## Contents

1  Overview of the eCard-API-Framework ................................................................. 5
1.1  Key Words ........................................................................................................ 5
1.2  XML-Schema .................................................................................................... 6
1.3  Overview (Part 1) ............................................................................................. 6
1.4  eCard-Interface (Part 2) .................................................................................. 6
1.4.1  Functions for identity management .............................................................. 6
1.4.2  Signature functions ....................................................................................... 6
1.4.3  Encryption functions .................................................................................... 7
1.5  Management-Interface (Part 3) ........................................................................ 7
1.5.1  Management of the eCard-API-Framework ................................................. 7
1.5.2  Card management ......................................................................................... 8
1.5.3  Card terminal management ......................................................................... 8
1.5.4  Trusted viewer management ...................................................................... 9
1.5.5  Identity management ................................................................................... 9
1.5.6  Service management ................................................................................... 10
1.6  ISO24727-3-Interface (Part 4) ......................................................................... 10
1.6.1  Card Application Service Access ................................................................. 10
1.6.2  Connection service ...................................................................................... 11
1.6.3  Card Application Service .......................................................................... 11
1.6.4  Named data service ..................................................................................... 12
1.6.5  Cryptographic service .................................................................................. 12
1.6.6  Differential identity service ........................................................................ 13
1.6.7  Authorization service .................................................................................. 13
1.7  Support-Interface (Part 5) ............................................................................... 14
1.8  IFD-Interface (Part 6) ..................................................................................... 14
1.8.1  Card terminal functions .............................................................................. 15
1.8.2  Card functions ............................................................................................. 15
1.8.3  User interaction functions ......................................................................... 16
1.8.4  IFD callback interface for card terminal events ........................................... 16
1.9  Protocols (Part 7) ........................................................................................... 16
2  Web Service interface ......................................................................................... 18
2.1  Structure of the WSDL files ......................................................................... 18
2.2  Web service binding ....................................................................................... 18
2.3  Structure of and links between the XSD and WSDL files .............................. 19
3  Other programming interfaces .......................................................................... 21
3.1  C-Language Binding for ExecuteAPI ............................................................. 21
3.2  Java-Language Binding for ExecuteAPI function ......................................... 22
4  Basic definitions ............................................................................................... 23
4.1  Request- and ResponseType ......................................................................... 23
4.1.1  RequestType ............................................................................................... 23
4.1.2  ResponseType ............................................................................................ 23
4.2  Codes for errors and warnings ....................................................................... 25
4.2.1  Error codes from the Application-Layer ..................................................... 25
4.2.2  Error codes from the Dispatcher ............................................................... 27
4.2.3  Error codes from the Identity-Layer .......................................................... 28
4.2.4 Error codes from the Service-Access-Layer ................................................................. 32
4.2.5 Error codes from the Terminal-Layer ........................................................................ 34
4.3 Additional URI definitions .......................................................................................... 36
4.3.1 Protocols .................................................................................................................. 36
4.3.2 Properties ................................................................................................................ 36
4.3.3 CardInfo .................................................................................................................. 36
4.4 Mandatory use of Profile-attribute in responses .......................................................... 36

5 List of Abbreviations ....................................................................................................... 37
6 Index of XML data types ............................................................................................... 38

Table of Figures

Figure 1: The dss:Result element ...................................................................................... 24
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 Management-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 Support-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 multiplicity of elements as compared to the schema definition.

1.3 Overview (Part 1)

This document is Part 1 of the document series, which provides an overview and general definitions.

1.4 eCard-Interface (Part 2)

The eCard-Interface encapsulates the main functions of the eCard-API-Framework in an application-orientated manner.

The eCard-Interface is specified in Part 2 of the document series and encapsulates the main functions of the eCard-API-Framework. For this purpose the eCard-Interface provides the following function groups:

- Functions for identity management
- Signature functions
- Encryption functions

With the GetCertificate function, certificate applications can be transferred to a certification authority, from where they obtain their certificates.

In addition, the invocations specified by [DSS] can be used for the creation and verification of (qualified) electronic signatures in the formats according to [RFC3275] and [RFC3369], as well as the corresponding extensions from ETSI. This functional group also contains an interface to a trustworthy display component which can be used in particular for the displaying the data and test results requiring a signature.

Finally, with the encryption functions documents can be easily encrypted and decrypted in accordance with [RFC3369] and [XMLEnc] by simple function invocations.

1.4.1 Functions for identity management

GetCertificate

With the GetCertificate function, certificate applications can be transferred to a certification authority, from where they obtain their certificates.

1.4.2 Signature functions

SignRequest

The SignRequest function conforms with [DSS], and permits the creation of (qualified) electronic signatures in popular high-level formats such as XML-DSSig in accordance with [RFC3275], or cryptographic message syntax in accordance with [RFC3369]. These signatures can also contain time stamps which can also be requested separately with this function.

VerifyRequest
The **VerifyRequest** function conforms with [DSS] and enables verification of signed objects (e.g. signatures, time stamps, certificates, CRLs, OCSP-responses and evidence records).

**ShowViewer**

This function enables trustworthy display of documents which can be used for the creation and verification of signatures.

### 1.4.3 Encryption functions

**EncryptRequest**

The **EncryptRequest** function enables encryption of data in accordance with [XMLEnc] and [RFC3369].

**DecryptRequest**

The **DecryptRequest** function enables decryption of data encrypted in accordance with [XMLEnc] and [RFC3369].

### 1.5 Management-Interface (Part 3)

The Management-Interface provides important administration functions for the eCard-API-Framework.

The Management-Interface provides the following function groups:

- Management of the eCard-API-Framework
- Card management
- Card terminal management
- Trusted viewer management
- Identity management
- Service management

#### 1.5.1 Management of the eCard-API-Framework

This function group includes functions for the management of the eCard-API framework itself:

**InitializeFramework**

The **InitializeFramework** function ensures initialisation of the eCard-API-Framework.

**TerminateFramework**

The **TerminateFramework** function terminates all sessions and services of the eCard-API-Framework.

**APIACLList**

The **APIACLList** function is OPTIONAL and MAY provide the currently defined access control regulations for access to the individual functions of the eCard-API-Framework. This function MAY ONLY be made available to the **administrator** who is authenticated in accordance with the security policies applicable for operation of the eCard framework.

**APIACLModify**
The ACLModify function is OPTIONAL and MAY be used to modify the access control rules which govern
the access to the functions of the eCard-API-Framework. Via this access control mechanism it is possible, for
example, to grant or refuse access of an application to the Transmit function in the IFD interface (also
refer to [TR-03112-6]) for the implementation of a "transparent channel" to a card. As a consequence, it is
also possible to define whether and under which circumstances remote eCard-API-Frameworks are permitted
access to a local eCard-API-Framework.

If this function is supported it MAY ONLY be made available to the administrator who is authenticated in
accordance with the security policies applicable for operation of the eCard framework.

FrameworkUpdate
The FrameworkUpdate function checks whether an update is available for the eCard-API-Framework
and performs such an update if necessary. The detailed processes during execution of this function are
protocol-specific (refer to [TR-03112-7]).

GetDefaultParameters
Default behaviour can be specified for the eCard-API-Framework to also permit the easiest possible
invocations by the client application for potentially complex operations (e.g. for creating and verifying
electronic signatures, refer to [TR-03112-2], section 3.2.1-3.2.2). The currently specified default parameters
MAY be read out with the GetDefaultParameters function.

SetDefaultParameters
The SetDefaultParameters function is used to write the default parameters, which then determine
the standard behaviour of the eCard-API-Framework.

1.5.2 Card management

GetCardInfoList
The GetCardInfoList function supplies the list of card types which are known from the CardInfo
files.

SetCardInfoList
The SetCardInfoList function saves an ordered list of card types in form of URIs, which determine the
steps during the card recognition procedure.

AddCardInfoFiles
With the AddCardInfoFiles function it is possible to add a series of CardInfo files.

DeleteCardInfoFiles
The DeleteCardInfoFiles function deletes a series of CardInfo files.

1.5.3 Card terminal management

RegisterIFD
With the RegisterIFD function it is possible to add a card terminal with all configuration information.

UnregisterIFD
The UnregisterIFD function deletes a card terminal.
1.5.4 Trusted viewer management

GetTrustedViewerList
The GetTrustedViewerList function provides a list of available trustworthy display components (trusted viewer).

GetTrustedViewerConfiguration
The GetTrustedViewerConfiguration function reads the configuration information for a specific trusted viewer which is saved in the eCard-API-Framework.

SetTrustedViewerConfiguration
The SetTrustedViewerConfiguration function writes the configuration information for a specific trusted viewer.

AddTrustedViewer
With the AddTrustedViewer function, a trusted viewer can be added with all configuration information.

DeleteTrustedViewer
The DeleteTrustedViewer function deletes a trusted viewer.

1.5.5 Identity management

GetTrustedIdentities
The GetTrustedIdentities function supplies a list of the trustworthy identities in form of trust-service status list (TSL) and trustworthy certificates.

AddTrustedCertificate
With the AddTrustedCertificate function, a certificate can be added to the list of trusted certificates.

AddCertificate
With the AddCertificate function, a non-trustworthy certificate which can be used for signature verification or encryption can be added to the certificate database.

ExportCertificate
With the ExportCertificate function, a (trustworthy or non-trustworthy) certificate can be exported.

DeleteCertificate
The DeleteCertificate function deletes an existing (trustworthy or non-trustworthy) certificate from the certificate database.

AddTSL
With the AddTSL function, a trust-service status list can be added to the eCard-API-Framework.

ExportTSL
With the ExportTSL function, a trust-service status list can be exported.

DeleteTSL
With the DeleteTSL function, a trust-service status list can be deleted from the list of trustworthy identities.
1.5.6 Service management

GetOCSPServices
The GetOCSPServices function reads the list of available OCSP responders together with the corresponding configuration information.

SetOCSPServices
The SetOCSPServices function writes the list of available OCSP responders together with the corresponding configuration information.

GetDirectoryServices
The GetDirectoryServices function reads the list of the directory services accessible via LDAP or HTTP with all corresponding configuration information.

SetDirectoryServices
The SetDirectoryServices function writes a list of the directory services accessible via LDAP or HTTP with all corresponding configuration information.

GetTSServices
The GetTSServices function reads the list of time stamp services with all corresponding configuration information.

