INTERNATIONAL STANDARD

ISO/IEC 24727-3

First edition
2008-12-01

AMENDMENT 1
2014-04-01

Identification cards — Integrated circuit card programming interfaces —

Part 3: Application interface

AMENDMENT 1

Cartes d'identification — Interfaces programmables de cartes à puce —

Partie 3: Interface d'application

AMENDEMENT 1

Reference number
ISO/IEC 24727-3:2008/Amd.1:2014(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 24727-3:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

ISO/IEC 24727-2 standardizes the use of ISO/IEC 7816-15 data structures as "discovery information" that is communicated throughout the ISO/IEC 24727 stack. This Amendment enhances the definition of the use of ISO/IEC 7816-15 to fully link the entities defined at the Service Access Layer (API) (e.g. Differential Identity, Authentication Protocol) with typical "on-card" entities such as keys, files and Access Control Rules. Examples are provided.

XML encodings have become more and more used in the field of IAS [Identity, Authentication and (digital) Signature] Identity Management and general networking communication. To enhance interoperability with existing networking systems and federated identification and authorization systems (e.g. SAML, OpenID, etc.) standardization of an XML representation of the API and data structures of ISO/IEC 24727-3 is essential.

This Amendment extends the scope of ISO/IEC 24727-3 in the following ways:

1. Make explicit (normative and informative elements, including examples) of the use of the ISO/IEC 7816-15-based Registry. XML representation of the ISO/IEC 24727-3 API including appropriate web service bindings specified as WSDL-structure.
2. Reaffirm that ASN.1 is the central definition of the API and data structures. All other bindings and representations are derived from ASN.1
3. ASN.1 and XML representations for ISO/IEC 24727-3 will reside in this part, which may necessitate movement of text/annexes from other parts of ISO/IEC 24727 (i.e. ISO/IEC 24727-2 and ISO/IEC 24727-4 and ISO/IEC 24727-5).
4. As a result of Amendments under development for other parts of ISO/IEC 24727, portions of this standard may be deleted and referenced.
5. Add to this standard the ISO/IEC 7816-13 application management and life cycle concepts.
6. Add a discovery mechanism to the API to indicate messaging is either ASN.1 or XML.
7. Enhance a Registry facility through which the SAL can record its use of the GCI and through which the GCI mechanisms can be conveyed to the SAL.
8. Consider enhancements to the API to resolve technical deficiencies.
9. Remove ambiguities by elaborating and re-specifying concepts that may not be clear in the current standard.

10. Incorporate concepts that are captured in other parts of ISO/IEC 24727 but are more relevant for ISO/IEC 24727-3.
11. Include C and Java bindings in a Normative Annex (C) and an Informative Annex(Java).
12. Pursue additional mechanisms for discovery and (card and application) capability description based e.g. on XML representations as part the development of a more comprehensive Registry. *This XML is restricted to a set of instructions that enable card recognition for legacy cards.*

Identification cards — Integrated circuit card programming interfaces —

Part 3: Application interface

AMENDMENT 1

Add the following new annexes after Annex C:

Annex D
(normative)

Web Services Interface Description

D.1 Contents of Annex D

- **Clause D.2** contains information for the connection establishment for web service based SAL-communication.
- **Clause D.3** contains an XML-based CardInfo-Structure, which facilitates the support of legacy cards.
- **Clause D.4** contains the XML-specification for the XML-based Service Access Layer Interface

D.2 Connection handling for web service based communication

This clause describes the connection handling for web service based communications in an ISO/IEC 24727-based environment.

D 2.1 General security requirements

The security requirements of ISO/IEC 24727 dictate that the TLS protocol in accordance with RFC 4346 shall be used.

Moreover, public server services shall have access to corresponding X.509 certificates.

