Bundesamt
für Sicherheit in der
Informationstechnik

Technical Guideline TR-03112-6

eCard-API-Framework – IFD-Interface

Version 1.1.5

7. April 2015

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn

E-Mail: ecard.api@bsi.bund.de
Internet: https://www.bsi.bund.de

Contents

Table of Figures

1 Overview of the eCard-API-Framework

The objective of the eCard-API-Framework is the provision of a simple and homogeneous interface to enable standardised use of the various smart cards (eCards) for different applications.

The eCard-API-Framework is sub-divided into the following layers:

- Application-Layer
- Identity-Layer
- Service-Access-Layer
- Terminal-Layer

The **Application-Layer** contains the various applications which use the eCard-API-Framework to access the eCards and their associated functions. Application-specific "convenience interfaces", in which the recurring invocation sequences may be encapsulated in application-specific calls, may also exist in this layer. However, these interfaces are currently *not* within the scope of the e-Card-API-framework.

The **Identity-Layer** comprises the eCard-Interface and the Management interface, and therefore functions for the use and management of electronic identities as well as for management of the eCard-API-Framework.

The *eCard-Interface* (refer to [TR-03112-2]) allows to request certificates as well as the encryption, signature and time-stamping of documents.

In the M*anagement-Interface* (refer to [TR-03112-3]), functions for updating the framework and the management of trusted identities, smart cards, card terminals, and default behaviour are available.

The **Service-Access-Layer** provides, in particular, functions for cryptographic primitives and biometric mechanisms in connection with cryptographic tokens, and comprises the ISO24727-3-Interface and the Support-Interface.

The *ISO24727-3-Interface* defined in the present document is a webservice-based implementation of the standard of the same name [ISO24727-3]. This interface contains functions to establish (cryptographically protected) connections to smart cards, to manage card applications, to read or write data, to perform cryptographic operations and to manage the respective key material (in the form of so-called "differential identities"). In the process, all functions which use or manage "differential identities" are parameterised by means of protocol-specific object identifiers so that the different protocols which are defined in the present document MAY be used with a standardised interface (refer to [TR-03112-7]).

The S*upport-Interface* (refer to [TR-03112-5]) contains a range of supporting functions.

The **Terminal-Layer** primarily contains the *IFD-Interface* (refer to [TR-03112-6]). This layer takes over the generalisation of specific card terminal types and various interfaces as well as communication with the smart card. For the user it is unimportant whether the card is addressed by PC/SC, a SICCT terminal or a proprietary interface, or whether it has contacts or is contact-less.

1.1 Key Words

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]. The key word “CONDITIONAL” is to be interpreted as follows:

CONDITIONAL: The usage of an item is dependent on the usage of other items. It is therefore further qualified under which conditions the item is REQUIRED or RECOMMENDED.

1.2 XML-Schema

A XML-Schema is provided together with this Technical Guideline. In case of incongruencies, the specifications in this text take precedence. The graphical representations of the XML-Schema illustrate the schema. Note that the text of this Guideline might further restrict the presence or mulitplicity of elements as compared to the schema definition.

2 Overview of the IFD-Interface

2.1 Objective

The IFD-Interface generalises the specific card terminal types and various interfaces and communicates with the smart card. For the user it is not relevant whether the card is addressed by PC/SC, CT-API, in a SICCT card terminal or via a proprietary interface, or whether it has contacts or is contactless.

2.2 Functions

The IFD-Interface provides the following function groups:

- Card terminal functions
- Card functions
- User interaction functions

In addition, there is an IFD-Callback-Interface for card terminal events and additional functions for the management of card terminals which are specified in the management interface [TR-03112-3]:

- With the `RegisterIFD` function it is possible to add a card terminal with all configuration information.
- The `UnregisterIFD` function deletes a card terminal.

2.2.1 Card terminal functions

- The `EstablishContext` function opens a session with the Terminal-Layer and returns a `ContextHandle` which is used to address this session during subsequent function invocations.
- The `ReleaseContext` terminates a session with the Terminal-Layer which had been addressed by a `ContextHandle`.
- With the `ListIFDs` function a list of available card terminals is returned to the calling layer.
- The `GetIFDCapabilities` function returns information on a specific card terminal and its functional units to the calling layer.
- The `GetStatus` function determines the current status of the card terminal.
- With the `Wait` function the invoking layer can be informed about card terminal events by the return of the wait function or by means of the `SignalEvent` callback function.
- The `Cancel` function terminates waiting for card terminal events or attempts to terminate processing of the last command sent via the current handle to a specific card terminal. In this case the success of the operation depends on the type of command and the time at which `Cancel` was invoked.
- The `ControlIFD` function sends a (proprietary) command to the card terminal. This serves to permit access to proprietary and application-specific functions for which there is no separate command in the IFD-Interface without changing the interface.

2.2.2 Card functions

- The `Connect` function activates a card captured by the IFD and returns a `SlotHandle` with which the card can be addressed in the future.
- The `Disconnect` function invalidates a `SlotHandle` and optionally performs an additional action (e.g. eCard ejection, if the IFD features the corresponding mechanical functionality).
- The `BeginTransaction` function starts a transaction within the framework of which several commands can be sent to the eCard. If an error occurs, the transaction is terminated and any modifications are reset.
- The `EndTransaction` function terminates an existing transaction.
- The `Transmit` function sends APDUs to an eCard addressed via a `SlotHandle`.

2.2.3 User interaction functions

- The `VerifyUser` function verifies the user by means of a PIN or a biometric characteristic.
- The `ModifyVerificationData` modifies the identification data (PIN or biometric characteristic).
- The `Output` function may be used to control the output units of a card terminal.

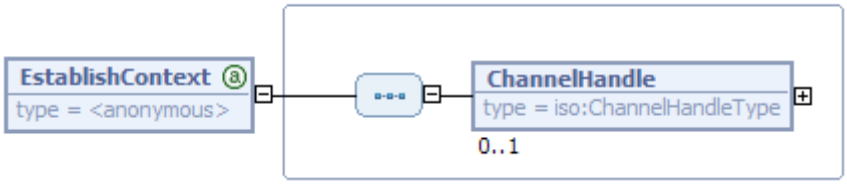
2.2.4 IFD callback interface for card terminal events

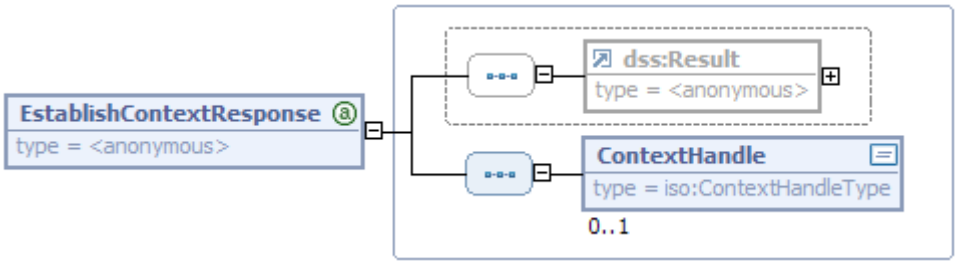
- With the `SignalEvent` function, layers above the Terminal-Layer can be informed of card terminal events. For this purpose the `SignalEvent` function must be offered as a webservice by these layers.

3 Specification of the IFD-Interface

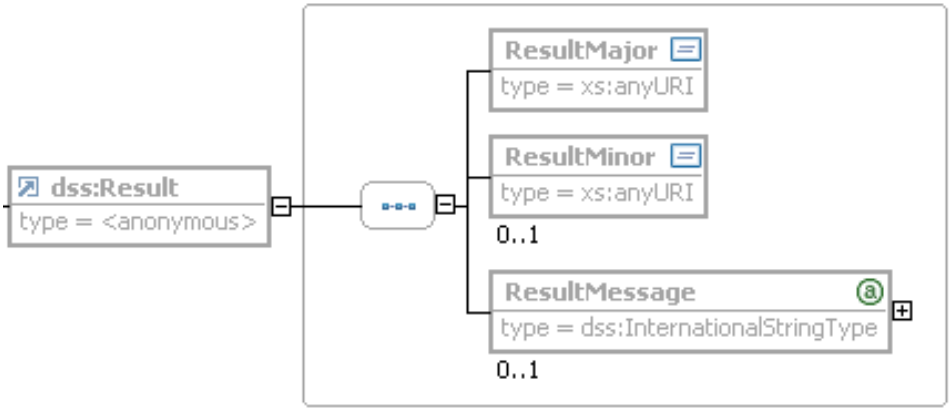
3.1 Card terminal functions

3.1.1 EstablishContext

<table>
<tr><td>Name</td><td colspan="2"><code>EstablishContext</code></td></tr>
<tr><td>Description</td><td colspan="2">The <code>EstablishContext</code> function opens a session with the Terminal-Layer and returns a <code>ContextHandle</code> which is used to address this session in further function invocations.</td></tr>
<tr><td rowspan="3">Invocation parameters</td><td colspan="2"> 

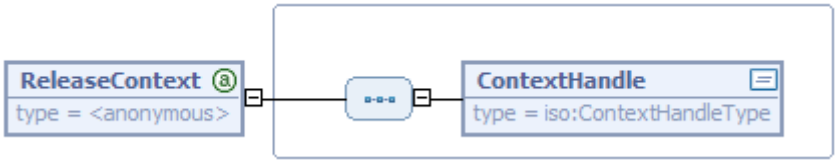
Invocation of the <code>EstablishContext</code> function.</td></tr>
<tr><td>Name</td><td>Description</td></tr>
<tr><td><code>ChannelHandle</code></td><td>Optional parameter with which a remote system can be addressed (also refer to <code>CardApplicationPath</code> in [TR-03112-4]). If the local system is to be addressed, this parameter MAY be omitted.</td></tr>
<tr><td rowspan="3">Return</td><td colspan="2"> 

Return of the <code>EstablishContext</code> function.</td></tr>
<tr><td>Name</td><td>Description</td></tr>
<tr><td><code>dss:Result</code></td><td>Contains the status information and the errors of an executed action. This element is described in more detail below.</td></tr>
<tr><td><code>ContextHandle</code></td><td>The session with the Terminal-Layer is addressed via the returned <code>ContextHandle</code>.</td></tr>
</table>

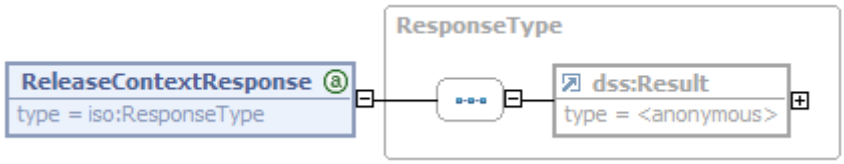
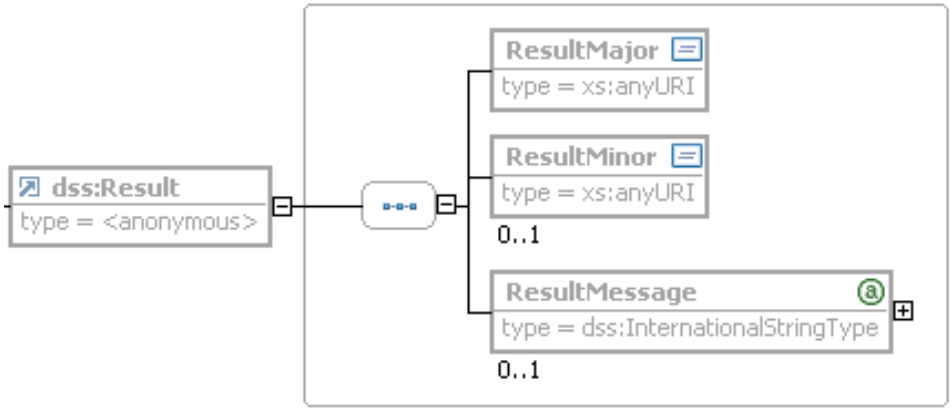
<table>
<tr><td></td><td>


Status information and errors in <code>EstablishContext</code>
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/ifdl/common#unknownSessionIdentifier
• /resultminor/dp#invalidChannelHandle
• /resultminor/al/common#unknownError</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Preconditions</td><td></td></tr>
<tr><td>Postconditions</td><td>A context is established with the IFD-Layer via which commands are sent to card terminals and connections to cards can be established.</td></tr>
<tr><td>Notes</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the <code>EstablishContext</code> function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.1.2).</td></tr>
</table>

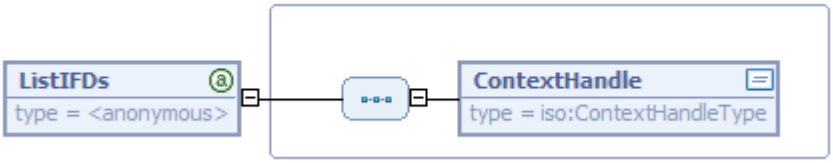
3.1.2 ReleaseContext

<table>
<tr><td>Name</td><td><code>ReleaseContext</code></td></tr>
<tr><td>Description</td><td>The <code>ReleaseContext</code> function terminates a session with the Terminal-Layer.</td></tr>
<tr><td>Invocation parameters</td><td>


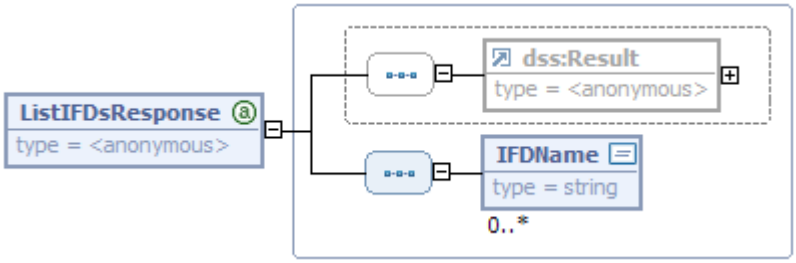
Invocation of the <code>ReleaseContext</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
</table>
</td></tr>
</table>

	`ContextHandle`	Handle with which the session with the Terminal-Layer can be addressed.
Return	 Return of the `ReleaseContext` function.	
	Name	**Description**
	`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
	 Status information and errors in `ReleaseContext`	
	Name	**Error codes**
	`ResultMajor`	• /resultmajor#ok • /resultmajor#error
	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/dp#invalidChannelHandle • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	A context was established with `EstablishContext`.	
Postconditions	The `ContextHandle` loses its validity.	
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `ReleaseContext` function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.1.2).	

