b>	string		On a numeric or alphanumeric key press this parameter holds the value of the key pressed. This value is not set if no numeric or alphanumeric key was pressed.
commandKey <b>(Required)</b>	string		On a Command key press this parameter holds the value of the Command key pressed, e.g. ckEnter. This value is not set when no command key was pressed.

**Example Message (generated)**

XFS4  
risk  
All rig

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

---

## Events