SetTSServices
The SetTSServices function writes a list of the time stamp services together with all corresponding configuration information.

1.6 ISO24727-3-Interface (Part 4)

The ISO24727-3-Interface provides a generic interface for all card-based functions of the various eCards.

The ISO24727-3-Interface provides the following function groups:

- Card Application Service Access
- Connection Service
- Card Application Service
- Named data service
- Cryptographic service
- Authorization service

1.6.1 Card Application Service Access

Initialize
The Initialize function is executed when the ISO24727-3-Interface is invoked for the first time. The interface is initialised with this function.

Terminate
The Terminate function is executed when the ISO24727-3-Interface is terminated. This function closes all processes.
CardApplicationPath
The CardApplicationPath function determines the path between a client application and a card application.

1.6.2 Connection service

CardApplicationConnect
The CardApplicationConnect function establishes an unauthenticated connection between the client application and the card application.

CardApplicationDisconnect
The CardApplicationDisconnect function terminates the connection between the client application and the card application.

CardApplicationStartSession
The CardApplicationStartSession function starts an authenticated session between the client application and the card application.

CardApplicationEndSession
The CardApplicationEndSession function closes an authenticated session between the client application and the card application.

1.6.3 Card Application Service

CardApplicationList
The CardApplicationList function returns the list of available card applications of an eCard as a list.

CardApplicationCreate
The CardApplicationCreate function creates a new card application.

CardApplicationDelete
The CardApplicationDelete function deletes a card application on an eCard.

CardApplicationServiceList
The CardApplicationServiceList function returns a list of the available services of a card application on an eCard.

CardApplicationServiceCreate
The CardApplicationServiceCreate function creates a new service for the card application on an eCard.

CardApplicationServiceLoad
The CardApplicationServiceLoad function loads executable code, which can be executed within a service of a card application on the eCard.

CardApplicationServiceDelete
The CardApplicationServiceDelete function deletes a service in a card application on an eCard.

CardApplicationServiceDescribe
The invocation parameters of a service of a card application can be determined with the CardApplicationServiceDescribe function.

**ExecuteAction**

The ExecuteAction function permits the execution of an action of a service which has been loaded into a card application on an eCard with the CardApplicationServiceLoad function.

### 1.6.4 Named data service

#### DataSetList

The DataSetList function supplies a list of data sets in a card application on an eCard.

#### DataSetCreate

The DataSetCreate function creates a new data set in a selected card application on an eCard.

#### DataSetSelect

The DataSetSelect function selects a data set of a card application on an eCard.

#### DataSetDelete

The DataSetDelete function deletes a data set of a card application on an eCard.

#### DSIList

The DSIList function returns a list of data structures for interoperability (DSIs) in the currently selected data set of a card application.

#### DSICreate

The DSICreate function creates a DSI in the currently selected data set of a card application.

#### DSIDelete

The DSIDelete function deletes a DSI in the currently selected data set of a card application.

#### DSIWrite

The DSIWrite function writes specific content into a DSI in a currently selected data set of an application.

#### DSIRead

The DSIRead function reads the content of a DSI in the currently selected data set of a card application.

### 1.6.5 Cryptographic service

The detailed functionality of the cryptographic service is determined by the protocol of the differential identity employed. Various protocols and especially the Generic Cryptography protocol are defined in [TR-03112-7].

#### GetRandom

The GetRandom function returns a random number which can be used, for example, for authentication.

#### VerifySignature

The VerifySignature function checks a digital signature.

#### VerifyCertificate

12

Bundesamt für Sicherheit in der Informationstechnik
The **VerifyCertificate** function validates a certificate.

**Sign**
The **Sign** function generates a signature for a communicated binary message.

**Encipher**
The **Encipher** function encrypts a transmitted plain text.

**Decipher**
The **Decipher** function decrypts a transmitted cipher text.

**Hash**
The **Hash** function calculates the hash value of a transmitted message.

### 1.6.6 Differential identity service

The detailed functionality of the **DIDCreate**, **DIDGet**, **DIDUpdate** and **DIDAuthenticate** functions is determined by the protocol (also refer to [TR-03112-7]) of the employed differential identity.

**DIDList**
The **DIDList** function returns a list of the existing differential identities (DIDs) in the card application of an eCard.

**DIDCreate**
The **DIDCreate** function creates a new differential identity in a card application of an eCard.

**DIDGet**
The **DIDGet** function determines the publicly accessible information (e.g. key reference) of a differential identity in a card application of an eCard.

**DIDUpdate**
The **DIDUpdate** function generates a new key (marker) for a differential identity in a card application of an eCard.

**DIDDelete**
The **DIDDelete** function deletes a given differential identity in a card application of an eCard.

**DIDAuthenticate**
Using one or more differential identities, the **DIDAuthenticate** function executes an authentication protocol which is implicitly specified by these identities.

### 1.6.7 Authorization service

**ACLList**
The **ACLList** function returns the currently defined access control rules for accessing a card application.

**ACLModify**
The **ACLModify** function permits modification of a certain access control rule for access to a card application.
1.7 Support-Interface (Part 5)

The Support-Interface contains a series of supporting functions which are typically not executed on an eCard. This comprises the following functions.

**Encode**

The *Encode* function encodes data. This function is also used, for example, for data compression.

**Decode**

The *Decode* function decodes data. This function is also used, for example, for data decompression.

**ValidateXMLDocument**

The *ValidateXMLDocument* function validates an XML document on the basis of a schema.

**GetCardInfoOrACD**

The *GetCardInfoOrACD* function may be used to retrieve CardInfo-files as specified in [TR-03112-4] and [CEN15480-3] or equivalent Application Capability Descriptions (ACD) according to [ISO24727-2], which allow to perform the mapping of generic calls at the ISO24727-3-Interface to card-specific APDUs.

1.8 IFD-Interface (Part 6)

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]:

**RegisterIFD**

With the *RegisterIFD* function it is possible to add a card terminal with all configuration information.

**UnregisterIFD**

The *UnregisterIFD* function deletes a card terminal.
1.8.1 Card terminal functions

EstablishContext
The EstablishContext function opens a session with the Terminal-Layer and returns a ContextHandle which is used in other invoked functions to address this session.

ReleaseContext
The ReleaseContext function terminates a session with the Terminal-Layer addressed by means of a ContextHandle.

ListIFDs
With the ListIFDs function a list of available card terminals is returned to the calling layer.

GetIFDCapabilities
The GetIFDCapabilities function provides information about a specific card terminal and its functional units to the calling layer.

GetStatus
The GetStatus function determines the current status of the card terminal.

Wait
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.

Cancel
The Cancel function terminates the waiting procedure for card terminal events, or attempts to cancel processing of the last command sent by means of the current handle on a specific card terminal. In this case, success depends on the type of command and the point of time at which Cancel was invoked.

ControlIFD
The ControlIFD function sends a (proprietary) command to the card reader. 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.

1.8.2 Card functions

Connect
The Connect function activates an eCard registered by the IFD and returns a CardHandle with which it can be addressed in future.

Disconnect
The Disconnect function invalidates a CardHandle and optionally executes an additional operation (e.g. ejection of the eCard, if the IFD has the corresponding mechanical functionality).

BeginTransaction
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 cancelled and any modifications performed are reset.
1.8.3 User interaction functions

VerifyUser
The VerifyUser function verifies a user by means of a PIN or a biometric characteristic.

ModifyVerificationData
The ModifyVerificationData function modifies the identification data (PIN or biometric characteristic).

Output
The Output function serves to control the output units of a card terminal.

1.8.4 IFD callback interface for card terminal events

SignalEvent
With the SignalEvent function, layers above the Terminal-Layer can be informed about card terminal events. To this purpose the SignalEvent function MUST be provided as a web service by these layers.

1.9 Protocols (Part 7)


The ISO/IEC 24727 protocols define Crypto and Differential Identity Services for some authentication protocols as required for the use of typical signature cards, electronic health insurance cards, healthcare professional ID cards and the planned electronic identity cards:

PIN Compare: This protocol defines the authentication of a user performed by means of a PIN, which is also specified in abridged form in Annex A.9 of [ISO24727-3].

Mutual Authentication: This protocol is specified in similar form in Annex A.12 of [ISO24727-3], Section 16.1.10f [eGK-1] and Section 8.8 of [EN14890-1] and provides the framework for mutual authentication with the exchange of keys using symmetric algorithms.

Extended Access Control: This protocol specified in [TR-03110] forms the framework for mutual authentication with key exchange using the Extended Access Control protocol.

RSA Authentication: This protocol is specified in a similar form in Annex A.15 of [ISO24727-3], Section 16 of [eGK-1] and Section 8.4 of [EN14890-1] and provides the framework for mutual authentication with an optional exchange of keys using the RSA algorithm.

Generic Cryptography: This generic protocol specifies how cryptographic operations can be used independently of specific authentication procedures.
In addition to that, [TR-03112-7] defines the connection establishment for SOAP and PAOS bindings whereas the use of the PAOS binding requires a more complex process to establish the connection. In this context [TR-03112-7] defines the functions

TC_API_OPEN

The function TC_API_Open can be used to initiate the establishment of a connection with a specific binding (e.g. [PAOSv2.0]) between two systems.

TC_API_CLOSE

The function TC_API_Close MAY be used to actively close down a previously established connection between two systems.

StartPAOS

The function StartPAOS is used for the establishment of a PAOS connection according to [PAOSv2.0].
2 Web Service interface

[WSDL] is used for interface description so that web services can be defined independently of the transport layer.

2.1 Structure of the WSDL files

On the basis of the definition – independently of the transport layer – the WSDL files comprise, on the one hand, XML elements which describe the service abstractly, and, on the other one, elements which specify compliance with a concrete transport protocol.