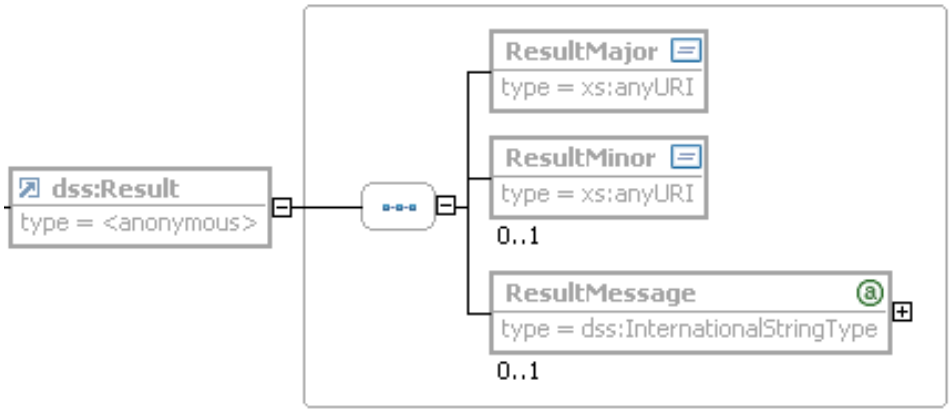
3.1.3 ListIFDs

Name	`ListIFDs`
Description	Returns the list of the card terminals assigned to the eCard-API-Framework, which at least contains the currently connected IFDs.
Invocation parameters	 Invocation of the `ListIFDs` function.

Name	Description
`ContextHandle`	Handle with which the session with the Terminal-Layer can be addressed.

Return	 Return of the `ListIFDs` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`IFDName`	Unique name of the card terminal.

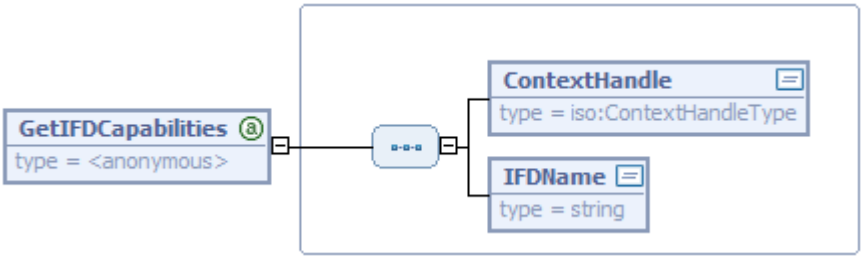


Status information and errors in `ListIFDs`

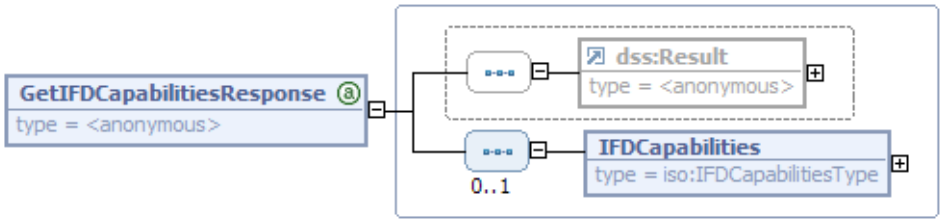
Name	Error codes

	`ResultMajor`	• /resultmajor#ok • /resultmajor#error
	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/dp#invalidChannelHandle • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	The potentially available card terminals were assigned to the eCard-API-Framework in an administrative operation (also refer to [TR-03112-3]). A context was established with `EstablishContext`.	
Postconditions	The status of the card terminals remains unchanged.	
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `ListReaders` function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.3.2). Also refer to `GetCardTerminals` in [TR-03112-2].	

3.1.4 GetIFDCapabilities

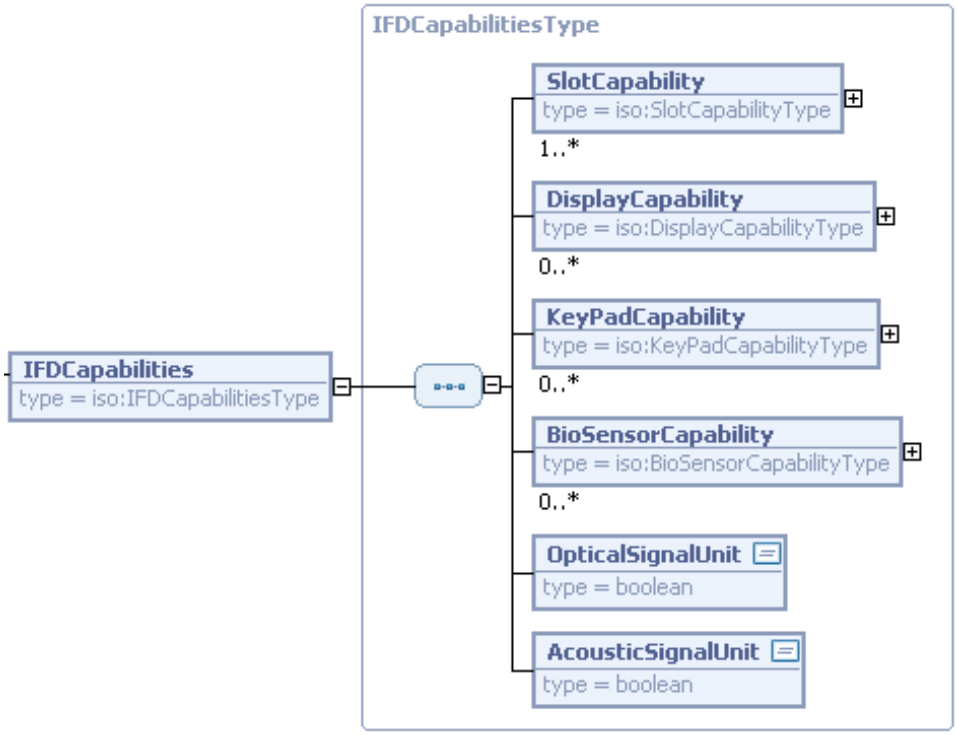
Name	**`GetIFDCapabilities`**	
Description	Returns information on the capabilities of a specific card terminal and its functional units.	
Invocation parameters	 Invocation of the `GetIFDCapabilities` function.	
	Name	**Description**
	ContextHandle	Handle with which the session with the Terminal-Layer is addressed.
	`IFDName`	Unique name of the card terminal.

Return



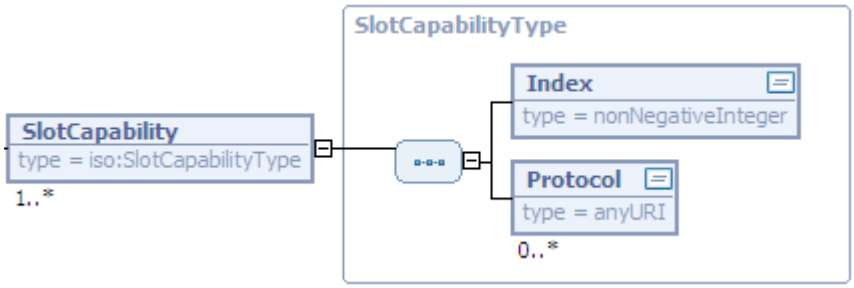
Return of the `GetIFDCapabilities` function.

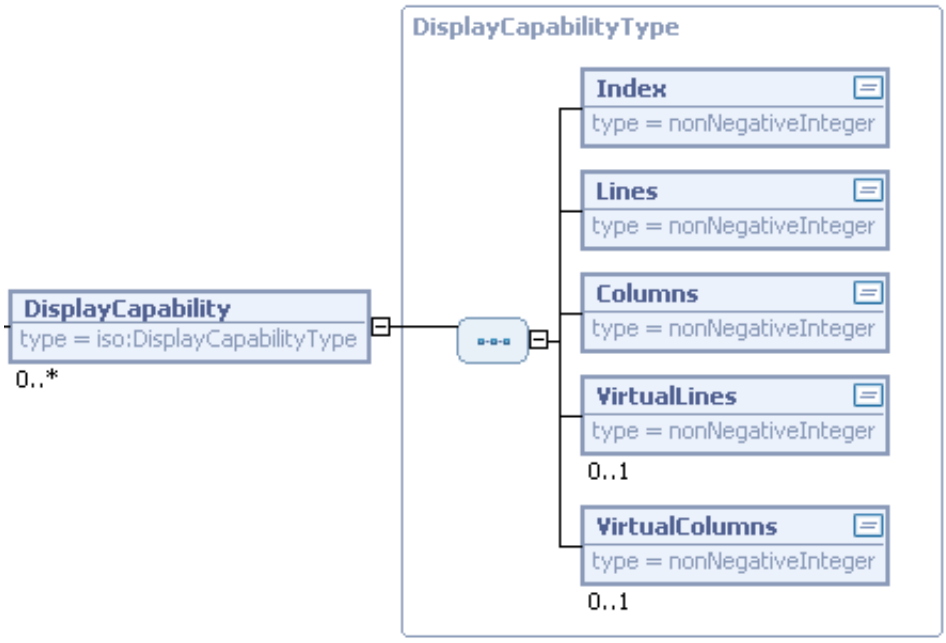
Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`IFDCapabilities`	Contains information on the capabilities of the terminal (see below for details).



The `IFDCapabilities` element contains information on the specified card terminal.

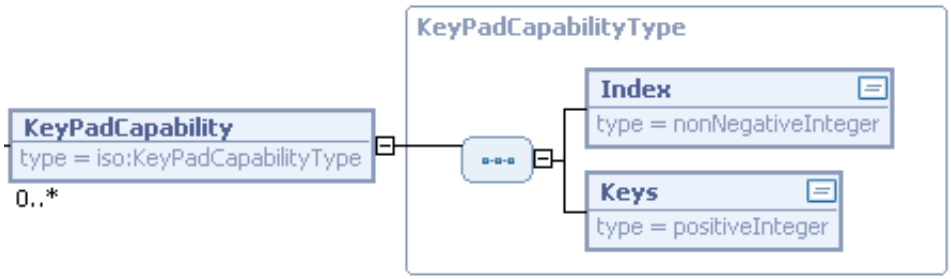
Name	Description
`SlotCapability`	This element is of the `SlotCapabilityType` and is provided for each slot of the card terminal containing information on the slot. See below for details.
`DisplayCapability`	An entry of the `DisplayCapabilityType` describing the display capabilities of the terminal is provided for each display on the IFD.

	`KeyPadCapability`	Such an entry with the capabilities of the keypad exists for each card terminal keypad. The entry is of the `KeyPadCapabilityType`.
	`BioSensorCapability`	Such an entry of the `BioSensorCapabilityType` with the capabilities of the sensor exists for each biometric sensor on the card terminal.
	`OpticalSignalUnit`	Contains information on whether the card terminal has an optical signal unit (e.g. LED).
	`AcousticSignalUnit`	Contains information on whether the card terminal has a unit for acoustic signals (e.g. beeping).
	 `SlotCapability` is part of `IFDCapabilities`.	
	Name	**Description**
	`Index`	Specifies the index of the slot in the range of 0 to the number of slots minus 1.
	`Protocol`	MAY be present multiple times and indicate the supported transport protocols (see `Interface`-element in [TR-03112-4]) or the supported DID-protocols (see [TR-03112-7]).Support of the IFD for PACE according to [TR-03119]/[PC/SC], Part 10 AMD1 is indicated by the URI `urn:oid:0.4.0.127.0.7.2.2.4:xx`, where `xx` is the decimal representation of the capabilites bitmap returned by `GetReader-PACECapabilites` as defined in [PC/SC], Part 10 AMD1.



The `DisplayCapabilityType` is part of `IFDCapabilities`.

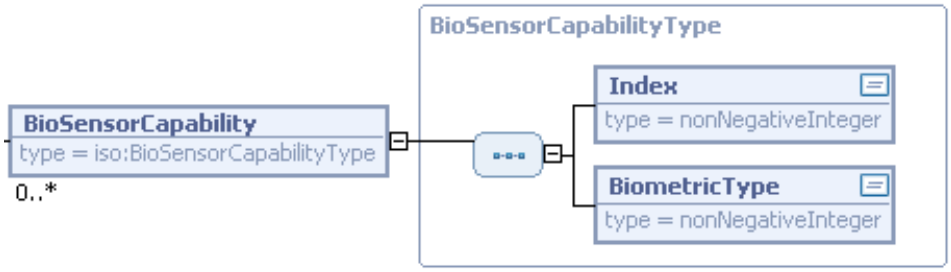
Name	Description
`Index`	The index of the display in the range of 0 to the number of displays minus 1.
`Lines`	Contains the number of visible lines of the display.
`Columns`	Specifies the number of visible columns of the display.
`VirtualLines`	If applicable, this optional parameter contains the number of lines which are supported by the display including scrolling.
`VirtualColumns`	If applicable, this optional parameter specifies how many columns of the display are supported with panning.



The `KeyPadCapabilityType` is part of `IFDCapabilities`.

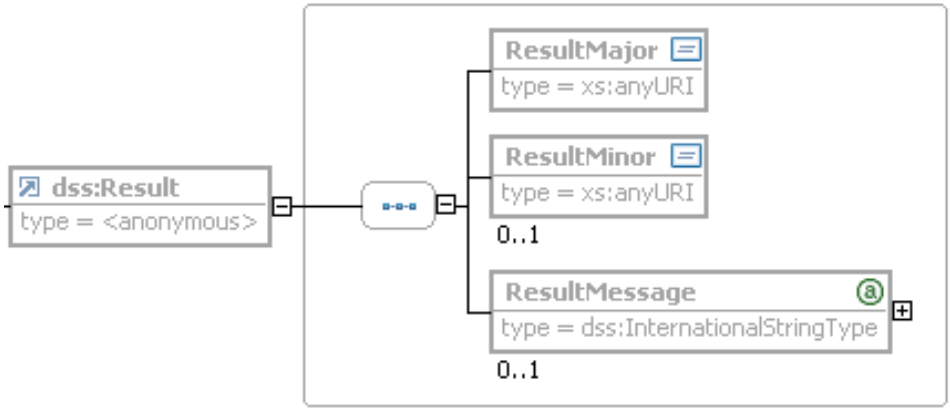
Name	Description
`Index`	Specifies the keypad index in the range of 0 to the number of keypads minus 1.

	`Keys`	Contains the number of keys on the keypad.



The `BioSensorCapabilityType` is part of `IFDCapabilities`.

Name	Description
`Index`	Specifies the biometric sensor index in the range of 0 to the number of biometric sensors minus 1.
`BiometricType`	Describes the type of the biometric characteristic in accordance with Section 7.8 of [ISO19784-1].