When it comes to the security of the communication between the different modules realizing the ECC-3 stack consisting of Service Access Layer (SAL) and IFD Layer), however, X.509 certificates shall be used, whereby the associated private keys shall be adequately protected. Alternatively, anonymous TLS cipher suites, such as `TLS_DH_anon` from RFC 4346 or `TLS_ECDH_anon` from RFC 4492, shall be used, although appropriate security measures need to be taken in the operational environment in order to avert man-in-the-middle attacks while the connection is being established.

In both cases there shall be an exclusive binding of the communication context at application level to the TLS channel which has been established in this process. This communication context is established on connection to the IFD-Layer via the function `EstablishContext` and represented by the `ContextHandle`. When connecting to the SAL, this communication context corresponds to a connection to the card application established by means of `CardApplicationConnect`, which is represented by a `ConnectionHandle`.

As such, one single TLS channel shall be sufficient to establish communication between a SAL and the IFD layer — irrespective of the number of card terminals and connected cards — whereas a separate TLS channel shall be required for every connection to a card application for communication to take place between the identity layer or the application layer and the SAL.

D.2.2 Connections for SOAP binding

When using the SOAP binding, the connection shall be established simply by setting up a TLS-protected channel between the user of the web service (service consumer) and the provider of the web service (service provider) via which web service messages can henceforth be exchanged. In this case the service consumer and service provider take the roles of TLS/http client and TLS/http server, respectively.

D.2.3 Connections for PAOS binding

When using the PAOS binding, however, a more complex process shall be required to establish the connection as, in this case, the TLS/http server acts as the user of the web service (service consumer) with eService because the TLS/http client acts as the provider of the web service (service provider) and shall initiate the connection.

Moreover, in this case there are typically two different TLS channels, and appropriate cryptographic mechanisms shall be used to safeguard their logical relationship while the connection is established.

An example general connection sequence is illustrated in Figure D.1.