The WSDL files comply with the [WSDL] specification and have the following structure:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://www.bsi.bund.de/eCard"
    ...
    <wsdl:types>
      ...
    </wsdl:types>
    ...
    <wsdl:message name="Function ">
      ...
    </wsdl:message>
    ...
    <wsdl:portType name="Interface">
      ...
    </wsdl:portType>
    ...
    <wsdl:binding name="IFD" type="ec:IFD">
      ...
    </wsdl:binding>
    <wsdl:service name="Interface">
      ...
    </wsdl:service>
</wsdl:definitions>
```

The name-space prefixes are defined in the definitions root element of the WSDL files. The following element types defines the data types. The defined XSD schemata are also integrated at this point. These constitute the basis for the message element. This contains a description of the message which is processed as an inquiry or output as a response. The portType element specifies a logical grouping of operations which are described by the operation element.

The binding element is used for defining protocols and data formats. This contains a description of all information required for depicting communication on an existing transport protocol. In addition, the binding also describes encoding of the messages which transmits the input and output values for the existing operations. The actual service is described in the service element. A service comprises at least one port element. A port specifies an address for a binding.

2.2 Web service binding

The following web service bindings MUST at least be supported for the interfaces defined by [TR-03112-WSDL]:

---

Bundesamt für Sicherheit in der Informationstechnik
• **http://schemas.xmlsoap.org/soap/http** - SOAP via HTTP according to [SOAPv1.1], Section 6
• **urn:liberty:paos:2006-08** – Reverse http Binding for SOAP (PAOS) according to [PAOSv2.0]

In addition, other bindings as well as language-specific interfaces derived from the [TR-03112-WSDL] WSDL descriptions MAY also be supported (also refer to Section 3).

While http-requests and SOAP-requests or responses result directly in a "standard binding" in accordance with [SOAPv1.1] (Section 6), with a "PAOS binding" the remote framework (in this case typically a server system) sends its SOAP requests with the respective http-responses to the local framework (in this case typically a client system).

As a result of this binding it is possible to send web service invocations to client systems which do not permit any incoming http-connections. The PAOS binding can, for example, be used for the electronic personal ID for authentication and identification on the internet.

### 2.3 Structure of and links between the XSD and WSDL files

There is a corresponding WSDL file (.wsdl) for each interface (the eCard-, Management-, ISO24727-3-, ISO24727-Protocols, Support-, IFD-, and IFDCallback-Interface) in which the corresponding (primary) schema files (.xsd) are integrated:

- **eCard.{xsd,wsdl}** – contains the definition of the structures for the eCard-Interface (also refer to Section 1.4 and [TR-03112-2]).
- **Management.{xsd,wsdl}** – contains the definition of the structures for the Management-Interface (also refer to Section 1.5 and [TR-03112-3]).
- **ISO24727-3.{xsd,wsdl}** – contains the definition of the structures for the ISO24727-3-Interface (also refer to Section 1.6, [TR-03112-4] and [ISO24727-3]).
- **ISO24727-Protocols.{xsd,wsdl}** – defines the specific card protocols and the functions for connection establishment according to [TR-03112-7] (TC_API_Open, TC_API_Close and StartPAOS).
- **Support.{xsd,wsdl}** – contains the definition of the structures for the Support-Interface (also refer to Section 1.7 and [TR-03112-5]).
- **ISOIFD.{xsd,wsdl}** – contains the definition of the structures for the IFD-Interface (also refer to Section 1.8, [TR-03112-6] and [ISO24727-4]).
- **ISOIFDCALLBACK.{xsd,wsdl}** – contains the definition of the structures for the IFD-Callback-Interface (also refer to Section 1.8.4 and [TR-03112-6]).

In addition, there are the following additional XSD- and WSDL-files, which correspond to supporting services:

- **UpdateService.wsdl** - defines the interface to an update service for the basic update protocol specified in [TR-03112-7].
- **CardInfoRepository.{xsd,wsdl}** – defines the interface to the CardInfo-Repository (also refer to [TR-03112-5], Section 3).

Moreover, the following (secondary) schemata exist within the eCard-API-Framework which are integrated into the above-mentioned schema files:

- **ISOCommon.xsd** – defines basic structures (refer to Section 4),
- **CardInfo.xsd** – defines the CardInfo-structure (also refer to [TR-03112-4], Annex A and [CEN15480-3]) and
• eCard-Protocols.xsd defines other protocols (e.g. for GetCertificate) (also refer to [TR-03112-7]).

These schemata developed in the scope of this Guideline are, in turn, for example based on the following standard schemata:

• ecdsa.xsd – contains definitions necessary for the handling of data structures related to elliptic curves.
• oasis-dssx-1.0-profiles-encryption-0.5.xsd – a profile for encryption and decryption.
• xmlers-schema-draft-v0.3.xsd – contains definitions for XML-based evidence records as used in in the comprehensive verification report profile.

An index of XML data types is provided as Annex of this document.
3 Other programming interfaces

In addition to the web service bindings mentioned in Section 2.2, language-specific programming interfaces derived from the WSDL specifications MAY be offered for the functions of the eCard-API-Framework.

3.1 C-Language Binding for ExecuteAPI

This function serves as a C-language binding for the functions defined in this specification. It is the only entry point for the C-programming language. It processes XML encoded function calls and returns XML encoded function call responses.

Prototype

```c
wchar_t* ExecuteAPI (  
    IN const wchar_t* FunctionCall,  
    IN/OUT StatusType* Status  
);  
```

Parameters

FunctionCall  
XML encoded function call as a Null-terminated buffer containing the XML payload.

Status  
Receives the status of the function call to be executed. This parameter can have one of the following values:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API_OK</td>
<td>Function executed successfully.</td>
</tr>
<tr>
<td>API_SMALL_BUFFER</td>
<td>Application allocated buffer for response data is too small.</td>
</tr>
<tr>
<td>API_XML_ERROR</td>
<td>Error in parsing XML encoded function call.</td>
</tr>
<tr>
<td>API_GENERIC_ERROR</td>
<td>Unknown error.</td>
</tr>
</tbody>
</table>

Return values

Upon successful completion, the function returns the XML encoded function call response as a Null-terminated buffer.

Prototype

```c
void freeAPIExecuteResponse (  
    IN wchar_t* Response);  
```

Parameters

Response  
A pointer to a Null-terminated buffer containing the XML encoded function call response to be freed.

Return values

None.
3.2 Java-Language Binding for ExecuteAPI function

The APIJavaBinding class serves as a Java-language binding for the functions defined in this document series. It has only one method ExecuteAPI which processes XML encoded function calls and returns XML encoded function call responses.

Prototype

```java
public class APIJavaBinding {
    public static String ExecuteAPI (String functionCall) throws APIException {}
}
```

Parameters

- functionCall
  XML encoded function call as a Java String.

Returns

- XML encoded function call response as a Java String.

Throws

```java
public class APIException extends Exception {
    public enum APIExceptionReason {ApiXmlError, ApiGenericError};

    APIExceptionReason reason;
    public APIException(APIExceptionReason reason) {
        this.reason = reason;
    }
    public APIExceptionReason getReason() {
        return reason;
    }
    public String getMessage() {
        switch (reason) {
            case ApiXmlError:
                return "Error in parsing XML encoded function call.";
            default:
                return "Generic error.";
        }
    }
}
```
4 Basic definitions

4.1 Request- and ResponseType

The RequestType used in the eCard-API-Framework and in ISO/IEC 24727 is based on the RequestBaseType defined in the [DSS] (with restrictions), and as a ResponseType uses the ResponseBaseType defined in [DSS]. Thus, trouble-free combination of the interfaces from [ISO24727-3] and [ISO24727-4] is possible with those of [DSS] and related profiles.

4.1.1 RequestType

The RequestType used in the eCard-API-Framework and in ISO/IEC 24727 is defined as follows:

```
<complexType name="RequestType">
  <complexContent>
    <restriction base="dss:RequestBaseType">
      </restriction>
  </complexContent>
</complexType>
```

4.1.2 ResponseType

The ResponseType used in the eCard-API-Framework and in ISO/IEC 24727 is based on [DSS] and is defined as follows:

```
<complexType name="ResponseType">
  <complexContent>
    <restriction base="dss:ResponseBaseType">
      <sequence>
        <element ref="dss:Result"/>
      </sequence>
    </restriction>
  </complexContent>
</complexType>
```

As a result, of the functions standardised by ISO/IEC only the dss:Result element of [DSS] is used in the ResponseType, which contains the respective status information and is described in more detail below.

With the [DSS] compliant functions in [CEN15480-3], the ResponseType is used directly from [DSS] so that the optional parameter dss:OptionalOutputs is also available.
The Result element contains a mandatory ResultMajor element as a URI and MAY additionally contain a ResultMinor element as an URI, and with the ResultMessage, also an additional message. The URIs used in the ResponseType of the eCard-API-Framework have the prefix http://www.bsi.bund.de/ecard/api/1.1, which is supplemented by a corresponding identifier.

The following URIs therefore result for ResultMajor, the meaning of which is described in detail below:

- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#ok
- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#error
- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#warning

The error messages in the ResultMinor element have the prefix http://www.bsi.bund.de/ecard/api/1.1/resultminor/, and are supplemented by the respective error code.

The URI http://www.bsi.bund.de/ecard/api/1.1/resultminor/al/common#noPermission shows, for example, that the use of the function by the Client application is not permitted (refer to Section 4.2).

### Result Major

The following values are defined for the ResultMajor element:

- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#ok
  No error occurred during execution of the operation.

- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#error
  An error occurred during execution of the operation. The reason for the error is shown by the URI (or the respective error code) for ResultMinor.

- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#warning
  If the result of the operation is in principle OK, but there is a detail which may require closer investigation, a warning is given as a response.

- http://www.bsi.bund.de/ecard/api/1.1/resultmajor#nextRequest
  This result appears if at least one more request is expected within a protocol.

Stating the ResultMajor element is REQUIRED. If no error has occurred, the message …/resultmajor#ok is output. If, however, a warning or error has occurred, a description MUST be given in the ResultMinor element.

### Result Minor

If the ResultMajor-element contains the value …/resultmajor#error or …/resultmajor#warning is displayed in the ResultMajor element, a more detailed description MUST be given with the ResultMinor element. The URIs returned in this element are defined in Section 4.2.

### Result Message

MAY contain additional URIs if necessary (if several errors or warnings occur)
4.2 Codes for errors and warnings

The occurring errors are assigned to the levels of the eCard-API-Framework with the aid of the prefixes of the URI as follows:

- \texttt{http://www.bsi.bund.de/ecard/api/1.1/resultminor/al} - The cause of the error must be assigned to the application layer.
- \texttt{http://www.bsi.bund.de/ecard/api/1.1/resultminor/dp} - The cause of the error must be assigned to the dispatcher.
- \texttt{http://www.bsi.bund.de/ecard/api/1.1/resultminor/il} - The cause of the error must be assigned to the identity layer.
- \texttt{http://www.bsi.bund.de/ecard/api/1.1/resultminor/sal} - The cause of the error must be assigned to the service access layer.
- \texttt{http://www.bsi.bund.de/ecard/api/1.1/resultminor/ifdl} - The cause of the error must be assigned to the Terminal-Layer.

In addition, the errors or warnings are grouped in the respective interface, if necessary by an additional directory level.

The error descriptions stated in round brackets (API_xyz) correspond to those defined in [CEN15480-3], [ISO24727-3] or [ISO24727-4].

4.2.1 Error codes from the Application-Layer

The error codes assigned to this layer indicate operation errors of the applications using the eCard-API-Framework. These include in particular incorrect or faulty (configuration) data.

4.2.1.1 CardInfo

The error codes of the errors which can occur in conjunction with faulty invocations of the management functions for CardInfo structures are grouped in the CardInfo category.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{../al/CardInfo#addNotPossible}</td>
<td>CardInfo file cannot be added</td>
</tr>
<tr>
<td>\texttt{../al/CardInfo#notExisting}</td>
<td>CardInfo file does not exist</td>
</tr>
<tr>
<td>\texttt{../al/CardInfo#deleteNotPossible}</td>
<td>CardInfo file cannot be deleted</td>
</tr>
<tr>
<td>\texttt{../al/CardInfo#alreadyExisting}</td>
<td>The CardInfo file already exists</td>
</tr>
<tr>
<td>\texttt{../al/CardInfo#incorrectFile}</td>
<td>The CardInfo file is incorrect</td>
</tr>
</tbody>
</table>
4.2.1.2 Common

This section describes error codes caused by incorrect use of the API functions in general and which cannot be assigned to any particular range of functions.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
</table>
| …/al/common#unknownError | There was some unknown error  
An unexpected error has occurred during processing which cannot be represented by the standard codes or specific service error codes. The error and detail texts can describe the error more closely.  
(API_UNKNOWN_ERROR) |
| …/al/common#noPermission | Use of the function by the client application is not permitted |
| …/al/common#internalError | Internal error |
| …/al/common#incorrectParameter | Parameter error  
There was some problem with a provided or omitted parameter.  
(API_INCORRECT_PARAMETER) |
| …/al/common#unknownAPIFunction | API function unknown |
| …/al/common#notInitialized | Framework or layer not initialised |
| …/al/FrameworkUpdate#serviceNotAvailable | Update service is not accessible |
| …/al/FrameworkUpdate#unknownModule | Unknown module |
| …/al/FrameworkUpdate#invalidVersionNumber | Invalid version number for module |
| …/al/FrameworkUpdate#operationSystemNotSupported | Operating system not supported |
| …/al/FrameworkUpdate#noSpaceAvailable | No available space |
| …/al/FrameworkUpdate#securityConditionsNotSatisfied | Access denied |

4.2.1.3 FrameworkUpdate

This section describes the error codes which can occur during execution of a framework update.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>…/al/FrameworkUpdate#serviceNotAvailable</td>
<td>Update service is not accessible</td>
</tr>
<tr>
<td>…/al/FrameworkUpdate#unknownModule</td>
<td>Unknown module</td>
</tr>
<tr>
<td>…/al/FrameworkUpdate#invalidVersionNumber</td>
<td>Invalid version number for module</td>
</tr>
<tr>
<td>…/al/FrameworkUpdate#operationSystemNotSupported</td>
<td>Operating system not supported</td>
</tr>
<tr>
<td>…/al/FrameworkUpdate#noSpaceAvailable</td>
<td>No available space</td>
</tr>
<tr>
<td>…/al/FrameworkUpdate#securityConditionsNotSatisfied</td>
<td>Access denied</td>
</tr>
</tbody>
</table>
### 4.2.1.4 IFD

The following errors can occur as the result of incorrect information during management of card terminals. These errors are grouped in the IFD category.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../al/IFD#writeConfigurationNotPossible</td>
<td>The card terminal configuration cannot be written</td>
</tr>
<tr>
<td>.../al/IFD#couldNotAdd</td>
<td>The card terminal cannot be added</td>
</tr>
<tr>
<td>.../al/IFD#deleteNotPossible</td>
<td>The card terminal cannot be deleted</td>
</tr>
<tr>
<td>.../al/IFD#addNotPossible</td>
<td>The card terminal already exists</td>
</tr>
</tbody>
</table>

### 4.2.1.5 Trusted viewer

The following errors can occur during use of the management functions for trusted viewers.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../al/TrustedViewer#deleteNotPossible</td>
<td>The trusted viewer cannot be deleted</td>
</tr>
<tr>
<td>.../al/TrustedViewer#invalidID</td>
<td>Invalid TrustedViewerId</td>
</tr>
<tr>
<td>.../al/TrustedViewer#invalidConfiguration</td>
<td>Invalid configuration information for the trusted viewer</td>
</tr>
<tr>
<td>.../al/TrustedViewer#alreadyExisting</td>
<td>The trusted viewer already exists with the entered ID</td>
</tr>
</tbody>
</table>

### 4.2.1.6 TSL

The following error codes are generated by the functions for management of trust service status lists (TSLs).

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../al/TSL#TSLSequenceNumberIgnoredWarning</td>
<td>TSLSequenceNumber has been ignored</td>
</tr>
<tr>
<td>.../al/TSLSequenceNumberIgnoredWarning</td>
<td>As only a TSLSequenceNumber but no SchemeName has been specified, the TSLSequenceNumber-element has been ignored.</td>
</tr>
</tbody>
</table>

### 4.2.2 Error codes from the Dispatcher

The errors caused by communication to and from the eCard-API-Framework are described here.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../dp#timeoutError</td>
<td>Time exceeded (timeout)</td>
</tr>
</tbody>
</table>
The operation was terminated as the set time was exceeded.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../dp#invalidChannelHandle</td>
<td>Invalid channel handle</td>
</tr>
<tr>
<td>.../dp#communicationError</td>
<td>Communication error (API_COMMUNICATION_FAILURE)</td>
</tr>
<tr>
<td>.../dp#trustedChannelEstablishmentFailed</td>
<td>Failure to open a trusted channel</td>
</tr>
<tr>
<td>.../dp#unknownProtocol</td>
<td>Unknown protocol</td>
</tr>
<tr>
<td>.../dp#unknownCipherSuite</td>
<td>Unknown cipher suite</td>
</tr>
<tr>
<td>.../dp#unknownWebserviceBinding</td>
<td>Unknown web service binding</td>
</tr>
<tr>
<td>.../dp#nodeNotReachable</td>
<td>Node not reachable</td>
</tr>
</tbody>
</table>

### 4.2.3 Error codes from the Identity-Layer

Errors caused by the identity layer of the eCard-API-Framework are assigned to the identity layer.

#### 4.2.3.1 Algorithm

The errors of the identity layer caused by the algorithms stated for use are described in the following table.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../il/algorithcassertAlgorithmNotSupported</td>