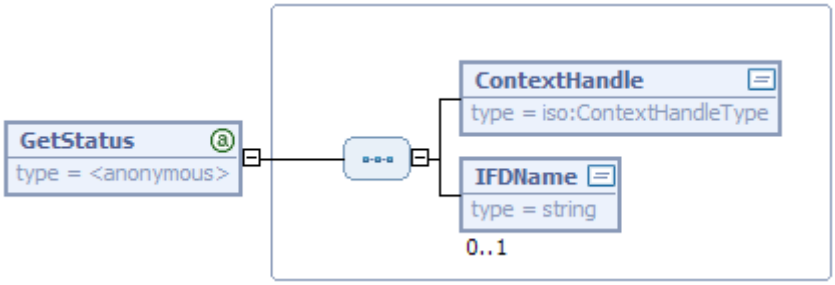
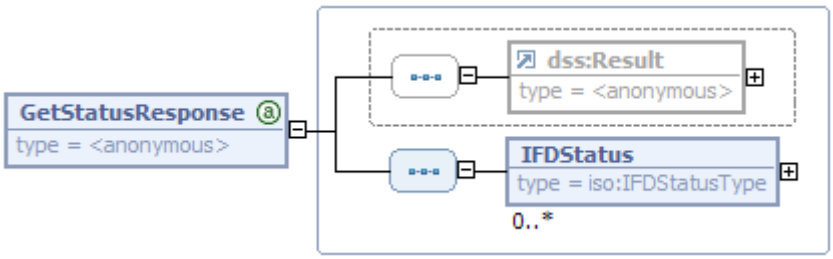
Status information and errors in `GetIFDCapabilities`

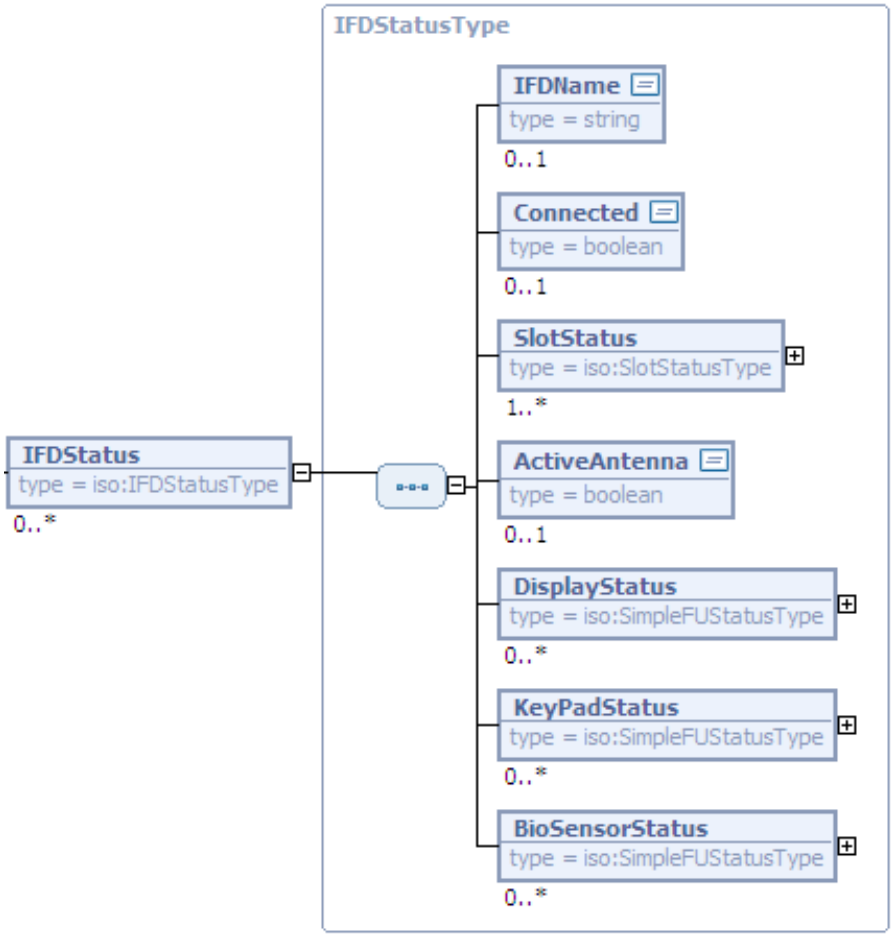
Name	Error codes
`ResultMajor`	• /resultmajor#ok • /resultmajor#error
`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/dp#invalidChannelHandle • /resultminor/ifdl/terminal#unknownIFD • /resultminor/al/common#unknownError
`ResultMessage`	MAY contain more detailed information on the error which occurred if required.

Preconditions	A context was established with `EstablishContext`.
Postconditions	The status of the card terminals remain unchanged.
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `GetReaderCapabilities` function in the PC/SC resource

	manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).

3.1.5 GetStatus

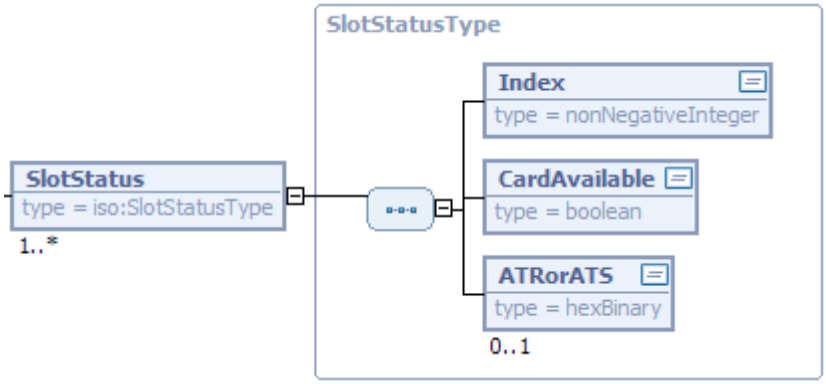
Name	`GetStatus`
Description	Determines the current status of one specific or all connected terminals.
Invocation parameters	 Invocation of the `GetStatus` function.
	Name / **Description**
	`ContextHandle` — Handle with which the session with the Terminal-Layer is addressed.
	`IFDName` — MAY contain the unique name of the card terminal. If this element is missing, the status of all connected terminals is returned.
Return	 Return of the `GetStatus` function.
	Name / **Description**
	`dss:Result` — Contains the status information and the errors of an executed action. This element is described in more detail below.
	`IFDStatus` — Contains information on the status of the respective terminal.



The `IFDStatus` element is part of `GetStatusResponse` and contains status information on the respective terminal.

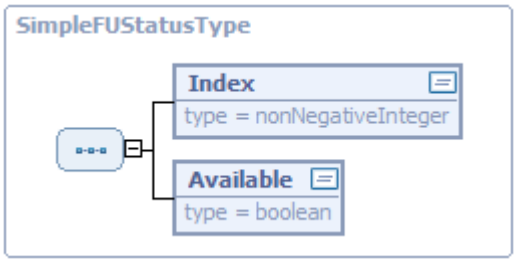
Name	Description
`IFDName`	MAY contain the unique name of the card terminal. This element MUST be present, if more than one terminal is present.
`Connected`	Contains information on whether a connection is currently established to the terminal. If the terminal is directly connected to the host (e.g. via RS232, USB, etc.), the parameter MAY be omitted.
`SlotStatus`	Contains status information for a specific slot for smart cards with contacts. The structure of `SlotStatusType` is defined below.
`ActiveAntenna`	Contains information on whether an existing radio antenna is activated. If no radio antenna is available, this element is omitted.
`DisplayStatus`	Contains status information on the available displays. See below for details. If no display is available, this element is omitted.

`KeyPadStatus`	Contains information on whether the existing keypads are currently available for an invoker. See below for details. If no keypad is available, this element is omitted.
`BioSensorStatus`	Contains information on the availability of the biometric sensors. See below for details. If no biometric sensors are available, this element is omitted.



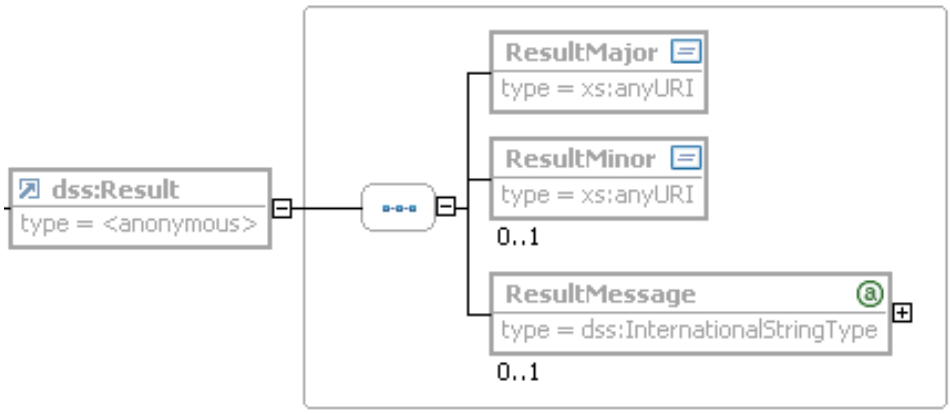
The `SlotStatus` element is of the `SlotStatusType` and is part of `IFDStatus`.

Name	Description
`Index`	Contains the index of the slot under consideration. The slots contain indexes in the range of 0 to the number of slots minus 1.
`CardAvailable`	Contains information on whether a card is currently captured by this slot.
`ATRorATS`	If a card is captured, this element contains the ATR or ATS of the card if available. If no card is captured, this element is omitted.



The `DisplayStatus` element, the `KeyPadStatus` element and the `BioSensorStatus` element are part of `IFDStatus` and of the `SimpleFUStatusType`.

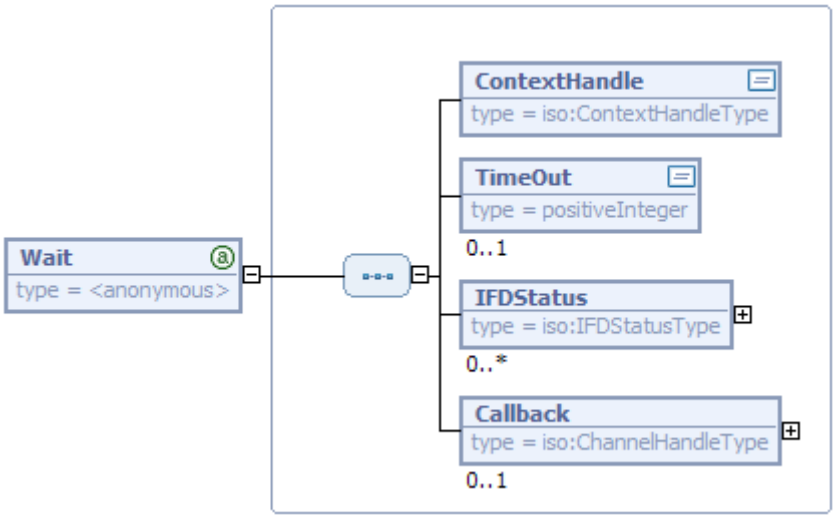
Name	Description
`Index`	Contains the index of the functional unit in the range of 0 to the number of functional units minus 1.

	`Available`	Specifies whether the functional unit is currently available for the invoker or if it is already being used.
	 Status information and errors in `GetStatusResponse`	
	Name	**Error codes**
	`ResultMajor`	• /resultmajor#ok • /resultmajor#error
	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/dp#invalidChannelHandle • /resultminor/ifdl/terminal#unknownIFD • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	A context was established with `EstablishContext`.	
Postconditions	The status of the card terminals remains unchanged.	
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `Status` function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).	

3.1.6 Wait

Name	**`Wait`**
Description	The `Wait` function informs the invoking layer about events on specific card terminals. Information on which events have occurred can be returned by the return of the `Wait` function or – if a corresponding callback address was transmitted – by the `SignalEvent` function.

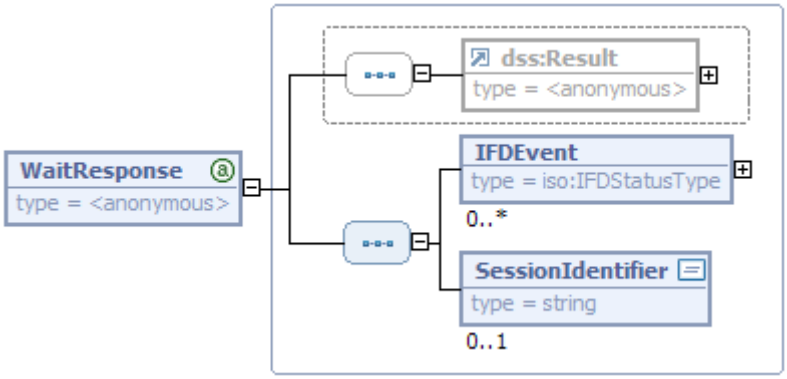
Invocation parameters



Invocation of the `Wait` function.

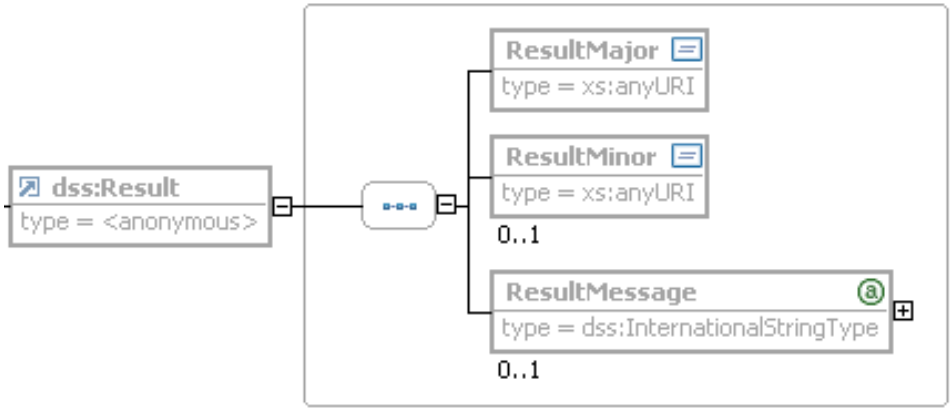
Name	Description
`ContextHandle`	Handle with which the session with the Terminal-Layer is addressed.
`TimeOut`	Optional parameter which contains the time until timeout in milliseconds. If the parameter is missing, waiting continues for an infinite period (until invocation of the `Cancel` or `ReleaseContext` function or termination of the complete framework with `TerminateFramework` (also refer to [TR-03112-3])).
`IFDStatus`	Such an entry of type `IFDStatusType` is provided for each card terminal requiring monitoring with the currently assumed status information for this terminal. Information on the `IFDStatusType` is given on page 18.
`Callback`	MAY specify a callback address and other corresponding parameters (also refer to `ChannelHandle` in [TR-03112-4]) to which a `SignalEvent` invocation specified in Annex A is sent when a card terminal event occurs. If this element is provided, the function immediately returns with `WaitResponse`. Otherwise return is delayed until a corresponding event or timeout occurs.