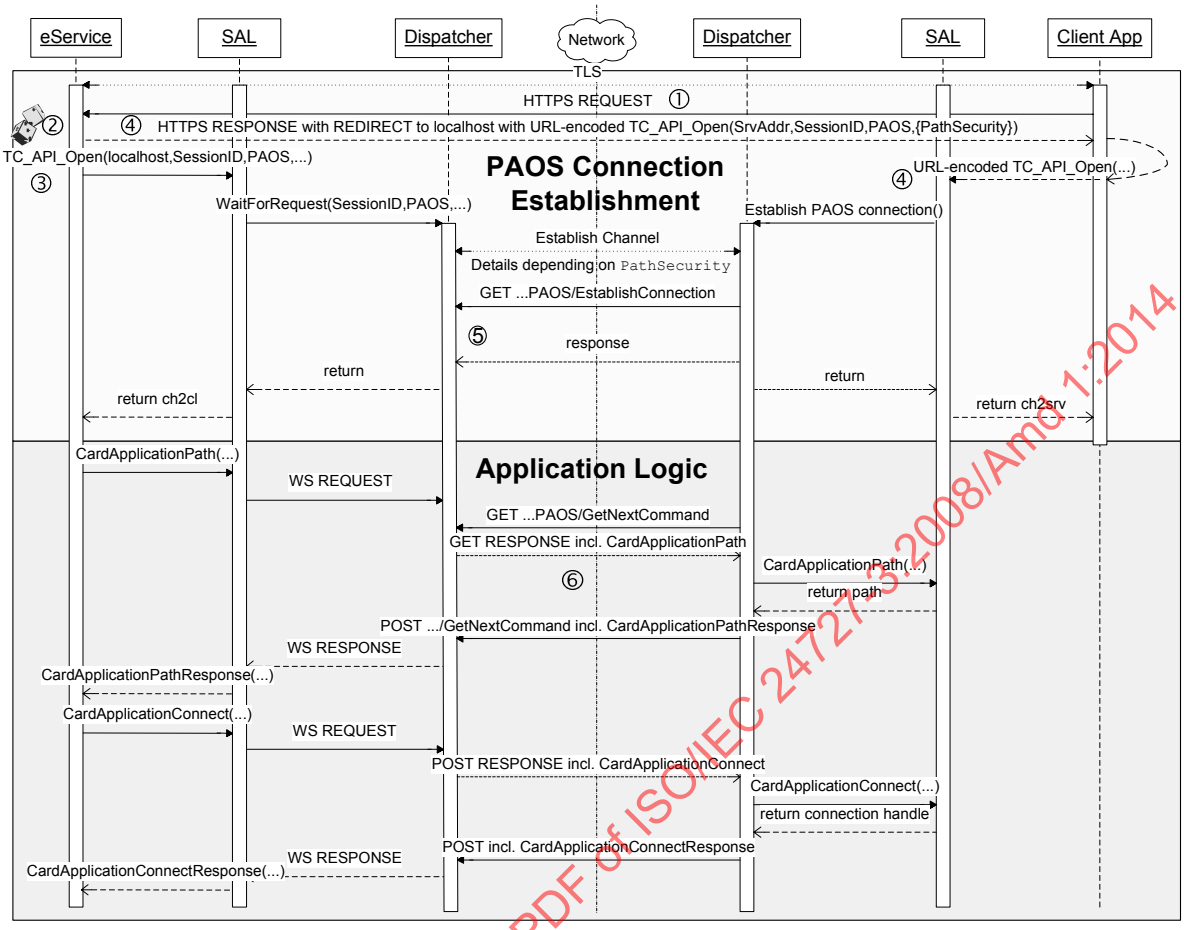


Figure D.1 — Example of a general connection process with PAOS binding

D.3 XML-based CardInfo-Structure for support of legacy cards

D.3.1 Introduction

In order to use the generic Service Access Layer (SAL) interface defined in this standard with legacy cards, it shall be necessary to provide a certain set of information, which allows to map the generic calls at the SAL to card-specific APDUs.

The XML-schema introduced in this clause defines a structure that shall both be used for the specification of card profiles *and* for the mapping of generic calls at the Service Access Interface to card-specific APDUs of legacy cards.

The rest of this annex is structured as follows: Clause D.3.2 provides an overview of the defined structure and explains the top-level element `<CardInfo>`. The following clauses D.3.3 through D.3.7 describe the various child-elements of `<CardInfo>`, Clause E.7 contains the complete XML-schema-definition

D.3.2 Overview

The `CardInfo` structure shall be used for the specification of card profiles *and* for the mapping of generic requests at the Service Access Layer to card-specific APDUs in case of legacy cards, which are not equipped with appropriate ACD and CCD structures according to ISO/IEC 24727-2.

Each card profile shall be described by a `<CardInfo>` element

```
<element name="CardInfo" type="iso:CardInfoType" />
```

of type `CardInfoType` which shall be defined as follows:

```
<complexType name="CardInfoType">
  <sequence>
    <element name="CardType" type="iso:CardTypeType" />
    <element name="CardIdentification" type="iso:CardIdentificationType" />
    <element name="CardCapabilities"
             type="iso:CardCapabilitiesType" maxOccurs="1" minOccurs="0" />
    <element name="ApplicationCapabilities"
             type="iso:ApplicationCapabilitiesType" maxOccurs="1" minOccurs="0" />
    <element name="Signature"
             type="ds:SignatureType" maxOccurs="unbounded" minOccurs="0" />
    </sequence>
    <attribute name="Id" type="ID" use="optional" />
  </complexType>
```

`<CardType>` `[required]`

Contains a unique identifier for the card type and optionally further links to specification documents.

`<CardIdentification>` `[required]`

Allows to determine the type of a given card by traversing an appropriate decision tree and checking whether the characteristic features are as expected.

`<CardCapabilities>` `[optional]`

Allows to specify the capabilities of the card. If the card is fully conformant to ISO/IEC 7816 this element MAY be omitted.

`<ApplicationCapabilities>` `[optional]`

Allows to specify the card-applications on the card and shall be used to realize the mapping from SAL-calls to card-specific APDUs. If the necessary information for this mapping is available on the card in adequate CIA-information structures according to ISO/IEC 7816-15 (see Clause 7.5) this element may be omitted.