<td>Stated hash algorithm is not supported</td>
</tr>
<tr>
<td>.../il/algorithmsignatureAlgorithmNotSupported</td>
<td>The stated signature algorithm is not supported</td>
</tr>
</tbody>
</table>

#### 4.2.3.2 CertificateRequest

The errors of the identity layer which can occur when a certificate is requested are grouped in this category. The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../il/certificateRequest#unknownAttribute</td>
<td>Unknown attribute in the certificate application</td>
</tr>
<tr>
<td>.../il/certificateRequest#creationOfCertificateRequestFailed</td>
<td>Generation of the certificate application failed</td>
</tr>
<tr>
<td>.../il/certificateRequest#submissionFailed</td>
<td>Submission of the certificate application failed</td>
</tr>
<tr>
<td>.../il/certificateRequest#unknownTransactionID</td>
<td>Unknown transaction identifier</td>
</tr>
<tr>
<td>.../il/certificateRequest#certificateDownloadFailed</td>
<td>Not possible to collect the certificate</td>
</tr>
<tr>
<td>.../il/certificateRequest#subjectMissing</td>
<td>No subject specified in request</td>
</tr>
</tbody>
</table>
4.2.3.3 Encryption

The Encryption category contains the errors of the identity layer which occur during encryption or decryption.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../il/encryption#encryptionOfCertainNodes</td>
<td>Specific nodes can only be encrypted in case of an XML document</td>
</tr>
<tr>
<td>OnlyForXMLDocuments</td>
<td></td>
</tr>
<tr>
<td>../il/encryption#encryptionFormatNotSupported</td>
<td>The encryption format is not supported</td>
</tr>
<tr>
<td>../il/encryption#invalidCertificate</td>
<td>The encryption certificate of an intended recipient is invalid</td>
</tr>
</tbody>
</table>

4.2.3.4 Key

The errors of the identity layer which can occur when a key is generated or keys are used are grouped in this category.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../il/key#keyGenerationNotPossible</td>
<td>Key generation is not possible</td>
</tr>
<tr>
<td>../il/key#encryptionAlgorithmNotSupported</td>
<td>The stated encryption algorithm is not supported</td>
</tr>
</tbody>
</table>

4.2.3.5 Service

The Service category contains the errors of the identity layer which occur due to the non-accessibility of the service to be used.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../il/service#ocspResponderUnreachable</td>
<td>The OCSP responder is inaccessible</td>
</tr>
<tr>
<td>../il/service#directoryServiceUnreachable</td>
<td>The directory service is inaccessible</td>
</tr>
<tr>
<td>../il/service#timeStampServiceUnreachable</td>
<td>The time stamp service is inaccessible</td>
</tr>
</tbody>
</table>

4.2.3.6 Signature

All errors and warnings of the identity layer which occur during signature generation or signature verification are assigned to the Signature category.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../il/signature#signatureFormatNotSupported</td>
<td>The signature format is not supported</td>
</tr>
<tr>
<td></td>
<td>The stated signature or time stamp format is not supported.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>../il/signature#PDFSignatureForNonPDFDocument</td>
<td>PDF signature for non-PDF document requested</td>
</tr>
<tr>
<td>../il/signature#unableToIncludeEContentWarning</td>
<td>IncludeEContent not possible</td>
</tr>
<tr>
<td></td>
<td>This warning is returned if the IncludeEContent flag is set when a PDF signature or a time stamp is generated, or when a hash value is transmitted for signature generation.</td>
</tr>
<tr>
<td>../il/signature#ignoredSignaturePlacementFlagWarning</td>
<td>The SignaturePlacement flag was ignored</td>
</tr>
<tr>
<td></td>
<td>This warning is returned when the SignaturePlacement flag was set for a non-XML-based signature.</td>
</tr>
<tr>
<td>../il/signature#certificateNotFound</td>
<td>The certificate is not available</td>
</tr>
<tr>
<td></td>
<td>The stated certificate is not available for the function. This could be due to an incorrect reference or a deleted data field.</td>
</tr>
<tr>
<td>../il/signature#certificateFormatNotCorrect</td>
<td>The certificate cannot be interpreted</td>
</tr>
<tr>
<td></td>
<td>The format of the stated certificate is unknown and cannot be interpreted.</td>
</tr>
<tr>
<td>../il/signature#invalidCertificateReference</td>
<td>Invalid certificate reference</td>
</tr>
<tr>
<td>../il/signature#certificateChainInterrupted</td>
<td>The certificate chain is interrupted</td>
</tr>
<tr>
<td></td>
<td>The stated certificate chain is interrupted. It is therefore not possible to complete full verification up to the root certificate.</td>
</tr>
<tr>
<td>../il/signature#resolutionOfObjectReferenceImpossible</td>
<td>It was not possible to resolve the object reference</td>
</tr>
<tr>
<td>../il/signature#transformationAlgorithmNotSupported</td>
<td>The transformation algorithm is not supported</td>
</tr>
<tr>
<td>../il/signature#unknownViewer</td>
<td>The viewer is unknown or not available</td>
</tr>
<tr>
<td>../il/signature#certificatePathNotValidated</td>
<td>The certificate path was not checked</td>
</tr>
<tr>
<td></td>
<td>Due to some problems it was not possible to validate the certificate path.</td>
</tr>
<tr>
<td>../il/signature#certificateStatusNotChecked</td>
<td>The certificate status was not checked</td>
</tr>
<tr>
<td></td>
<td>Due to some problems it was not possible to check the certificate status.</td>
</tr>
<tr>
<td>../il/signature#signatureManifestNotCheckedWarning</td>
<td>The signature manifest was not verified</td>
</tr>
<tr>
<td></td>
<td>This is a warning.</td>
</tr>
<tr>
<td>../il/signature#suitabilityOfAlgorithmsNotChecked</td>
<td>The suitability of the signature and hash algorithms was not checked</td>
</tr>
<tr>
<td>../il/signature#detachedSignatureWithoutEContent</td>
<td>No signature-related data were found (detached signature without EContent)</td>
</tr>
<tr>
<td>Error Code</td>
<td>Error Message</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>.../il/signature/improperRevocationInformation</td>
<td>It is not possible to interpret revocation information</td>
</tr>
<tr>
<td>.../il/signature/invalidSignatureFormat</td>
<td>The signature format is incorrect. The format of the transmitted signature does not correspond to the respective specification. This error occurs when a supported format is recognised (e.g. in accordance with [RFC3275] or [RFC3369]), but the signature does not meet the respective form requirements. If the transmitted format was already not recognised, error / .../il/signature/signatureFormatNotSupported is returned.</td>
</tr>
<tr>
<td>.../il/signature/signatureAlgorithmNotSuitable</td>
<td>The security of the signature algorithm is not suitable at the relevant point of time.</td>
</tr>
<tr>
<td>.../il/signature/hashAlgorithmNotSuitable</td>
<td>The security of the hash algorithm is not suitable at the relevant point of time.</td>
</tr>
<tr>
<td>.../il/signature/invalidCertificatePath</td>
<td>The certificate path is invalid</td>
</tr>
<tr>
<td>.../il/signature/certificateRevoked</td>
<td>The certificate has been revoked</td>
</tr>
<tr>
<td>.../il/signature/referenceTimeNotWithinCertificateValidityPeriod</td>
<td>The reference time is outside the validity period of a certificate</td>
</tr>
<tr>
<td>.../il/signature/invalidCertificateExtension</td>
<td>Invalid extensions in a certificate</td>
</tr>
<tr>
<td>.../il/signature/signatureManifestNotCorrect</td>
<td>Verification of a signature manifest has failed</td>
</tr>
<tr>
<td>.../il/signature/signatureTypeDoesNotSupportSignatureFormClarificationWarning</td>
<td>The stated SignatureType does not support SignatureForm parameter</td>
</tr>
<tr>
<td>.../il/signature/unknownSignatureForm</td>
<td>Unknown SignatureForm</td>
</tr>
<tr>
<td>.../il/signature/includeObjectOnlyForXMLSignatureAllowedWarning</td>
<td>IncludeObject only permitted with XML signatures</td>
</tr>
<tr>
<td>.../il/signature/xPathEvaluationError</td>
<td>It was not possible to resolve the XPath expression</td>
</tr>
<tr>
<td>.../il/signature/wrongMessageDigest</td>
<td>Wrong message digest. The calculated digest of the message is not equal to the message digest in the MessageDigest-attribute of the CMS-Signature or the DigestValue-element of the XML-signature respectively.</td>
</tr>
<tr>
<td>.../il/signature/IFDInconsistency</td>
<td>IFD inconsistency. The compatibility-check with the present IFD failed.</td>
</tr>
</tbody>
</table>
4.2.3.7 Viewer

The Viewer category contains the errors which occur in connection with the trusted viewer within the identity layer.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../il/viewer#unsuitableStylesheetForDocument</td>
<td>Unsuitable stylesheet for transmitted document</td>
</tr>
<tr>
<td>.../il/viewer#cancellationByUser</td>
<td>Cancellation by the viewer</td>
</tr>
<tr>
<td>.../il/viewer#timeout</td>
<td>Time exceeded (timeout)</td>
</tr>
<tr>
<td></td>
<td>The operation was terminated as the set time was exceeded.</td>
</tr>
<tr>
<td>.../il/viewer#viewerMessageTooLong</td>
<td>The ViewerMessage is too long</td>
</tr>
</tbody>
</table>

4.2.4 Error codes from the Service-Access-Layer

Errors resulting from the Support-Interface or the ISO24727-3-Interface are assigned to the Service-Access-Layer.

4.2.4.1 Support-Interface

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../sal/support#encodingError</td>
<td>Encoding not possible</td>
</tr>
<tr>
<td>.../sal/support#decodingError</td>
<td>Decoding not possible</td>
</tr>
<tr>
<td>.../sal/support#schemaValidationError</td>
<td>Validation of the schema has failed</td>
</tr>
<tr>
<td>.../sal/support#schemaValidationWarning</td>
<td>A warning occurred during validation of the schema.</td>
</tr>
<tr>
<td>.../sal/support#noAppropriateSchema</td>
<td>No suitable schema is available</td>
</tr>
<tr>
<td>.../sal/support#cardInfoRepositoryUnreachable</td>
<td>The CardInfo repository server is not accessible</td>
</tr>
</tbody>
</table>

4.2.4.2 ISO24727-3-Interface

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../sal#cancellationByUser</td>