Return



Return of the `Wait` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`IFDEvent`	`IFDEvent` can occur several times and contains information on an event which occurred on a card terminal. Similar to the input parameter `IFDStatus`, this parameter is of the `IFDStatusType`, and contains the status information changed by the event.
`SessionIdentifier`	Is available if a `Callback` address was provided when the `Wait` function was invoked and specifies an identifier unique in the Terminal-Layer with which waiting for card terminal events can be terminated with the `Cancel` function.

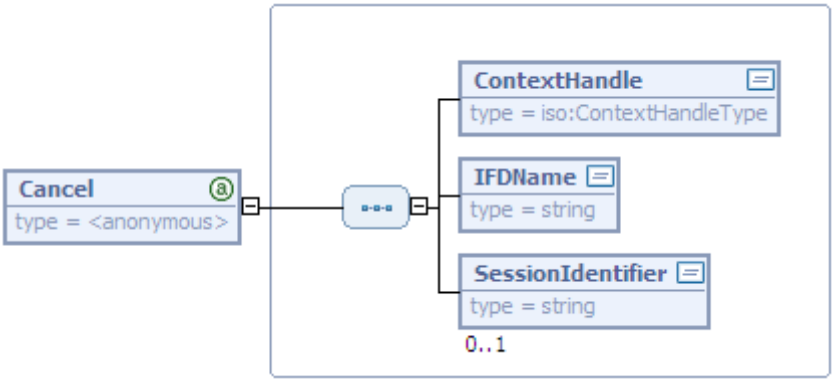


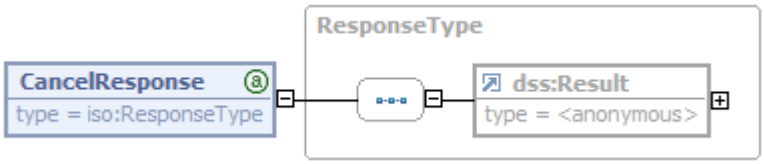
Status information and errors in `WaitResponse`

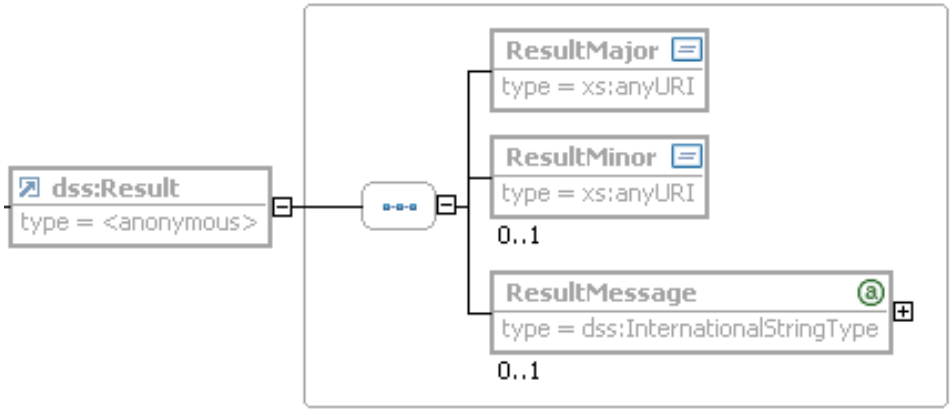
Name	Error codes
`ResultMajor`	• /resultmajor#ok • /resultmajor#error

	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/dp#invalidChannelHandle • /resultminor/ifdl/terminal#unknownIFD • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	A context was established with `EstablishContext`.	
Postconditions		
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `GetStatusChange` function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.4.2).	

3.1.7 Cancel

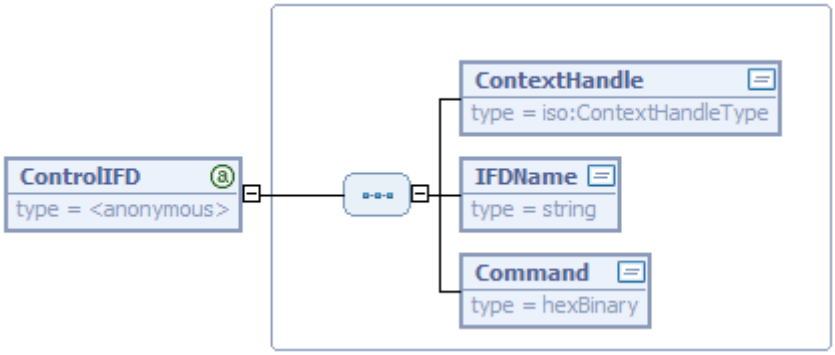
Name	**`Cancel`**	
Description	Terminates waiting for card terminal events started with Wait, or attempts to cancel processing of the last command sent to a terminal.	
Invocation parameters	 Invocation of the `Cancel` function.	
	Name	**Description**
	`ContextHandle`	Handle with which the session with the Terminal-Layer is addressed.
	`IFDName`	Unique name of the card terminal at which a command currently being executed should be terminated.
	`SessionIdentifier`	Specifies which waiting process should be terminated with `Wait`.

<table>
<tr><td>Return</td><td>


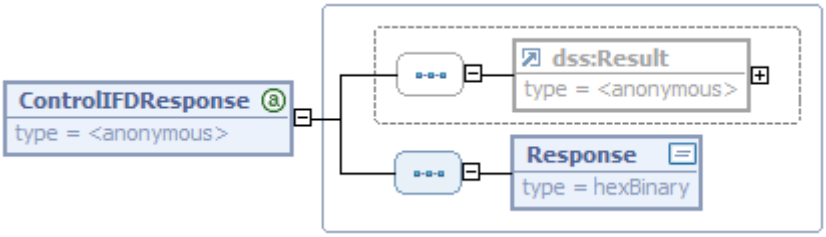
Return of the <code>Cancel</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>dss:Result</code></td><td>Contains the status information and the errors of an executed action. This element is described in more detail below.</td></tr>
</table>


Status information and errors in <code>Cancel</code>
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error
• /resultmajor#warning</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/ifdl/IO#cancelNotPossible
• /resultminor/dp#invalidChannelHandle
• /resultminor/ifdl/terminal#unknownIFD
• /resultminor/al/common#unknownError</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Preconditions</td><td>A context was established with <code>EstablishContext</code>.</td></tr>
<tr><td>Postconditions</td><td>Depending on the command and time of the request, the command is either terminated or fully executed.</td></tr>
<tr><td>Notes</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the <code>Cancel</code> function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.4.2).</td></tr>
</table>

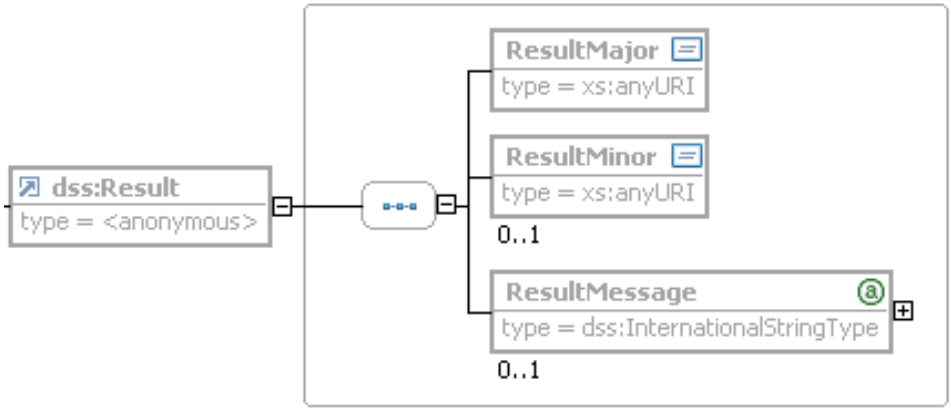
3.1.8 ControlIFD

Name	`ControlIFD`
Description	Sends any arbitrary command directly to the card terminal and returns the corresponding result.
Invocation parameters	 Invocation of the `ControlIFD` function.

Name	Description
`ContextHandle`	Handle with which the session with the Terminal-Layer is addressed.
`IFDName`	Unique name of the card terminal to which a command is to be sent.
`Command`	Command protocol data unit for the card terminal.

Return	 Return of the `ControlIFD` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`Response`	Response protocol data unit which was returned from the card terminal.

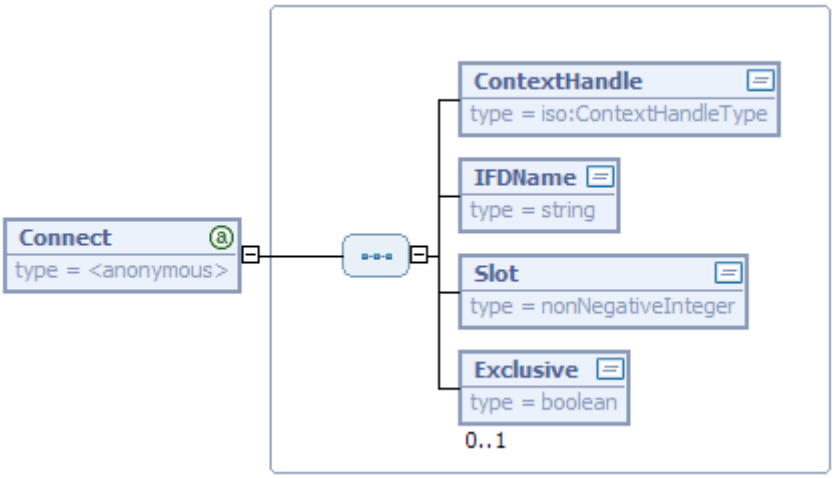
<table>
<tr><td></td><td>


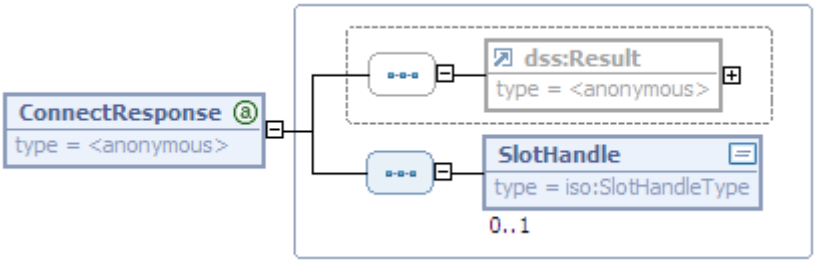
Status information and errors in <code>ControlIFD</code>
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/dp#invalidChannelHandle
• /resultminor/ifdl/terminal#unknownIFD
• /resultminor/al/common#unknownError</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Preconditions</td><td>A context was established with <code>EstablishContext</code>.</td></tr>
<tr><td>Postconditions</td><td>The command was executed by the card terminal.</td></tr>
<tr><td>Notes</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function using FEATURE_MCT_READERDIRECT (also refer to [PC/SC], Part 10, Section 2.14).</td></tr>
</table>

3.2 Card functions

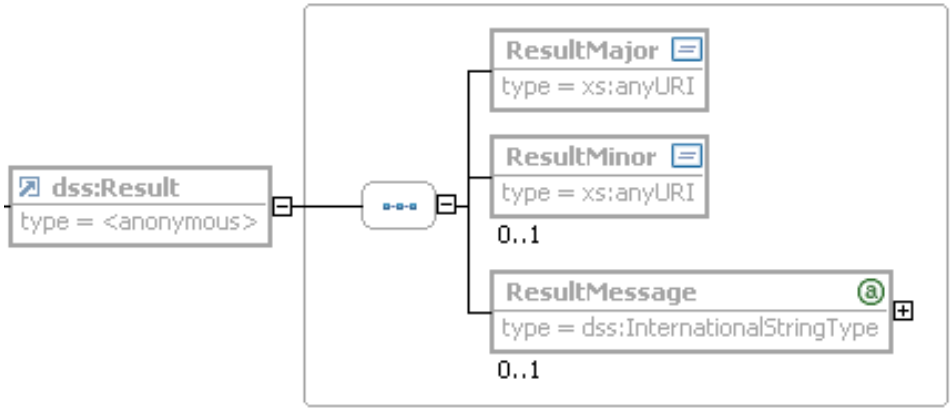
3.2.1 Connect

Name	**`Connect`**
Description	Establishes a connection to an eCard in a specific card terminal slot and returns a corresponding `SlotHandle` to the calling layer.

<table>
<tr><td>Invocation parameters</td><td>


Invocation of the <code>Connect</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>ContextHandle</code></td><td>Handle with which the session with the Terminal-Layer is addressed.</td></tr>
<tr><td><code>IFDName</code></td><td>Unique name of the card terminal.</td></tr>
<tr><td><code>Slot</code></td><td>Addresses the slot (ICC slot or PCD) of an IFD by which a card is detected and with which a connection should be established. Assignment of the slot indexes is explained in the definition of the <code>SlotStatusType</code> (also refer to page 19).</td></tr>
<tr><td><code>Exclusive</code></td><td>Is <code>TRUE</code> if an exclusive connection should be established to the eCard.</td></tr>
</table>
</td></tr>
<tr><td>Return</td><td>


Return of the <code>Connect</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>dss:Result</code></td><td>Contains the status information and the errors of an executed action. This element is described in more detail below.</td></tr>
<tr><td><code>SlotHandle</code></td><td>If successful, a <code>SlotHandle</code> is returned with which the connection established with <code>Connect</code> to the eCard is addressed in future.</td></tr>
</table>
</td></tr>
</table>

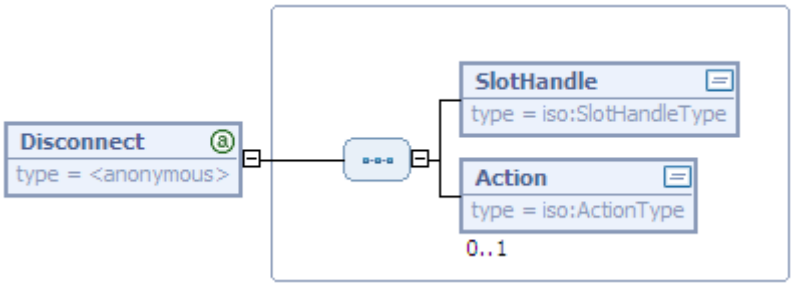
<table>
<tr><td></td><td>


Status information and errors in <code>Connect</code>
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/ifdl/common#invalidContextHandle
• /resultminor/ifdl/terminal#unknownIFD
• /resultminor/ifdl/terminal#unknownSlot
• /resultminor/ifdl/terminal#IFDSharingViolation
• /resultminor/ifdl/terminal#noCard
• /resultminor/al/common#unknownError</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Preconditions</td><td>A context was established with <code>EstablishContext</code>.</td></tr>
<tr><td>Postconditions</td><td>A logical connection to the card was established. APDUs can then be sent to the card via the returned <code>SlotHandle</code> with the <code>Transmit</code> function. As a result, the path to a card application can also be determined with <code>CardApplicationPath</code> and established by means of <code>CardApplicationConnect</code> (also refer to [TR-03112-4]).</td></tr>
<tr><td>Notes</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the <code>Connect</code> function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).</td></tr>
</table>

3.2.2 Disconnect

Name	**`Disconnect`**
Description	The `Disconnect` function terminates the connection to a card and MAY execute an additional action (e.g. ejection of the card), if the corresponding mechanical functionality is supported by the terminal.