<td>Cancellation by the user</td>
</tr>
</tbody>
</table>
A necessary user intervention (e.g. PIN entry or confirmation of the signature generation in the trusted viewer) was terminated by cancellation.  
(API_CANCELLATION_BY_USER)

**/sal#nameExists**  
The name already exists  
(API_NAME_EXISTS)

**/sal#prerequisitesNotSatisfied**  
The prerequisite is not met  
(API_PREREQUISITES_NOT_SATISFIED)

**/sal#protocolNotRecognized**  
Unknown protocol  
(API_PROTOCOL_NOT_RECOGNIZED)

**/sal#inappropriateProtocolForAction**  
Unsuitable protocol for the required action  
(API_INAPPROPRIATE_PROTOCOL_FOR_ACTION)

**/sal#invalidSignature**  
The verified signature is not valid  
(API_INVALID_SIGNATURE)

**/sal#invalidKey**  
The selected key is not valid  
(API_INVALID_KEY)

**/sal#notInitialized**  
No initialisation carried out  
The used operation requires initialisation  
(API_NOT_INITIALIZED)

**/sal#tooManyResults**  
Warning - Too many results  
(API_TOO_MANY_RESULTS)

**/sal#warningConnectionDisconnected**  
Warning - The connection has been disconnected  
(API_WARNING_CONNECTION_DISCONNECTED)

**/sal#warningSessionEnded**  
Warning – An established session was terminated  
(API_WARNING_SESSION_ENDED)

**/sal#namedEntityNotFound**  
The name does not exist  
The stated name of a card application service, DID, Data Set etc. does not exist.  
(API_NAMED_ENTITY_NOT_FOUND)

**/sal#insufficientResources**  
The resources are insufficient  
(API_INSUFFICIENT_RESOURCES)

**/sal#securityConditionsNotSatisfied**  
Access denied  
(API_SECURITY_CONDITION_NOT_SATISFIED)
Exclusive reservation is not possible
Exclusive reservation of the eCard is not possible, because other applications are currently accessing the eCard.
(API_EXCLUSIVE_NOT_AVAILABLE)

Warning - there is no active session
This warning indicates that there is no active session, which can be terminated with CardApplicationEndSession.

Decryption not possible
No suitable keys for decryption found

Invalid access control information

Unknown protocol
The requested protocol is unknown

Unknown card type
Unknown card type specified

Unknown DID name
Unknown DID name specified

File not found

### 4.2.5 Error codes from the Terminal-Layer

Errors which occur in connection with the card terminal are assigned to the Terminal-Layer.

#### 4.2.5.1 Common

The errors of the Terminal-Layer which are not assigned to any specific function are grouped in the Common category.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../ifdl/common#timeoutError</td>
<td>Time exceeded (timeout)</td>
</tr>
<tr>
<td></td>
<td>The operation was terminated as the set time was exceeded.</td>
</tr>
<tr>
<td></td>
<td>(API_TIMEOUT_ERROR)</td>
</tr>
<tr>
<td>../ifdl/common#invalidContextHandle</td>
<td>Unknown context handle</td>
</tr>
<tr>
<td></td>
<td>(IFD_INVALID_CONTEXT_HANDLE)</td>
</tr>
<tr>
<td>../ifdl/common#cancellationByUser</td>
<td>Cancellation by the user</td>
</tr>
<tr>
<td></td>
<td>A necessary user intervention (e.g. PIN entry) was terminated by cancellation.</td>
</tr>
</tbody>
</table>
### 4.2.5.2 IO

The errors of the Terminal-Layer which occur in connection with the input or output of data on the terminal are grouped in the IO category.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../ifdl/IO#unknownInputUnit</td>
<td>Unknown input unit</td>
</tr>
<tr>
<td></td>
<td>(IFD_UNKNOWN_INPUT_UNIT)</td>
</tr>
<tr>
<td>../ifdl/IO#unknownDisplayIndex</td>
<td>Unknown display index</td>
</tr>
<tr>
<td></td>
<td>(IFD_UNKNOWN_DISPLAY_INDEX)</td>
</tr>
<tr>
<td>../ifdl/IO#cancelNotPossible</td>
<td>It is not possible to cancel the command</td>
</tr>
<tr>
<td></td>
<td>(IFD_CANCEL_NOT_POSSIBLE)</td>
</tr>
<tr>
<td>../ifdl/IO#noTransactionStarted</td>
<td>No smart card transaction has been started</td>
</tr>
<tr>
<td></td>
<td>(IFD_NO_TRANSACTION_STARTED)</td>
</tr>
<tr>
<td>../ifdl/IO#repeatedDataMismatch</td>
<td>Newly recorded identification data do not correspond</td>
</tr>
<tr>
<td></td>
<td>(IFD_REPEATED_DATA_MISMATCH)</td>
</tr>
<tr>
<td>../ifdl/IO#unknownPINFormat</td>
<td>Unknown pin format</td>
</tr>
<tr>
<td></td>
<td>(IFD_UNKNOWN_PIN_FORMAT)</td>
</tr>
<tr>
<td>../ifdl/IO#unknownOutputDevice</td>
<td>Unknown output device</td>
</tr>
<tr>
<td>../ifdl/IO#unknownBiometricSubtype</td>
<td>Unknown biometric sub-type</td>
</tr>
<tr>
<td></td>
<td>(IFD_UNKNOWN_BIOMETRIC_SUBTYPE)</td>
</tr>
</tbody>
</table>

### 4.2.5.3 Terminal

The Terminal category groups the errors which occur due to the status or properties of the terminal.

The following table contains the error codes and the corresponding description.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../ifdl/terminal#unknownIFD</td>
<td>The card terminal does not exist</td>
</tr>
<tr>
<td></td>
<td>The addressed card terminal (IFDName) is unknown.</td>
</tr>
<tr>
<td></td>
<td>(IFD UNKNOWN_IFD)</td>
</tr>
<tr>
<td>../ifdl/terminal#noCard</td>
<td>No eCard available</td>
</tr>
<tr>
<td></td>
<td>The request was not successful, because there is</td>
</tr>
</tbody>
</table>
4.3 Additional URI definitions

This section contains other URI definitions.

4.3.1 Protocols

In addition to the protocols stipulated by ISO/IEC, the following protocols are also defined in [TR-03112-7]:

http://www.bsi.bund.de/ecard/api/1.1/protocols/GetCertificate/#SimpleEnrollmentProtocol
http://www.bsi.bund.de/ecard/api/1.1/protocols/FrameworkUpdate/#BasicUpdateProtocol

4.3.2 Properties

In addition to the URI defined for Properties in [DSS] and [AdES], the following are also used in SignRequest from [CEN15480-3]:

http://www.bsi.bund.de/ecard/api/1.1/properties/previousTimeStampHash

4.3.3 CardInfo

In the GetCardInfoOrACD function from [TR-03112-5], the actions to be executed are specified with the following URIs:

http://www.bsi.bund.de/ecard/api/1.1/cardinfo/action#getSpecifiedFile
http://www.bsi.bund.de/ecard/api/1.1/cardinfo/action#getRelatedFiles
http://www.bsi.bund.de/ecard/api/1.1/cardinfo/action#getOtherFiles

4.4 Mandatory use of Profile-attribute in responses

As specified in Section 2.10 of [DSS] any request MAY contain a Profile-attribute. All responses defined in the present specification MUST include a Profile-attribute equal to http://www.bsi.bund.de/ecard/api/1.1.
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>ATR</td>
<td>Answer to Reset</td>
</tr>
<tr>
<td>ATS</td>
<td>Answer to Select</td>
</tr>
<tr>
<td>CAMS</td>
<td>Card Application Management System</td>
</tr>
<tr>
<td>DID</td>
<td>Differential-Identity</td>
</tr>
<tr>
<td>DSI</td>
<td>Data Structure for Interoperability</td>
</tr>
<tr>
<td>ICC</td>
<td>Integrated Circuit Card</td>
</tr>
<tr>
<td>ICCSN</td>
<td>ICC Serial Number</td>
</tr>
<tr>
<td>IFD</td>
<td>Interface Device</td>
</tr>
<tr>
<td>OAEP</td>
<td>Optimal Asymmetric Encryption Padding</td>
</tr>
<tr>
<td>OCSP</td>
<td>Online Certificate Status Protocol</td>
</tr>
<tr>
<td>PAN</td>
<td>Primary Account Number</td>
</tr>
<tr>
<td>PCD</td>
<td>Proximity Coupling Device</td>
</tr>
<tr>
<td>PIN</td>
<td>Personal Identification Number</td>
</tr>
<tr>
<td>PUK</td>
<td>Personal Unblocking Key</td>
</tr>
<tr>
<td>SAL</td>
<td>Service Access Layer</td>
</tr>
<tr>
<td>SICCT</td>
<td>Secure Interoperable Chipcard Terminal</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TSL</td>
<td>Trusted Service List</td>
</tr>
<tr>
<td>TSP</td>
<td>Time Stamping Protocol</td>
</tr>
<tr>
<td>WSDL</td>
<td>Webservice Description Language</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
</tbody>
</table>
6 Index of XML data types

The following list of XML data types provides a reference to the schema definition file (cf. Section 2.3) in which the respective data type (complexType and simpleType) is defined.

- APIAccessControlListType → (complex) Management.xsd
- APIAccessControlRuleType → (complex) Management.xsd
- APIAccessEntryPointName → (simple) ISO24727-3.xsd
- APIAccessRuleType → (complex) Management.xsd
- APIAuthenticationStateType → (complex) Management.xsd
- APESecurityConditionType → (complex) Management.xsd
- ATRInterfaceBytesType → (complex) CardInfo.xsd
- ATRType → (complex) CardInfo.xsd
- ATSType → (complex) CardInfo.xsd
- ATSInterfaceBytesType → (complex) CardInfo.xsd
- AccessControlListType → (complex) ISO24727-3.xsd
- AccessRuleType → (complex) ISO24727-3.xsd
- ActionNameType → (complex) ISO24727-3.xsd
- ActionType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
- ActionType → (complex) saml-schema-assertion-2.0.xsd
- ActionType → (simple) ISOIFD.xsd
- AddCertificateOptionsType → (complex) Management.xsd
- AdditionalInformationType → (complex) draft_ts102231v020101xsd.xsd
- AdditionalInformationType → (complex) gematik-TSL-02231v2.xsd
- AdditionalInformationType → (complex) ts_102231v030102_xsd.xsd
- AdditionalServiceInformationType → (complex) ts_102231v030102_xsd.xsd
- AdditionalTimeInfoType → (complex) oasis-dss-core-schema-v1.0-os.xsd
- AddressType → (complex) draft_ts102231v020101xsd.xsd
- AddressType → (complex) gematik-TSL-02231v2.xsd
- AddressType → (complex) ts_102231v030102_xsd.xsd
- AdviceType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
- AdviceType → (complex) saml-schema-assertion-2.0.xsd
- AgreementMethodType → (complex) xenc-schema.xsd
- AlgorithmIdentifierType → (complex) ISO24727-Protocols.xsd
- AlgorithmInfoType → (complex) ISO24727-Protocols.xsd
- AlgorithmValidityType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
- AltMVDMessagesType → (complex) ISOIFD.xsd
- AltVUMessagesType → (complex) ISOIFD.xsd
- AnyType → (complex) XAdES-1-3-2.xsd
- AnyType → (complex) draft_ts102231v020101xsd.xsd
- AnyType → (complex) gematik-TSL-02231v2.xsd
- AnyType → (complex) oasis-dss-core-schema-v1.0-os.xsd
- AnyType → (complex) ts_102231v030102_xsd.xsd
- ApplicationCapabilitiesType → (complex) CardInfo.xsd
- ApplicationDataRefType → (complex) CardInfo.xsd
- ApplicationIdentifierType → (simple) ISO24727-3.xsd
- ArchiveTime StampValidityType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
- AssertionType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AssertionType → (complex) saml-schema-assertion-2.0.xsd
AttachmentReferenceType → (complex) oasis-dss-core-schema-v1.0-os.xsd
AttrCertIDType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
Attribute → (complex) xmlers-schema-draft-v0.3.xsd
AttributeCertificateContentType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
AttributeCertificateValidityType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
AttributeDesignatorType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AttributeStatementType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AttributeType → (complex) saml-schema-assertion-2.0.xsd
AttributeValue → (complex) xmlers-schema-draft-v0.3.xsd
Attributes → (complex) xmlers-schema-draft-v0.3.xsd
AudienceRestrictionConditionType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AudienceRestrictionType → (complex) saml-schema-assertion-2.0.xsd
AuthenticationStatementType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AuthnContextType → (complex) saml-schema-assertion-2.0.xsd
AuthnStatementType → (complex) saml-schema-assertion-2.0.xsd
AuthorityBindingType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AuthorizationDecisionStatementType → (complex) oasis-sstc-saml-schema-protocol-1.xsd
AuthorizationServiceActionName → (simple) ISO24727-3.xsd
AuthzDecisionStatementType → (complex) saml-schema-assertion-2.0.xsd
BaseIDAbstractType → (complex) saml-schema-assertion-2.0.xsd
BaseIDAbstractType → (complex) saml-schema-assertion-2.0.xsd
BasePointParamsType → (complex) ecda.xsd
BasicRequirementsType → (simple) CardInfo.xsd
BioSensorCapabilityType → (complex) ISOIFD.xsd
BiometricInputType → (complex) ISOIFD.xsd
BitReqType → (complex) CardInfo.xsd
BitString → (simple) ISOIFD.xsd
ByteMaskType → (complex) CardInfo.xsd
ByteType → (simple) ISO24727-Protocols.xsd
CAAuthenticationTokenType → (complex) ISO24727-Protocols.xsd
CADIDCreateDataType → (complex) ISO24727-Protocols.xsd
CADIDUpdateDataType → (complex) ISO24727-Protocols.xsd
CAInputType → (complex) ISO24727-Protocols.xsd
CAKeyInfoType → (complex) ISO24727-Protocols.xsd
CMarkerType → (complex) ISO24727-Protocols.xsd
CRLContentType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
CRLIdentifierType → (complex) XAdES-1-3-2.xsd
CRLRefType → (complex) XAdES-1-3-2.xsd
CRLRefsType → (complex) XAdES-1-3-2.xsd
CRLValidityType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
CRLValuesType → (complex) XAdES-1-3-2.xsd
CanonicalizationMethodType → (complex) xmlsig-core-schema.xsd
CanonicalizationMethodType → (complex) xmlers-schema-draft-v0.3.xsd
CapabilityInfoType → (complex) CardInfoRepository.xsd
CardApplicationPathType → (complex) ISO24727-3.xsd
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>CardApplicationServiceActionName</td>
<td>(simple)</td>
<td>ISO24727-3.xsd</td>
</tr>
<tr>
<td>CardApplicationServiceDescriptionType</td>
<td>(complex)</td>
<td>ISO24727-3.xsd</td>
</tr>
<tr>
<td>CardApplicationServiceLoadPackageType</td>
<td>(simple)</td>
<td>ISO24727-3.xsd</td>
</tr>
<tr>
<td>CardApplicationServiceType</td>
<td>(complex)</td>
<td>CardInfo.xsd</td>
</tr>
<tr>
<td>CardApplicationType</td>
<td>(complex)</td>
<td>CardInfo.xsd</td>
</tr>
<tr>
<td>CardCallSequenceType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>CardCallType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>CardCapabilitiesType</td>
<td>(complex)</td>
<td>CardInfo.xsd</td>
</tr>
<tr>
<td>CardIdentificationType</td>
<td>(complex)</td>
<td>CardInfo.xsd</td>
</tr>
<tr>
<td>CardInfoType</td>
<td>(complex)</td>
<td>CardInfo.xsd</td>
</tr>
<tr>
<td>CertIDListType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CertIDType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CertificateContentType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>CertificatePathValidityType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>CertificatePathValidityVerificationDetailType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>CertificateRefType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>CertificateStatusType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>CertificateValidityType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>CertificateValuesType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CertificateValuesType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>CertifiedRolesListType</td>
<td>(complex)</td>
<td>oasis-dssx-1.0-profiles-verification-report-cs1.xsd</td>
</tr>
<tr>
<td>ChannelHandleType</td>
<td>(complex)</td>
<td>ISOCommon.xsd</td>
</tr>
<tr>
<td>CharTwoFieldElemType</td>
<td>(complex)</td>
<td>ecdsa.xsd</td>
</tr>
<tr>
<td>CharTwoFieldParamsType</td>
<td>(complex)</td>
<td>ecdsa.xsd</td>
</tr>
<tr>
<td>CipherDataType</td>
<td>(complex)</td>
<td>xenc-schema.xsd</td>
</tr>
<tr>
<td>CipherReferenceType</td>
<td>(complex)</td>
<td>xenc-schema.xsd</td>
</tr>
<tr>
<td>ClaimedRolesListType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CommandSpecificLengthInfoType</td>
<td>(complex)</td>
<td>CardInfo.xsd</td>
</tr>
<tr>
<td>CommitmentTypeIndicationType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CommitmentTypeQualifiersListType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CompleteCertificateRefsType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CompleteRevocationRefsType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>ConclusionType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>ConditionAbstractType</td>
<td>(complex)</td>
<td>oasis-sstc-saml-schema-protocol-1.xsd</td>
</tr>
<tr>
<td>ConditionAbstractType</td>
<td>(complex)</td>
<td>saml-schema-assertion-2.0.xsd</td>
</tr>
<tr>
<td>ConditionsType</td>
<td>(complex)</td>
<td>oasis-sstc-saml-schema-protocol-1.xsd</td>
</tr>
<tr>
<td>ConditionsType</td>
<td>(complex)</td>
<td>saml-schema-assertion-2.0.xsd</td>
</tr>
<tr>
<td>ConnectionHandleType</td>
<td>(complex)</td>
<td>ISO24727-3.xsd</td>
</tr>
<tr>
<td>ConnectionServiceActionName</td>
<td>(simple)</td>
<td>ISO24727-3.xsd</td>
</tr>
<tr>
<td>ContextHandleType</td>
<td>(simple)</td>
<td>ISOCommon.xsd</td>
</tr>
<tr>
<td>CounterSignatureType</td>
<td>(complex)</td>
<td>XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>CryptoBinary</td>
<td>(simple)</td>
<td>xmldsig-core-schema.xsd</td>
</tr>
<tr>
<td>CryptoDIDUpdateDataType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>CryptoKeyInfoType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>CryptoMarkerType</td>
<td>(complex)</td>
<td>ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>CryptographicServiceActionName</td>
<td>(simple)</td>
<td>ISO24727-3.xsd</td>
</tr>
<tr>
<td>Type</td>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>CurveParamsType</td>
<td>(complex) ecdsa.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDAbstractMarkerType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDAuthenticationDataType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDAuthenticationStateType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDInfoType</td>
<td>(complex) CardInfo.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDMarkerType</td>
<td>(complex) ISO24727-Protocols.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDNameListType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDNameType</td>
<td>(simple) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDQualifierType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDScopeType</td>
<td>(simple) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDStructureType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DIDUpdateDataType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DSAKeyValue</td>
<td>(complex) xmlsig-core-schema.xsd</td>
<td></td>
</tr>
<tr>
<td>DSNNameListType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DSNNameType</td>
<td>(simple) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DSIType</td>
<td>(complex) CardInfo.xsd</td>
<td></td>
</tr>
<tr>
<td>DataMaskType</td>
<td>(complex) ISO24727-Protocols.xsd</td>
<td></td>
</tr>
<tr>
<td>DataObjectFormatType</td>
<td>(complex) XAdES-1-3-2.xsd</td>
<td></td>
</tr>
<tr>
<td>DataRefType</td>
<td>(complex) ISO24727-Protocols.xsd</td>
<td></td>
</tr>
<tr>
<td>DataSetInfoType</td>
<td>(complex) CardInfo.xsd</td>
<td></td>
</tr>
<tr>
<td>DataSetNameListType</td>
<td>(complex) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DataSetNameType</td>
<td>(simple) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DecisionType</td>
<td>(simple) oasis-sstc-saml-schema-protocol-1.xsd</td>
<td></td>
</tr>
<tr>
<td>DecisionType</td>
<td>(simple) saml-schema-assertion-2.0.xsd</td>
<td></td>
</tr>
<tr>
<td>DefaultParametersType</td>
<td>(complex) Management.xsd</td>
<td></td>
</tr>
<tr>
<td>DestinationSelectorType</td>
<td>(complex) oasis-dssx-1.0-profiles-encryption-0.5.xsd</td>
<td></td>
</tr>
<tr>
<td>DetailType</td>
<td>(complex) oasis-dss-core-schema-v1.0-os.xsd</td>
<td></td>
</tr>
<tr>
<td>DetailedSignatureReportType</td>
<td>(complex) oasis-dssx-1.0-profiles-verification-report-csl.xsd</td>
<td></td>
</tr>
<tr>
<td>DifferentialIdentityServiceActionName</td>
<td>(simple) ISO24727-3.xsd</td>
<td></td>
</tr>
<tr>
<td>DifferentialIdentityType</td>
<td>(complex) CardInfo.xsd</td>
<td></td>
</tr>
<tr>
<td>DigestAlgAndValueType</td>
<td>(complex) XAdES-1-3-2.xsd</td>
<td></td>
</tr>
<tr>
<td>DigestMethodType</td>
<td>(complex) xmlsig-core-schema.xsd</td>
<td></td>
</tr>
<tr>
<td>DigestMethodType</td>
<td>(complex) xmlers-schema-draft-v0.3.xsd</td>
<td></td>
</tr>
<tr>
<td>DigestValueType</td>
<td>(simple) xmlsig-core-schema.xsd</td>
<td></td>
</tr>
<tr>
<td>DigitalIdentityListType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
<td></td>
</tr>
<tr>
<td>DigitalIdentityListType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
<td></td>
</tr>
<tr>
<td>DigitalIdentityListType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
<td></td>
</tr>
<tr>
<td>DigitalIdentityType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
<td></td>
</tr>
<tr>
<td>DigitalIdentityType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
<td></td>
</tr>
<tr>
<td>DigitalIdentityType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
<td></td>
</tr>
<tr>
<td>DisplayCapabilityType</td>
<td>(complex) ISO1FD.xsd</td>
<td></td>
</tr>
<tr>
<td>DoNotCacheConditionType</td>