Invocation parameters



Invocation of the `Disconnect` function.

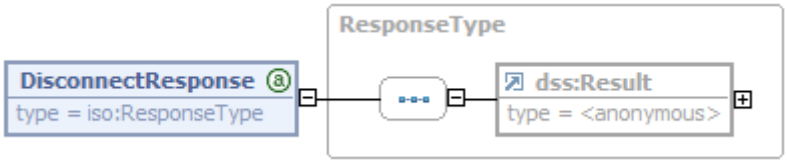
Name	Description
`SlotHandle`	With the `SlotHandle` the connection to the eCard established with `Connect` is addressed.
`Action`	Optional parameter which MAY specify an action which is to be performed additionally. The `ActionType` is defined as follows:

```
<simpleType name="ActionType">
        <restriction base="string">
                <enumeration value="Reset" />
                <enumeration value="Unpower" />
                <enumeration value="Eject" />
                <enumeration value="Confiscate" />
        </restriction>
</simpleType>
```

The values have the following meaning:

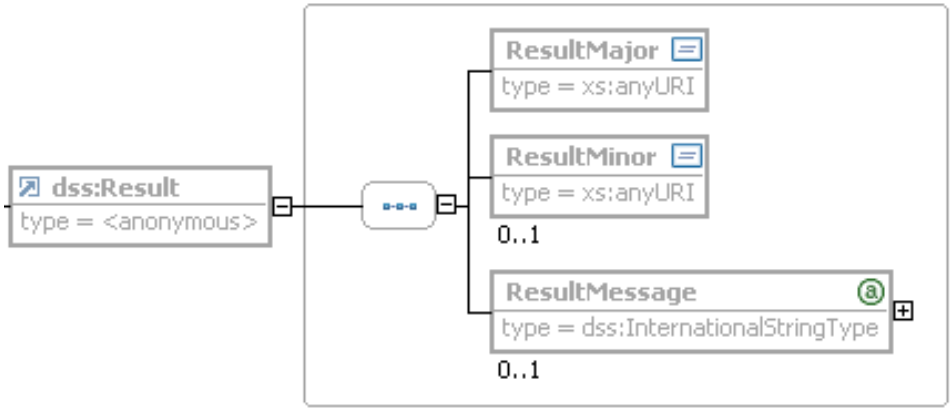
Value	Meaning
`Reset`	Reset of the eCard.
`Unpower`	Interrupts the power supply of the card.
`Eject`	Ejection of the eCard from the card terminal if the mechanical functionality is available.
`Confiscate`	Confiscation of the eCard if the corresponding functionality is available.

Return



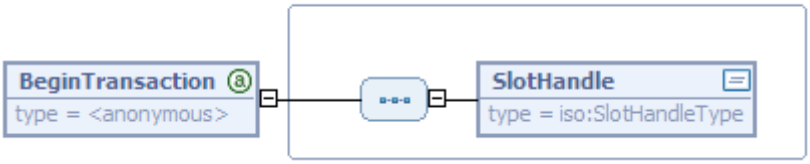
Return of the `Disconnect` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.

<table>
<tr><td></td><td>


Status information and errors in <code>Disconnect</code>
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/ifdl/common#invalidSlotHandle
• /resultminor/ifdl/terminal#unknownAction
• /resultminor/al/common#unknownError</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Preconditions</td><td>A connection to an eCard was established with <code>Connect</code>.</td></tr>
<tr><td>Postconditions</td><td>The <code>SlotHandle</code> loses its validity.</td></tr>
<tr><td>Notes</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the <code>Disconnect</code> function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).</td></tr>
</table>

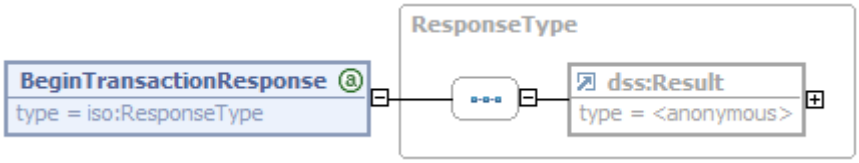
3.2.3 BeginTransaction

<table>
<tr><td>Name</td><td><code>BeginTransaction</code></td></tr>
<tr><td>Description</td><td>The <code>BeginTransaction</code> function starts a transaction within which a series of commands can be sent to the eCard without permitting access of another process to the eCard. If a command is not successful in the transaction, the complete transaction is reset.</td></tr>
<tr><td>Invocation parameters</td><td>


Invocation of the <code>BeginTransaction</code> function.
</td></tr>
</table>

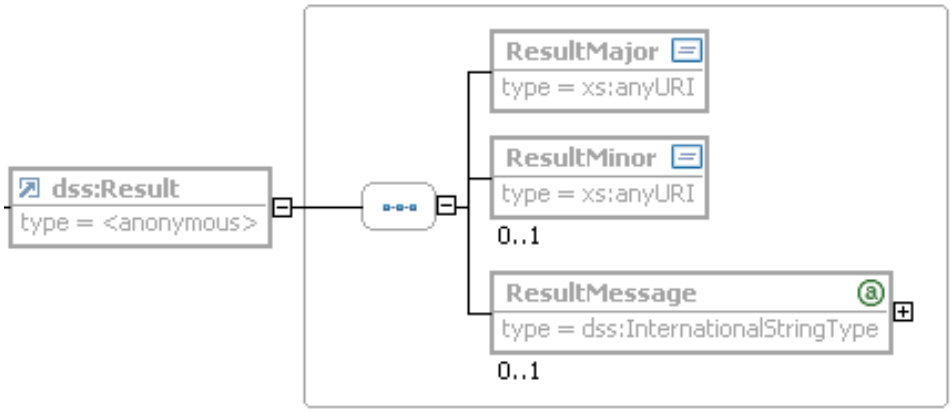
	Name	Description
	`SlotHandle`	With the `SlotHandle` the connection established with `Connect` to the eCard is addressed.

Return



Return of the `BeginTransaction` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.

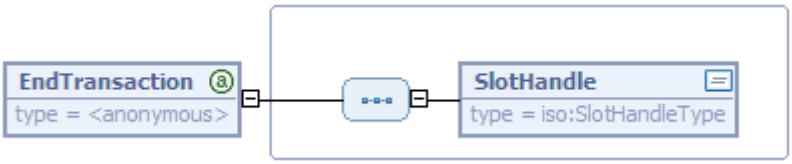


Status information and errors in `BeginTransaction`

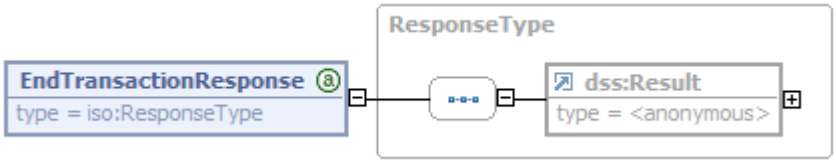
Name	Error codes
`ResultMajor`	• /resultmajor#ok • /resultmajor#error
`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/ifdl/common#invalidSlotHandle • /resultminor/al/common#unknownError
`ResultMessage`	MAY contain more detailed information on the error which occurred if required.

Preconditions	A connection to an eCard was established with `Connect`.
Postconditions	A transaction is started.
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `BeginTransaction` function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).

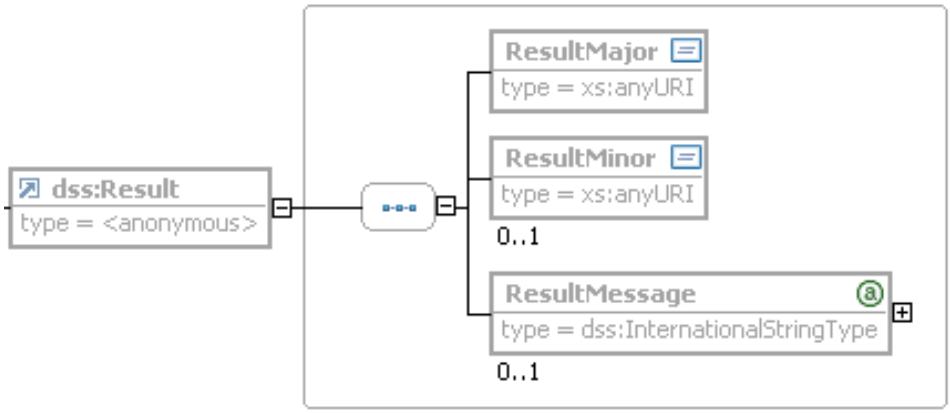
3.2.4 EndTransaction

Name	`EndTransaction`
Description	The `EndTransaction` function terminates an existing transaction with a selected eCard.
Invocation parameters	 Invocation of the `EndTransaction` function.

Name	Description
`SlotHandle`	The `SlotHandle` addresses the connection to the eCard.

Return	 Return of the `EndTransaction` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.

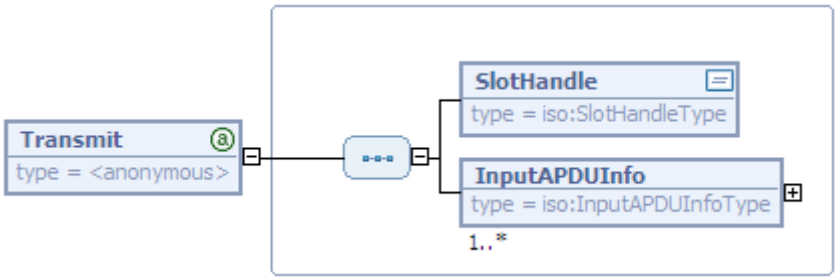


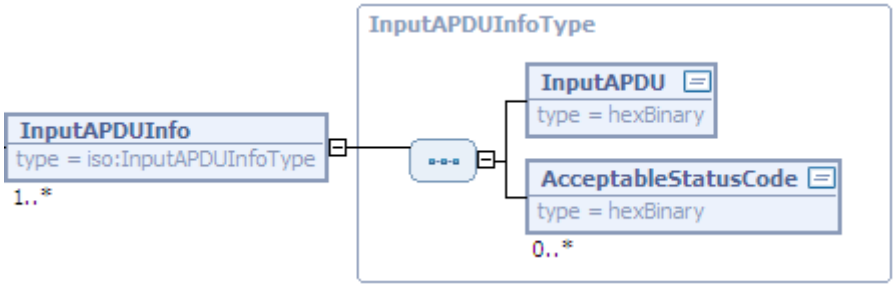
Status information and errors in `EndTransaction`

Name	Error codes
`ResultMajor`	• /resultmajor#ok • /resultmajor#error • /resultmajor#warning

	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/ifdl/common#invalidSlotHandle • /resultminor/ifdl/IO#noTransactionStarted • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	A connection to an eCard was established with `Connect` and a transaction started for the thus-connected card with `BeginTransaction`.	
Postconditions	The transaction is terminated.	
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the `EndTransaction` function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).	

3.2.5 Transmit

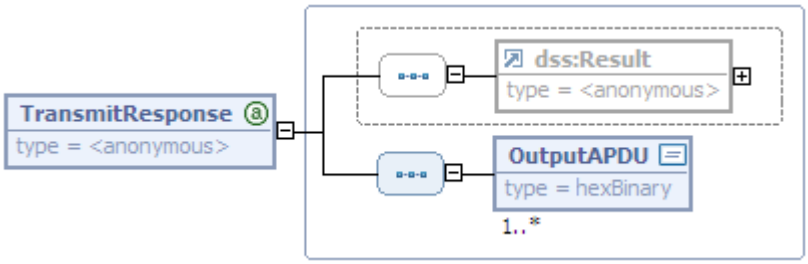
Name	**`Transmit`**	
Description	The `Transmit` function sends one or more APDU(s) to a connected eCard. In order to support the batch processing a set of `AcceptableStatusCode`-elements (9000 etc.) MAY be attached to each `InputAPDU`. If the eCard returns some not expected status code it is – even in case of secure messaging – clear that there is a serious error and it does not make sense to feed the remaining `InputAPDU`-elements in the batch to the eCard.	
Invocation parameters	 Invocation of the `Transmit` function.	
	Name	**Description**
	`SlotHandle`	With the `SlotHandle` the connection established with `Connect` to the eCard is addressed.
	`InputAPDUInfo`	MAY be present multiple times and contains the command APDU, which is sent to the eCard and optionally acceptable status codes. It is of type `InputAPDUInfoType`, which is explained below.



The `InputAPDUInfo` contains information about an APDU, which will be sent to the card.