<td>(complex) oasis-sstc-saml-schema-protocol-1.xsd</td>
<td></td>
</tr>
<tr>
<td>DocFormatType</td>
<td>(complex) oasis-dss-profiles-AdES-schema-v1.0-os.xsd</td>
<td></td>
</tr>
<tr>
<td>DocReferenceType</td>
<td>(complex) oasis-dss-profiles-AdES-schema-v1.0-os.xsd</td>
<td></td>
</tr>
<tr>
<td>DocReferencesType</td>
<td>(complex) oasis-dss-profiles-AdES-schema-v1.0-os.xsd</td>
<td></td>
</tr>
<tr>
<td>DocsFormatType</td>
<td>(complex) oasis-dss-profiles-AdES-schema-v1.0-os.xsd</td>
<td></td>
</tr>
<tr>
<td>DocumentBaseType</td>
<td>(complex) oasis-dss-core-schema-v1.0-os.xsd</td>
<td></td>
</tr>
<tr>
<td>DocumentType</td>
<td>(complex) oasis-dss-core-schema-v1.0-os.xsd</td>
<td></td>
</tr>
</tbody>
</table>
MultiLangStringType  →  (complex) ts_102231v030102_xsd.xsd
MutualAuthDIDAuthExternalAuthType  →  (complex) ISO24727-Protocols.xsd
MutualAuthDIDAuthInternalAuthType  →  (complex) ISO24727-Protocols.xsd
MutualAuthDIDAuthMutualAuthType  →  (complex) ISO24727-Protocols.xsd
MutualAuthDIDUpdateDataType  →  (complex) ISO24727-Protocols.xsd
NULL  →  (complex) ISO24727-3.xsd
NameIDType  →  (complex) saml-schema-assertion-2.0.xsd
NameIdentifierType  →  (complex) oasis-sstc-saml-schema-protocol-1.xsd
NameType  →  (simple) ISO24727-3.xsd
NamedDataServiceActionName  →  (complex) draft_ts102231v020101xsd.xsd
NextUpdateType  →  (complex) gematik-TSL-02231v2.xsd
NextUpdateType  →  (complex) ts_102231v030102_xsd.xsd
NonEmptyMultiLangURILListType  →  (complex) draft_ts102231v020101xsd.xsd
NonEmptyMultiLangURILListType  →  (complex) gematik-TSL-02231v2.xsd
NonEmptyMultiLangURILListType  →  (complex) ts_102231v030102_xsd.xsd
NonEmptyMultiLangURIType  →  (complex) draft_ts102231v020101xsd.xsd
NonEmptyMultiLangURIType  →  (complex) gematik-TSL-02231v2.xsd
NonEmptyMultiLangURIType  →  (complex) ts_102231v030102_xsd.xsd
NonEmptyNormalizedString  →  (simple) draft_ts102231v020101xsd.xsd
NonEmptyNormalizedString  →  (simple) gematik-TSL-02231v2.xsd
NonEmptyNormalizedString  →  (simple) ts_102231v030102_xsd.xsd
NonEmptyString  →  (simple) draft_ts102231v020101xsd.xsd
NonEmptyString  →  (simple) gematik-TSL-02231v2.xsd
NonEmptyString  →  (simple) ts_102231v030102_xsd.xsd
NonEmptyURILListType  →  (complex) draft_ts102231v020101xsd.xsd
NonEmptyURILListType  →  (complex) gematik-TSL-02231v2.xsd
NonEmptyURILListType  →  (complex) ts_102231v030102_xsd.xsd
NonEmptyURIType  →  (simple) draft_ts102231v020101xsd.xsd
NonEmptyURIType  →  (simple) gematik-TSL-02231v2.xsd
NonEmptyURIType  →  (simple) ts_102231v030102_xsd.xsd
NoticeReferenceType  →  (complex) XAdES-1-3-2.xsd
OCSPContentType  →  (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
OCSPIdentifierType  →  (complex) XAdES-1-3-2.xsd
OCSPRefType  →  (complex) XAdES-1-3-2.xsd
OCSPRefsType  →  (complex) XAdES-1-3-2.xsd
OCSPValidityType  →  (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
OCSPValuesType  →  (complex) XAdES-1-3-2.xsd
ObjectIdentifier  →  (simple) xmlers-schema-draft-v0.3.xsd
ObjectIdentifierType  →  (complex) XAdES-1-3-2.xsd
ObjectType  →  (complex) xmlers-schema-draft-v0.3.xsd
OddCharExtensionFieldElemType  →  (complex) ecdsa.xsd
OddCharExtensionFieldParamsType  →  (complex) ecdsa.xsd
OneTimeUseType  →  (complex) saml-schema-assertion-2.0.xsd
OpenType  →  (complex) xmlers-schema-draft-v0.3.xsd
OtherCertStatusRefsType  →  (complex) XAdES-1-3-2.xsd
OtherCertStatusValuesType  →  (complex) XAdES-1-3-2.xsd
OtherTSLPointerType
OtherTSLPointerType
OtherTSLPointerType
OtherTSLPointerType
OtherTSLPointerType
OtherTSLPointerType
OtherTimeStampType
OutputInfoType
PACEDIDAuthenticateInputType
PACEDIDAuthenticateOutputType
PACEDIDUpdateDataType
PACEMarkerType
PGPDataType
PSSParameterType
PadCharType
PasswordAttributesType
PasswordFlagsType
PasswordTypeType
PathSecurityType
PathType
PinCompareDIDAuthenticateInputType
PinCompareDIDAuthenticateOutputType
PinCompareDIDUpdateDataType
PinCompareMarkerType
PinInputType
PnBFieldParamsType
PolicyOrLegalnoticeType
PolicyOrLegalnoticeType
PolicyOrLegalnoticeType
PolicySignaturePairType
PolicySignaturePairsType
PostalAddressListType
PostalAddressListType
PostalAddressListType
PostalAddressType
PostalAddressType
PostalAddressType
PrimeFieldElemType
PrimeFieldParamsType
ProfilingType
PropertiesType
PropertiesType
ProtocolDataType
ProxyRestrictionType
QualifierType
QualifyingPropertiesReferenceType
QualifyingPropertiesType
RIDIDAuthInputType
<table>
<thead>
<tr>
<th>Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TADIDUpdateDataType</td>
<td>(complex) ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>TAMarkerType</td>
<td>(complex) ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>TLS_FSK_ParametersType</td>
<td>(complex) ISO24727-Protocols.xsd</td>
</tr>
<tr>
<td>TSLSchemeInformationType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
</tr>
<tr>
<td>TSLSchemeInformationType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TSLTagType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TSLTagType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TSLTagType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TSLType</td>
<td>(complex) Management.xsd</td>
</tr>
<tr>
<td>TSPInformationType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
</tr>
<tr>
<td>TSPInformationType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TSPInformationType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TSPServiceInformationType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
</tr>
<tr>
<td>TSPServiceInformationType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TSPServiceType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TSPServiceType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TSPServiceType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
</tr>
<tr>
<td>TSPType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TSPType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TSPType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TargetNameType</td>
<td>(complex) ISO24727-3.xsd</td>
</tr>
<tr>
<td>TimeSignatureInstructionType</td>
<td>(complex) oasis-dss-core-schema-v1.0-os.xsd</td>
</tr>
<tr>
<td>TimeStampValidityType</td>
<td>(complex) oasis-dssx-1.0-profiles-verification-report-csl.xsd</td>
</tr>
<tr>
<td>TnBFieldParamsType</td>
<td>(complex) ecdsa.xsd</td>
</tr>
<tr>
<td>TransformType</td>
<td>(complex) xmldsig-core-schema.xsd</td>
</tr>
<tr>
<td>TransformsType</td>
<td>(complex) xenc-schema.xsd</td>
</tr>
<tr>
<td>TransformsType</td>
<td>(complex) xmldsig-core-schema.xsd</td>
</tr>
<tr>
<td>TruType</td>
<td>(simple) ISO24727-3.xsd</td>
</tr>
<tr>
<td>TrustServiceProviderListType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
</tr>
<tr>
<td>TrustServiceProviderListType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TrustServiceProviderListType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TrustStatusListType</td>
<td>(complex) draft_ts102231v020101xsd.xsd</td>
</tr>
<tr>
<td>TrustStatusListType</td>
<td>(complex) gematik-TSL-02231v2.xsd</td>
</tr>
<tr>
<td>TrustStatusListType</td>
<td>(complex) ts_102231v030102_xsd.xsd</td>
</tr>
<tr>
<td>TrustedViewerIdType</td>
<td>(simple) eCard.xsd</td>
</tr>
<tr>
<td>TrustedViewerInfoType</td>
<td>(complex) eCard.xsd</td>
</tr>
<tr>
<td>TstContentType</td>
<td>(complex) oasis-dssx-1.0-profiles-verification-report-csl.xsd</td>
</tr>
<tr>
<td>UnregisterIFMModeType</td>
<td>(simple) Management.xsd</td>
</tr>
<tr>
<td>UnsignedDataObjectPropertiesType</td>
<td>(complex) XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>UnsignedPropertiesType</td>
<td>(complex) XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>UnsignedPropertiesType</td>
<td>(complex) XAdES-1-3-2.xsd</td>
</tr>
<tr>
<td>UnsignedSignaturePropertiesType</td>
<td>(complex) XAdES-1-3-2.xsd</td>
</tr>
</tbody>
</table>

48 Bundesamt für Sicherheit in der Informationstechnik
UnsignedSignaturePropertiesType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
UpdateActionType → (simple) eCard-Protocols.xsd
UpdateCounterType → (complex) ISO24727-Protocols.xsd
UpdateModuleInfoType → (complex) eCard-Protocols.xsd
UpdatePriorityType → (simple) eCard-Protocols.xsd
UpdateSignatureInstructionType → (complex) oasis-dss-core-schema-v1.0-os.xsd
UpdatedSignatureType → (complex) oasis-dss-core-schema-v1.0-os.xsd
UseVerificationTimeType → (complex) oasis-dss-core-schema-v1.0-os.xsd
ValidityPeriodType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
VariantIndicatorType → (simple) CardInfo.xsd
VerificationReportType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
VerificationResultType → (complex) oasis-dssx-1.0-profiles-verification-report-cs1.xsd
VerificationTimeInfoType → (complex) oasis-dss-core-schema-v1.0-os.xsd
VerifiedUnderSignaturePolicyType → (complex) oasis-dssx-1.0-profiles-sigpolicy-schema-cd01.xsd
VerifyManifestResultsType → (complex) oasis-dss-core-schema-v1.0-os.xsd
VerifyUnderSignaturePolicyType → (complex) oasis-dssx-1.0-profiles-sigpolicy-schema-cd01.xsd
ViewerConfigurationType → (complex) Management.xsd
WriteBehaviourType → (simple) CardInfo.xsd
X509DataType → (complex) xmldsig-core-schema.xsd
X509IssuerSerialType → (complex) xmldsig-core-schema.xsd
XAdESTimeStampType → (complex) XAdES-1-3-2.xsd
References

[TR-03110] BSI: TR-03110: Advanced Security Mechanisms for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE) and Restricted Identification (RI)


[EN14890-1] CEN: EN14890-1: Application Interface for smart cards used as secure signature creation devices - Part 1: Basic services


[eGK-1] gematik: Specification of the electronic health insurance card - Part 1: Commands, algorithms and functions of the operating system platform, Version 2.2.2

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


[PAOSv2.0] Liberty Alliance Project: Liberty Reverse HTTP Binding for SOAP Specification, Version v2.0

[AdES] OASIS: Advanced Electronic Signature Profiles of the OASIS Digital Signature Service Version 1.0

[DSS] OASIS: Digital Signature Service Core Protocols, Elements, and Bindings

[SOAPv1.1] W3C: W3C Note: Simple Object Access Protocol (SOAP) 1.1

[WSDL] W3C: W3C Recommendation: Web Services Description Language (WSDL) Version 1.1