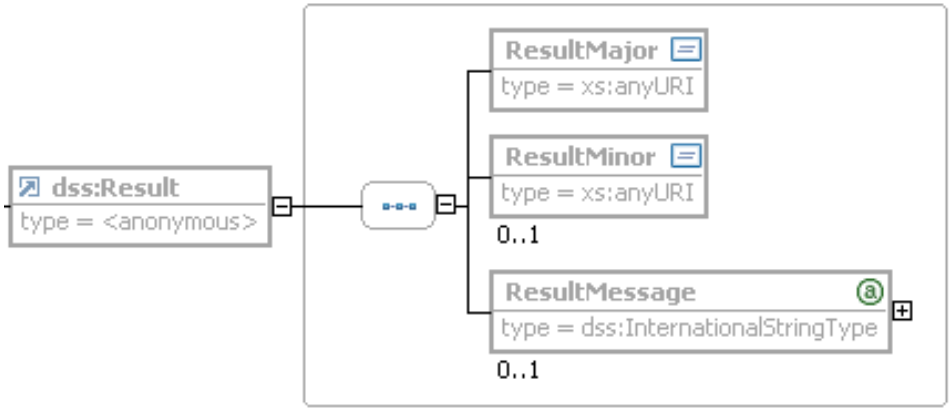
Name	Description
`InputAPDU`	Contains the APDU which is to be sent to the eCard.
`AcceptableStatusCode`	MAY be present multiple times per `InputAPDU`-element in order to specify the set of expected status codes. If the status code which is returned from the eCard is not among the expected values the batch processing SHALL be stopped and the result of the processing returned to the caller as this indicates that there is a serious error condition. If the `AcceptableStatusCode`-element is omitted, any returned status code is assumed to be acceptable. `AcceptableStatusCode`-elements containing only one byte match all status codes starting with this byte.

Return



Return of the `Transmit` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`OutputAPDU`	MAY be present multiple times and contains the APDU returned by the eCard. If the batch processing is stopped because a non-acceptable status word (see above) was returned, all response ADPUs including the one containing the non-acceptable status word SHALL be included. A successful call of the `Transmit` function MUST contain exactly as many `InputAPDU`- as `OutputAPDU`-elements.

<table>
<tr><td></td><td>


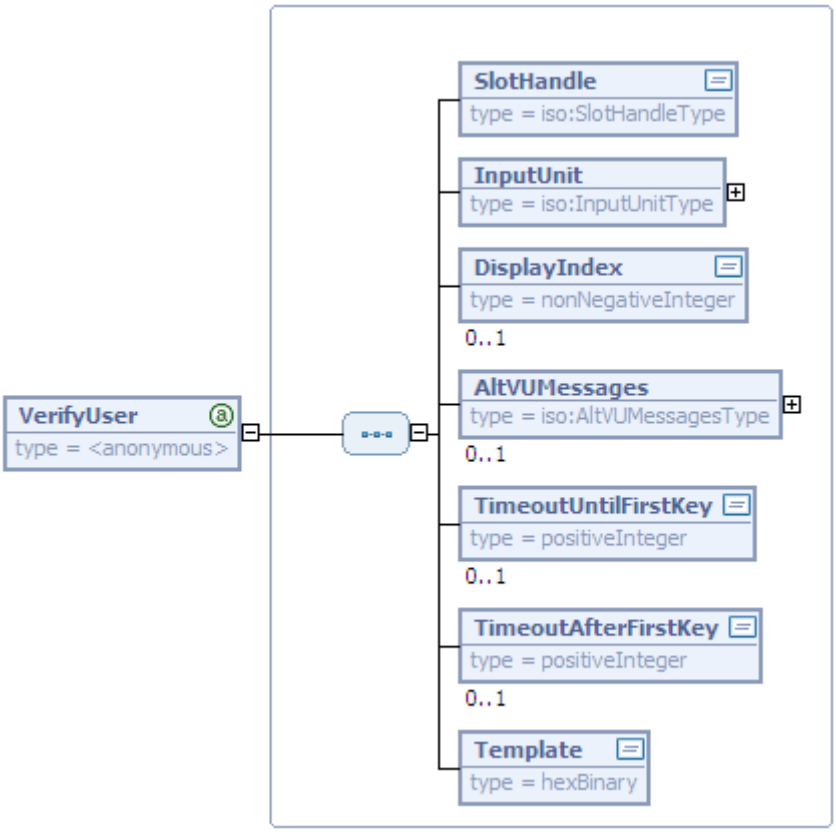
Status information and errors in <code>Transmit</code>
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/ifdl/common#invalidSlotHandle
• /resultminor/al/common#unknownError</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Preconditions</td><td>A connection to an eCard was established with <code>Connect</code>.</td></tr>
<tr><td>Postconditions</td><td>The commands specified by the batch of APDU are executed by the eCard.</td></tr>
<tr><td>Notes</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the <code>Transmit</code> function in the PC/SC resource manager (also refer to [PC/SC], Part 5, Section 3.2.5.2).</td></tr>
</table>

3.3 User interaction functions

3.3.1 VerifyUser

Name	**`VerifyUser`**
Description	The `VerifyUser` function initiates user verification with a PIN or a biometric characteristic.

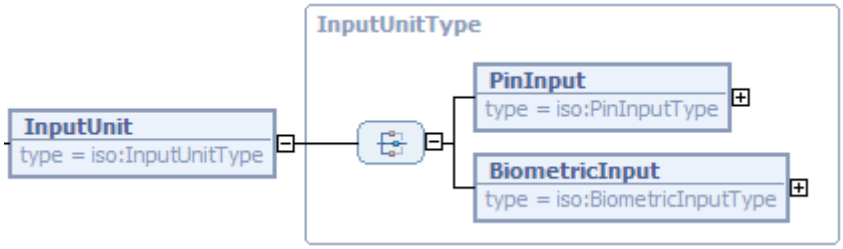
Invocation parameters



Invocation of the `VerifyUser` function.

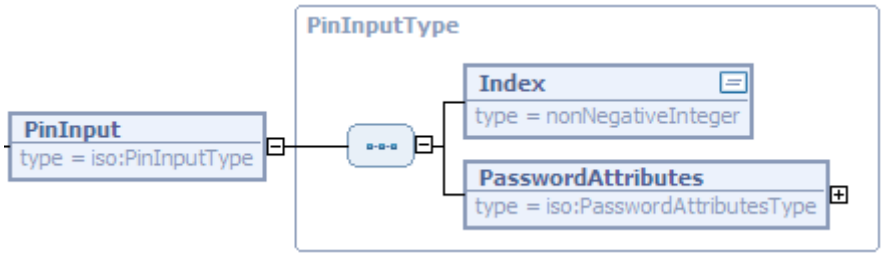
Name	Description
`SlotHandle`	With the `SlotHandle` the connection (established with `Connect`) to the eCard is addressed.
`InputUnit`	Specifies the input unit which is to be used for user verification and also contains additional information on possible processing of user verification data. The entry is of the `InputUnitType` described below.
`DisplayIndex`	Specifies the index of the display on which the messages should be shown for user guidance. If no information is to be displayed or if the card terminal does not have a display, this parameter is omitted.
`AltVUMessages`	Is an optional parameter which specifies the various messages which should be shown on the display during the verification process (see below for details). If this parameter is missing, standard texts are displayed.
`TimeoutUntilFirstKey`	Is an optional parameter which describes the timeout in milliseconds before the first key actuation.

	`TimeoutAfterFirstKey`	Is an optional parameter which describes the timeout in milliseconds after the first key actuation.
	`Template`	If applicable, the acquired verification data are entered into the template before the data are sent to the eCard. The structure of the template corresponds to the structure of an APDU for the VERIFY command in accordance with [ISO7816-4] (Section 7.5.6).



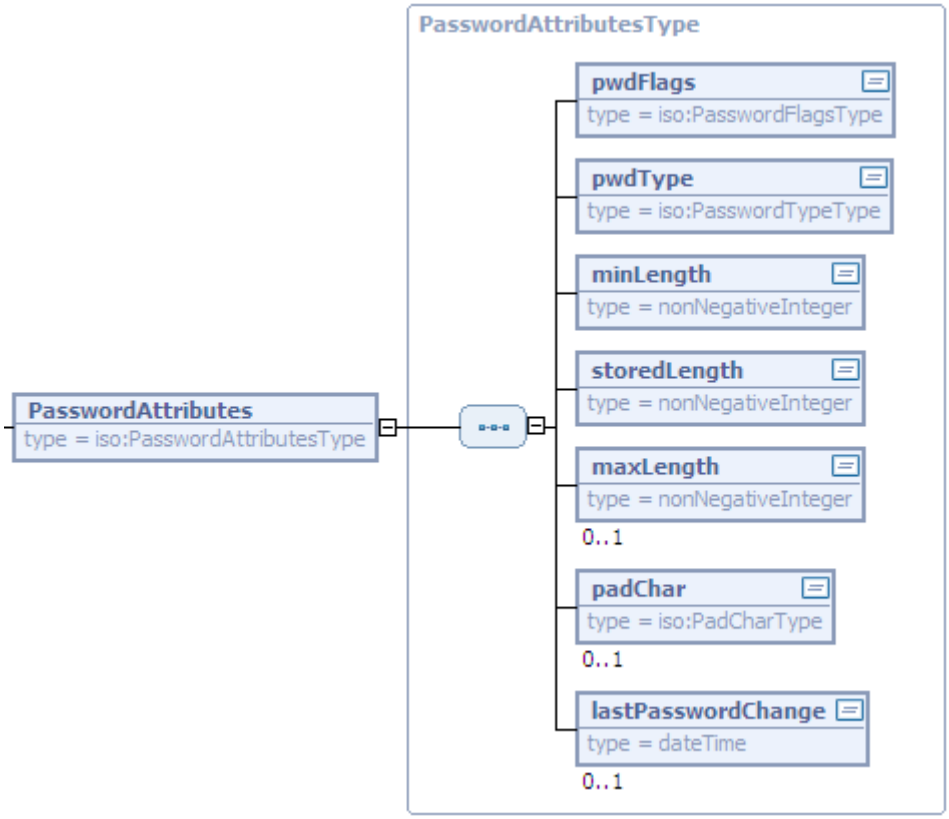
`InputUnit` is an invocation parameter of `VerifyUser`

Name	Description
`PinInput`	This parameter is used if the user is to be authenticated by means of a PIN. It is of the `PinInputType` described below.
`BiometricInput`	If the user is authenticated by a biometric characteristic, a parameter of the `BiometricInputType` (see below for details) must be specified.



The `PinInput` element is a possible child element of the `InputUnit` element.

Name	Description
`Index`	The index of the PIN pad to be used.
`PasswordAttributes`	Contains the password attributes as defined in [ISO7816-15]. Also refer to [TR-03112-4]. See below for details.



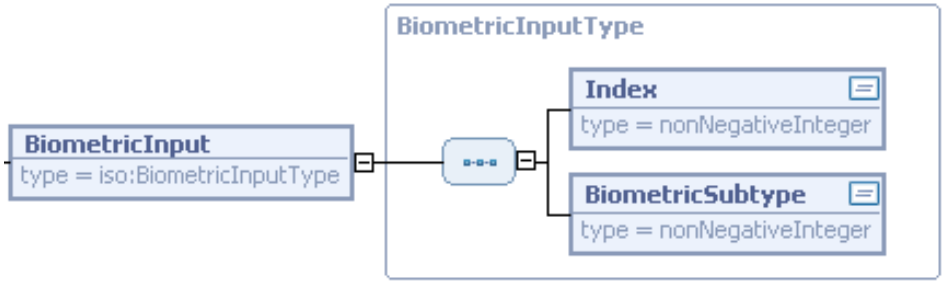
The `PasswordAttributes` element is part of the `PinInputType` (see above) and of the `PinCompareQualifierType` (refer to [TR-03112-4]).

Name	Description

	pwdFlags	Contains information on the character of the PIN (also refer to pwdFlags in accordance with [ISO7816-15]). The PasswordFlagsType is defined as follows:
		`<simpleType name="PasswordFlagsType">`
		`<union memberTypes="iso:BitString">`
		`<simpleType>`
		`<list>`
		`<simpleType>`
		`<restriction base="token">`
		`<enumeration value="case-sensitive" />`
		`<enumeration value="local" />`
		`<enumeration value="change-disabled" />`
		`<enumeration value="unblock-disabled" />`
		`<enumeration value="initialized" />`
		`<enumeration value="needs-padding" />`
		`<enumeration value="unblockingPassword" />`
		`<enumeration value="soPassword" />`
		`<enumeration value="disable-allowed" />`
		`<enumeration value="integrity-protected" />`
		`<enumeration value="confidentiality-protected"/>`
		`<enumeration value="exchangeRefData" />`
		`<enumeration value="resetRetryCounter1" />`
		`<enumeration value="resetRetryCounter2" />`
		`</restriction>`
		`</simpleType>`
		`</list>`
		`</simpleType>`
		`</union>`
		`</simpleType>`

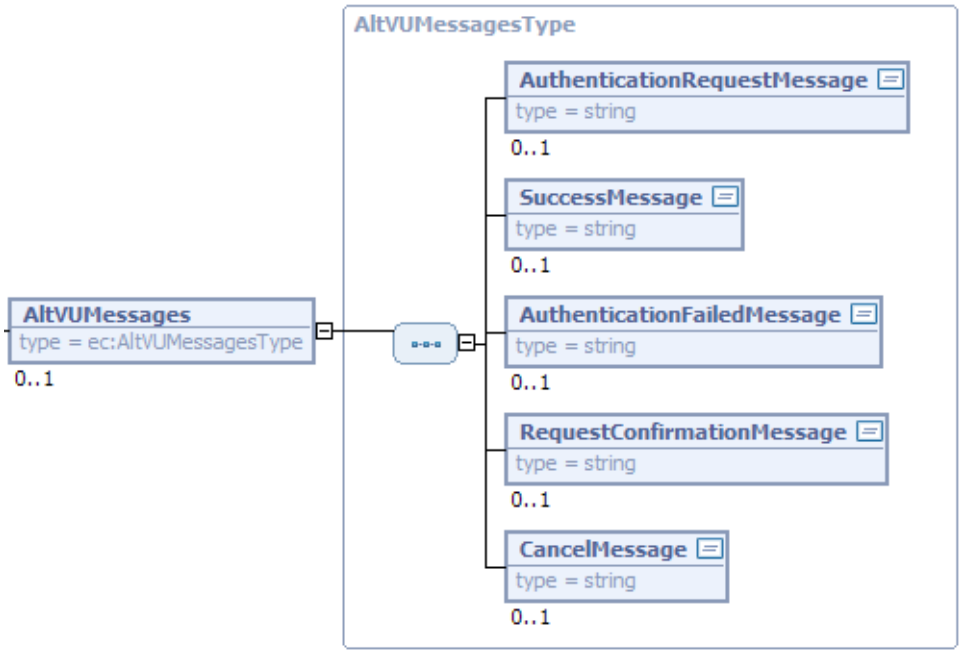
	pwdType	Contains information on the type of PIN (also refer to pwdType in accordance with [ISO7816-15]). The PasswordTypeType is defined as follows:
		`<simpleType name="PasswordTypeType">`
		`<restriction base="string">`
		`<enumeration value="bcd" />`
		`<enumeration value="ascii-numeric" />`
		`<enumeration value="utf8" />`
		`<enumeration value="half-nibble-bcd" />`
		`<enumeration value="iso9564-1" />`
		`</restriction>`
		`</simpleType>`

	minLength	Contains the minimum length of the PIN.
	storedLength	Contains the length of the PIN as stored on the card.
	maxLength	MAY contain the maximum length.
	padChar	MAY contain the padding character which is to be used for padding.
	lastPassword Change	MAY contain the time of the last PIN modification.



`BiometricInput` is a parameter of the `InputUnitType`

Name	Description
`Index`	The index of the biometric sensor which is to be used.
`BiometricSubtype`	Specifies the subtype of the biometric characteristic in accordance with Section 7.14 of [ISO19784-1].

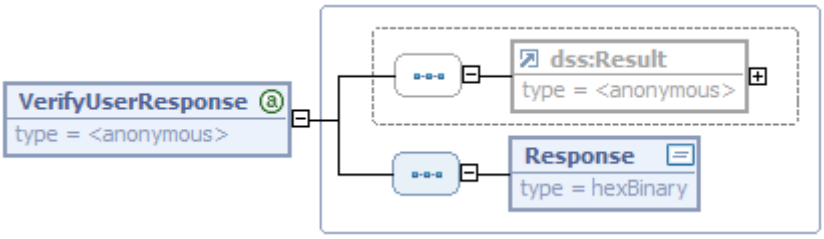


`AltVUMessages` is part of `VerifyUser`

Name	Description
`Authentication RequestMessage`	Prompts the user to perform user verification (e.g. by entering a PIN). If this element is missing, the configured default message is used.
`SuccessMessage`	Informs the user that user verification was successful. If this element is missing, the configured default message is used.
`AuthenticationFa iledMessage`	Shows the user that the user verification was not successful and that the eCard MAY therefore be blocked. If this element is missing, the configured default message is used.

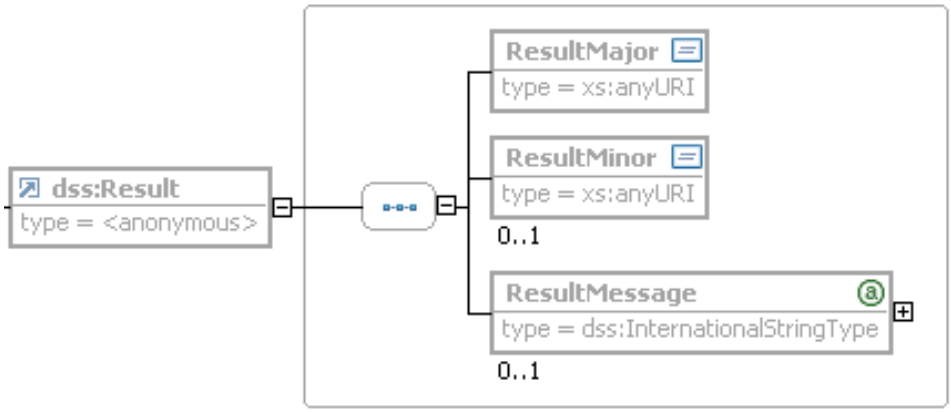
	`RequestConfirmationMessage`	Prompts the user to repeat the entry. If this element is missing, the configured default message is used.
	`CancelMessage`	Shows the user that the entry was cancelled. If this element is missing, the configured default message is used.

Return



Return of the `VerifyUser` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`Response`	The response of the eCard (e.g. 9000 in the event of successful user verification).



Status information and errors in `VerifyUser`

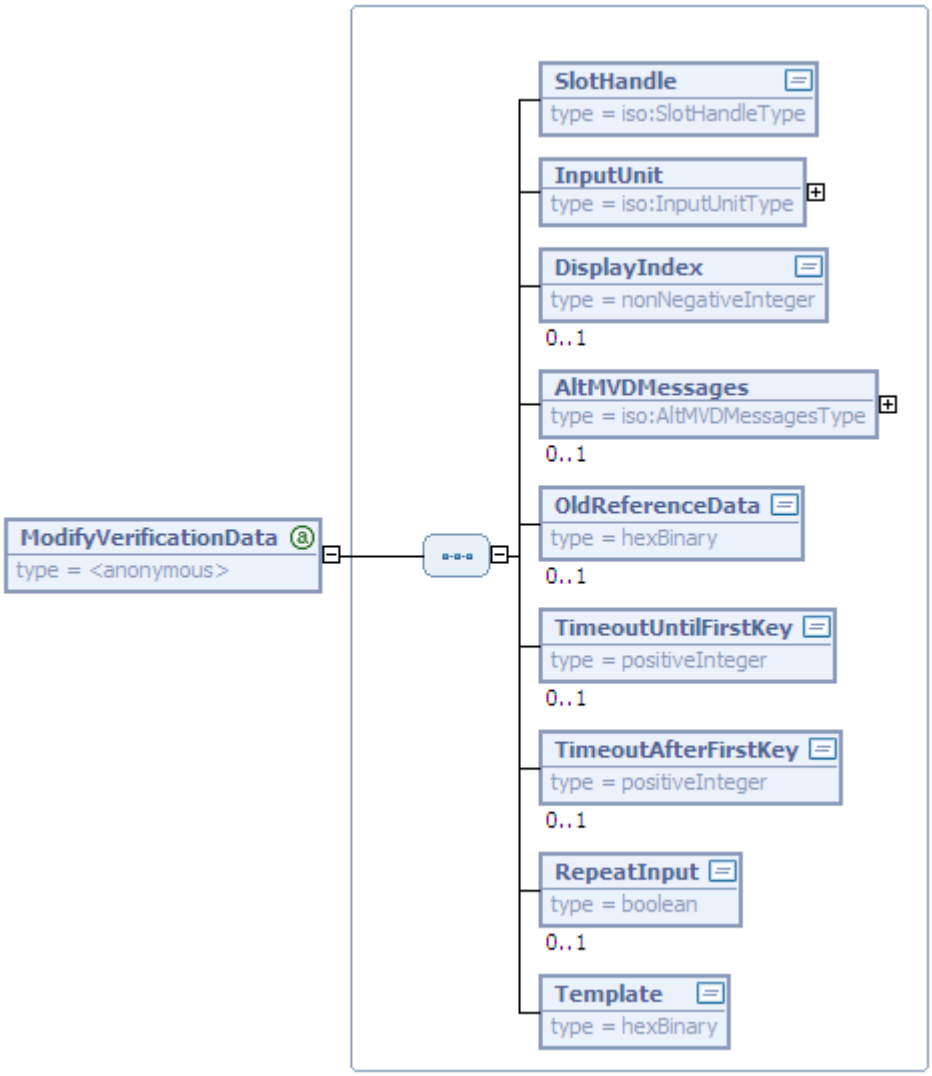
Name	Error codes
`ResultMajor`	• /resultmajor#ok • /resultmajor#error
`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/ifdl/common#invalidSlotHandle • /resultminor/ifdl/IO#unknownInputUnit • /resultminor/ifdl#cancellationByUser • /resultminor/al/common#unknownError • /resultminor/ifdl/IO#unknownPINFormat • /resultminor/ifdl/IO#unknownBiometricSubtype

	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	A connection to an eCard was established with `Connect`.	
Postconditions	The corresponding authentication status on the card was established by invocation of VERIFY in accordance with [ISO7816-4].	
Notes	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with FEATURE_VERIFY_PIN_START specified in [PC/SC], Part 10 (Section 2.9) using the PIN_VERIFY_STRUCTURE defined in [PC/SC], Part 10 (Section 2.5).	

3.3.2 ModifyVerificationData

Name	**`ModifyVerificationData`**
Description	With this function the data for user authentication (PIN or biometric reference data) on an eCard are changed.

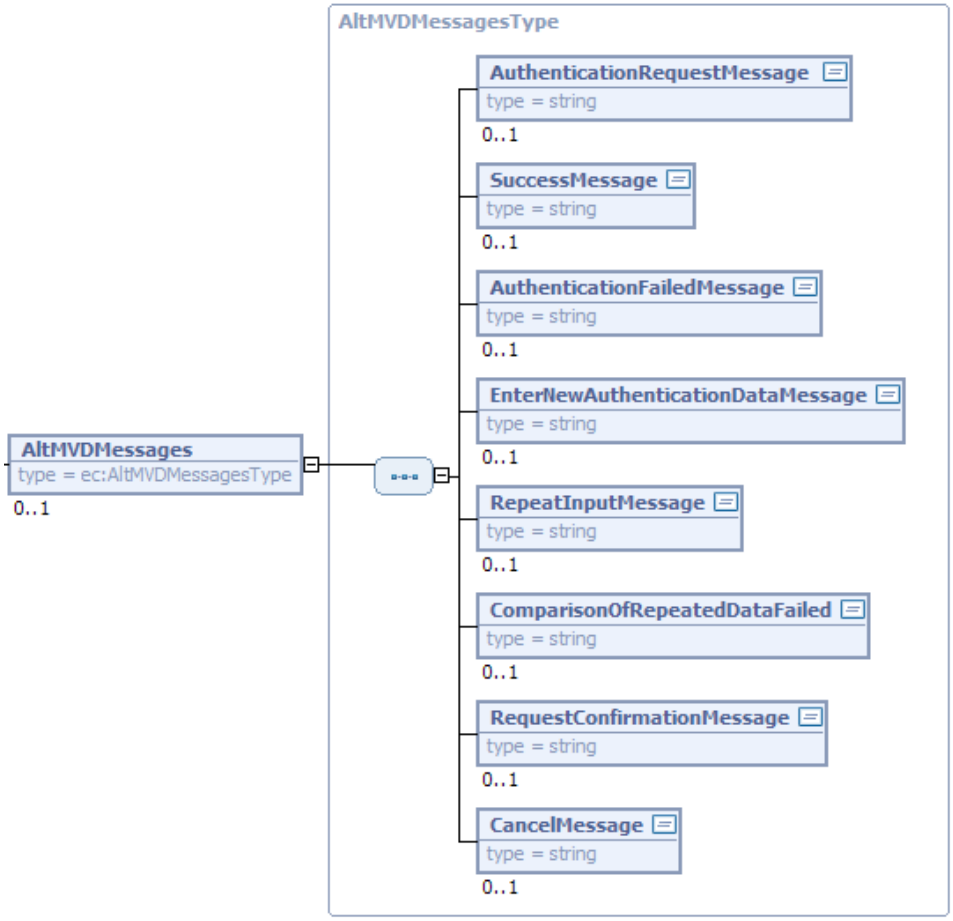
Invocation parameters



Invocation of the `ModifyVerificationData` function.

Name	Description
`SlotHandle`	With the `SlotHandle` the connection to the eCard (established with `Connect`) is addressed.
`InputUnit`	Specifies the input unit (for details refer to page 37).
`DisplayIndex`	Specifies the index of the display on which the messages should be shown for user guidance. If no information is to be displayed or if the card terminal does not have a display, this parameter can be omitted.

	`AltMVDMessages`	Is an optional parameter which specifies the various messages which should be shown on the display if the identification data are modified (see below for details). If this element is missing, standard texts are displayed.
	`OldReferenceData`	MAY contain the old reference data so that it is only necessary to enter the new data on the terminal. In this case the data is formatted by the calling layer. If this element is missing, the old reference data are to be entered on the terminal if necessary.
	`TimeoutUntilFirstKey`	Is an optional parameter which specifies the timeout in milliseconds before the first key actuation.
	`TimeoutAfterFirstKey`	Is an optional parameter which specifies the timeout in milliseconds after the first key actuation.
	`RepeatInput`	If this element is `TRUE`, repeated entry is mandatory. If the element is missing, it is not necessary to repeat the entry.
	`Template`	The acquired verification data are entered into the template before the data are sent to the eCard as APDU for the CHANGE REFERENCE DATA command (also refer to [ISO7816-4], Section 7.5.7) or the RESET RETRY COUNTER command (also refer to [ISO7816-4], Section 7.5.10).

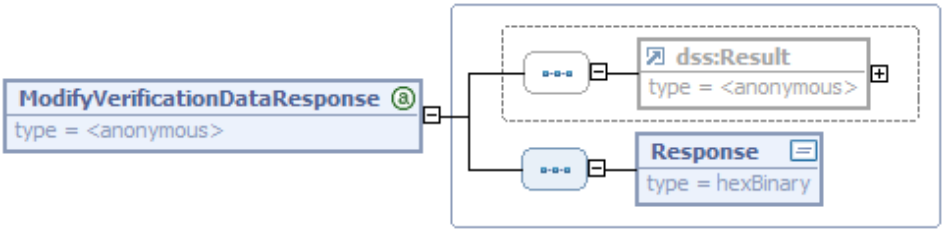


`AltMVDMessages` can be used to transfer alternative messages. Part of `ModifyVerificationData` (see above).

Name	Description
`AuthenticationRequestMessage`	Prompts the user to perform user verification (e.g. by entering a PIN). If this element is missing, the configured default message is used.
`SuccessMessage`	Shows the user that user verification was successful. If this element is missing, the configured default message is used.
`AuthenticationFailedMessage`	Shows the user that the user verification was not successful and that the eCard MAY be blocked. If this element is missing, the configured default message is used.
`EnterNewAuthenticationDataMessage`	If this element is missing, the configured default message is used.
`RepeatInputMessage`	If this element is missing, the configured default message is used.

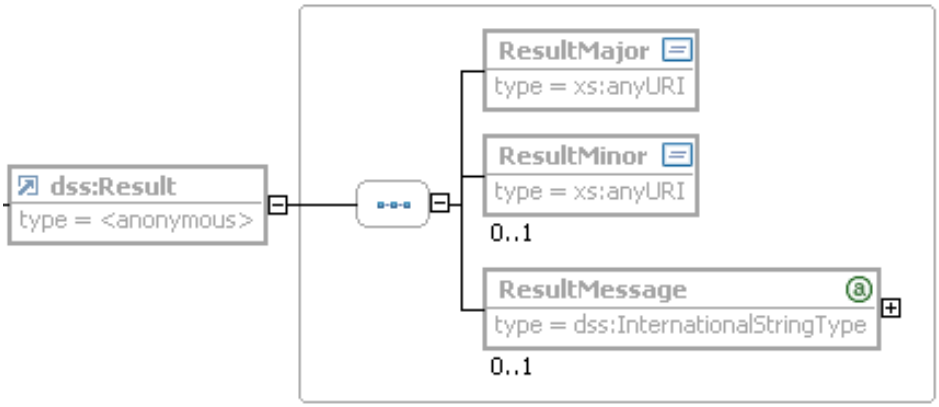
	`ComparisonOfRepeatedDataFailed`	If this element is missing, the configured default message is used.
	`RequestConfirmationMessage`	Prompts the user to repeat the entry. If this element is missing, the configured default message is used.
	`CancelMessage`	Shows the user that entry was canceled. If this element is missing, the configured default message is used.

Return



Return of the `ModifyVerificationData` function.

Name	Description
`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
`Response`	The response of the eCard.

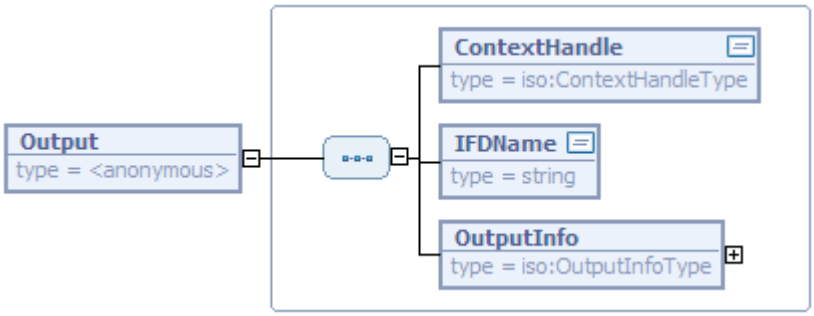


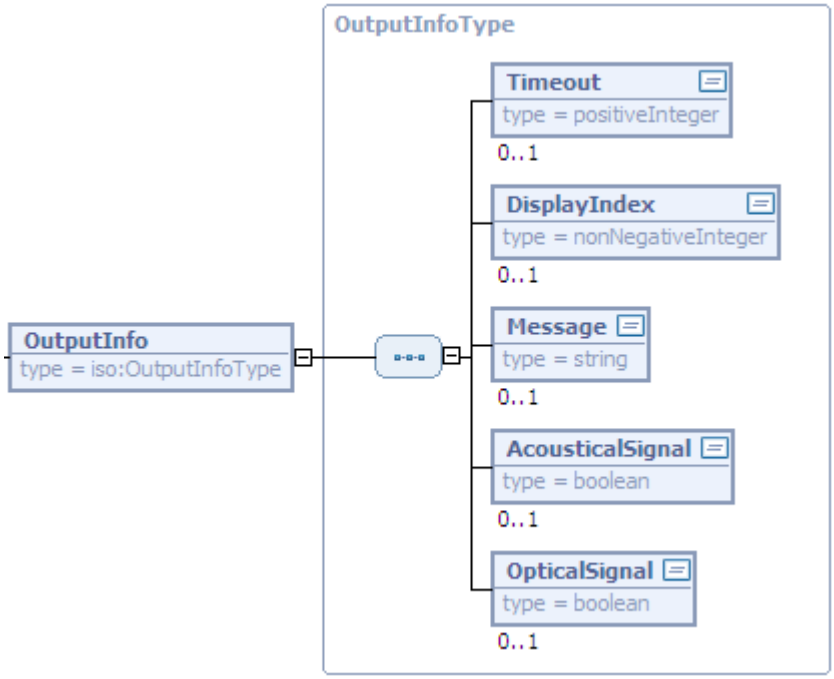
Status information and errors in `ModifyVerificationData`

Name	Error codes
`ResultMajor`	• /resultmajor#ok • /resultmajor#error

	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • /resultminor/ifdl/common#invalidSlotHandle • /resultminor/ifdl/IO#unknownInputUnit • /resultminor/ifdl#cancellationByUser • /resultminor/ifdl/IO#repeatedDataMismatch • /resultminor/ifdl/IO#unknownPINFormat • /resultminor/ifdl/IO#unknownBiometricSubtype • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Precondition	A connection to an eCard was established with `Connect`.	
Postcondition	The corresponding command - CHANGE REFERENCE DATA or RESET RETRY COUNTER in accordance with [ISO7816-4] – was executed on the card.	
Note	A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with FEATURE_MODIFY_PIN_START specified in [PC/SC], Part 10 (Section 2.9) using the PIN_MODIFY_STRUCTURE defined in [PC/SC], Part 10 (Section 2.6).	

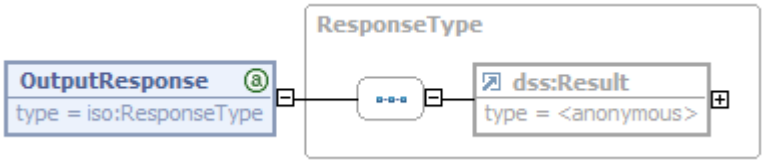
3.3.3 Output

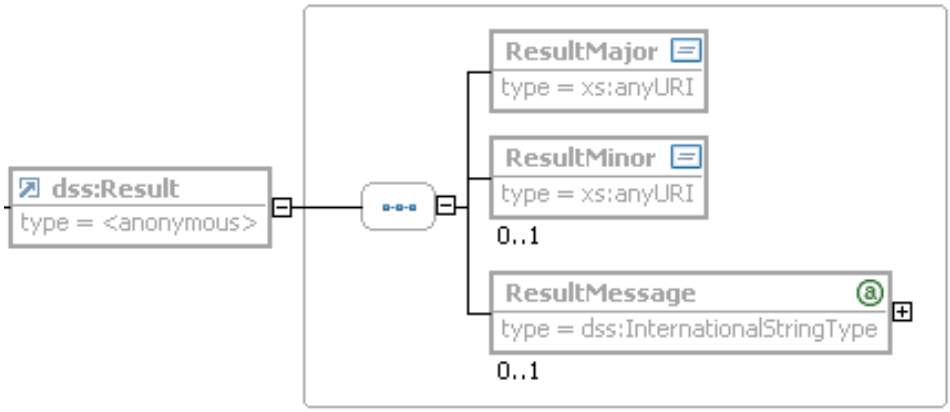
Name	**`Output`**	
Description	The command is used to output a message on the display of a card terminal and/or output a visual or acoustic signal on a card terminal.	
Invocation parameters	 Invocation of the `Output` function.	
	Name	**Description**
	`ContextHandle`	Handle with which the session with the Terminal-Layer is addressed.
	`IFDName`	Unique name of the card terminal.
	`OutputInfo`	Provides information about the output. Details are specified below.



The `OutputInfoType` is used in the specification of the `Output` function above and is used in addition in `CardApplicationConnect` (also refer to [TR-03112-4]).

Name	Description
`Timeout`	MAY specify how long the output in milliseconds is maintained. If this element is missing, the output remains until `Cancel` or `ReleaseContext` is invoked.
`DisplayIndex`	MAY specify the index of the display on which any existing message should be output. If there is only one display, the parameter MAY be omitted.
`Message`	Optionally contains the message which should be output.
`AcousticalSignal`	This optional parameter specifies whether an acoustic signal should be output. If the card terminal does not feature a suitable device, the parameter is ignored.
`OpticalSignal`	This optional parameter specifies whether a visual signal should be output. If the card terminal does not feature a suitable device, the parameter is ignored.

<table>
<tr><td>Return</td><td>


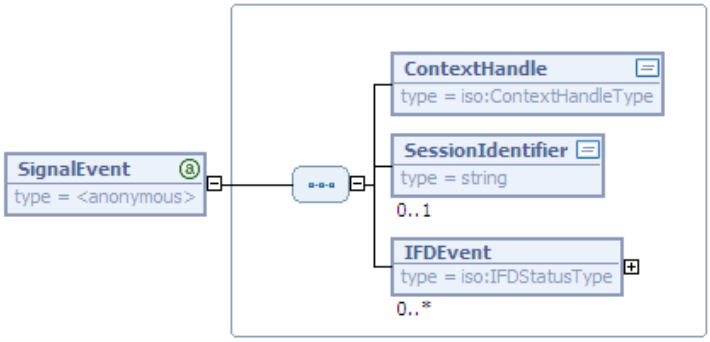
Return of the <code>Output</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>dss:Result</code></td><td>Contains the status information and the errors of an executed action. This element is described in more detail below.</td></tr>
</table>


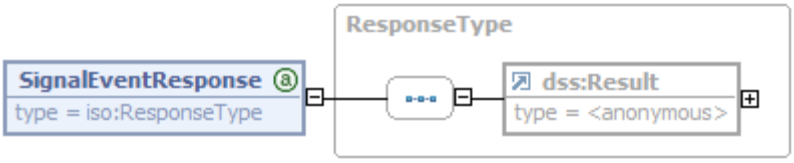
Status information and errors in <code>Output</code>.
<table>
<tr><th>Name</th><th>Error codes</th></tr>
<tr><td><code>ResultMajor</code></td><td>• /resultmajor#ok
• /resultmajor#error
• /resultmajor#warning</td></tr>
<tr><td><code>ResultMinor</code></td><td>• /resultminor/ifdl/common#timeoutError
• /resultminor/ifdl/common#invalidContextHandle
• /resultminor/ifdl/terminal#unknownIFD
• /resultminor/ifdl/IO#unknownDisplayIndex
• /resultminor/al/common#unknownError
• /resultminor/ifdl/IO#unknownOutputDevice</td></tr>
<tr><td><code>ResultMessage</code></td><td>MAY contain more detailed information on the error which occurred if required.</td></tr>
</table>
</td></tr>
<tr><td>Precondition</td><td>A context was established with <code>EstablishContext</code>.</td></tr>
<tr><td>Postcondition</td><td>The respective output on the card terminal is completed.</td></tr>
<tr><td>Note</td><td>A PC/SC handler in the IFD-Layer (also refer to Annex A) MAY implement this function with the <code>DisplayMessage</code> function specified in [PC/SC], Part 9 (Section 4.1.8.2).</td></tr>
</table>

3.4 IFD-Callback-Interface for card terminal events

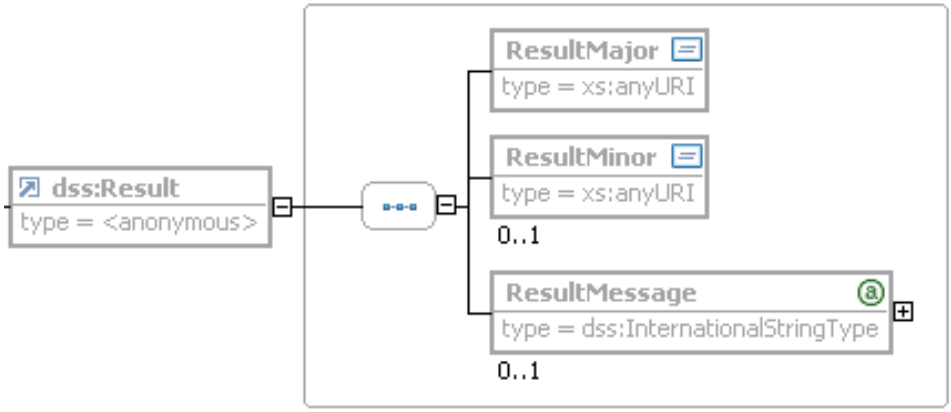
The IFD-Callback-Interface is made available from layers above the Terminal-Layer and contains exactly the function `SignalEvent`.

3.4.1 SignalEvent

<table>
<tr><td>Name</td><td><code>SignalEvent</code></td></tr>
<tr><td>Description</td><td>With the <code>SignalEvent</code> function layers above the Terminal-Layer can be informed of card terminal events. The Terminal-Layer was informed with the <code>Wait</code> function of the callback address provided for this invocation and additional necessary parameters.</td></tr>
<tr><td>Invocation parameters</td><td>


Invocation of the <code>SignalEvent</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>ContextHandle</code></td><td>Handle with which the session with the Terminal-Layer is addressed.</td></tr>
<tr><td><code>SessionIdentifier</code></td><td>Identifier transmitted during return of the <code>Wait</code> function.</td></tr>
<tr><td><code>IFDEvent</code></td><td><code>IFDEvent</code> MAY occur several times and contains information about an event which occurred on a card terminal. The parameter is of the <code>IFDStatusType</code> (refer to page 18), and contains the status information modified by the event.</td></tr>
</table>
</td></tr>
<tr><td>Return</td><td>


Return of the <code>SignalEvent</code> function.
<table>
<tr><th>Name</th><th>Description</th></tr>
</table>
</td></tr>
</table>

	`dss:Result`	Contains the status information and the errors of an executed action. This element is described in more detail below.
	 Status information and errors in `SignalEventResponse`	
	Name	**Error codes**
	`ResultMajor`	• /resultmajor#ok • /resultmajor#error
	`ResultMinor`	• /resultminor/ifdl/common#timeoutError • / resultminor/ifdl/common#invalidContextHandle • /resultminor/ifdl/terminal#unknownIFD • /resultminor/al/common#unknownError
	`ResultMessage`	MAY contain more detailed information on the error which occurred if required.
Preconditions	Registration for events was performed on specific terminals by means of `Wait`, whereby the respective callback address was specified.	
Postconditions	When `SignalEvent` is invoked, waiting for events on the specified card terminals is terminated and as a result the session identifier loses its validity.	
Notes	Also refer to `Wait` and `Cancel`.	

References

[TR-03112-2] BSI: TR-03112-2: eCard-API-Framework – Part 2: eCard-Interface

[TR-03112-3] BSI: TR-03112-3: eCard-API-Framework – Part 3: Management-Interface

[TR-03112-4] BSI: TR-03112-4: eCard-API-Framework – Part 4: ISO24727-3-Interface

[TR-03112-5] BSI: TR-03112-5: eCard-API Framework – Part 5: Suppor- Interface

[TR-03112-6] BSI: TR-03112-6: eCard-API-Framework – Part 6: IFD-Interface

[TR-03112-7] BSI: TR-03112-7: eCard-API-Framework – Part 7: Protocols

[TR-03119] BSI: TR-03119: Anforderungen an Chipkartenleser mit ePA-Unterstützung

[RFC2119] IETF: RFC 2119: S. Bradner: Key words for use in RFCs to Indicate Requirement Levels

[ISO19784-1] ISO: ISO/IEC 19784-1: Information technology — Biometric application programming interface — Part 1: BioAPI specification

[ISO24727-3] ISO: ISO/IEC 24727-3: Identification Cards — Integrated Circuit Cards Programming Interfaces — Part 3: Application Interface

[ISO7816-15] ISO: ISO/IEC 7816-15: Identification cards - Integrated circuit(s) cards with contacts — Part 15: Cryptographic information application

[ISO7816-4] ISO: ISO/IEC 7816-4: Identification cards — Integrated circuit cards — Part 4:Organization, security and commands for interchange

[PC/SC] PC/SC Workgroup: PC/SC Workgroup Specifications 1.0/2.0