`<Signature>` `[optional]`

Shall be used to protect the integrity and authenticity of (parts of) the `CardInfo`-element.

D.3.3 CardType

The `<CardType>` element in the `CardInfoType` is of type `CardTypeType` and contains a unique identifier for the card type and optionally further links to specification documents. It is specified as follows:

```
<complexType name="CardTypeType">
  <sequence>
    <element name="ProfilingInfo" maxOccurs="1" minOccurs="0">
      <complexType>
        <sequence>
          <element name="BasisSpecification" type="anyURI" />
          <element name="ProfilingRelation" type="iso:ProfilingType" />
        </sequence>
```

```
      </complexType>
    </element>
    <element name="ObjectIdentifier" type="anyURI" />
    <element name="SpecificationBodyOrIssuer"
             type="string" maxOccurs="1" minOccurs="0" />
    <element name="CardTypeName" type="string" maxOccurs="1" minOccurs="0" />
    <element name="Version" maxOccurs="1" minOccurs="0">
      <complexType>
        <sequence>
          <element name="Major" type="string" />
          <element name="Minor" type="string" maxOccurs="1" minOccurs="0" />
          <element name="SubMinor" type="string" maxOccurs="1" minOccurs="0" />
        </sequence>
      </complexType>
    </element>
    <element name="Status" type="string" maxOccurs="1" minOccurs="0" />
    <element name="Date" type="date" maxOccurs="1" minOccurs="0" />
    <element name="CardInfoRepository" type="anyURI" maxOccurs="1" minOccurs="0"/>
          <any namespace="##any" processContents="lax" minOccurs="0"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional" />
</complexType>
```

This type defines the following elements and attributes:

`<ProfilingInfo> [optional]`

This element shall contain information about a basic specification (`<BasisSpecification>` element) which is extended, profiled or redefined (cf. `<ProfilingRelation>` element below) by the present `CardInfo` structure. Using this element it shall be possible to re-use existing `CardInfo`-structures in a modular approach.

`<ObjectIdentifier> [required]`

This element shall contain the unique identifier of the card type, which MAY be the object identifier of a profile defined in Part 4 of the present standard.

`<SpecificationBodyOrIssuer> [optional]`

This element may be used to specify the card issuer or the organization, which is responsible for the specification.

`<CardTypeName> [optional]`

This element may contain the name of the card type.

`<Version> [optional]`

This element may contain the version number of the card type.

`<Status> [optional]`

This element may contain information about the state of the present `CardInfo` file (e.g. ‘draft’).

`<Date> [optional]`

This element may contain the date of creation of the `CardInfo` file.

`<CardInfoRepository> [optional]`

This element may contain the address of a CardInfo-repository, which may provide related CardInfo-files.

Furthermore there may be some additional element, which structure is defined by some other specification.

The <ProfilingRelation> element is of type ProfilingType and describes the relation between the basic specification and the present CardInfo file.

```
<simpleType name="ProfilingType">
  <restriction base="string">
    <enumeration value="extends" />
    <enumeration value="redefines" />
  </restriction>
</simpleType>
```

The three cases have got the following meaning:

— extends – indicates that the present CardInfo file is just an extension of the basic specification. All definitions in the basic specification remain valid and the new specifications in the CardInfo file just extend them (e.g. a new card application).

— redefines – indicates that the elements of the CardInfo file overwrite the according elements of the basic specification. Elements of the basic specification not appearing in the CardInfo file remain valid.

D.3.4 CardIdentification

The <CardIdentification> element, which is part of the CardInfoType, allows to determine the type of a given card by traversing the decision tree and checking whether the characteristic features are as expected.

```
<complexType name="CardIdentificationType">
   <sequence>
      <element name="ATR" maxOccurs="unbounded" minOccurs="0"
         type="iso:ATRType" />
      <element name="ATS" type="iso:ATSType" maxOccurs="1"
         minOccurs="0" />
      <element name="CharacteristicFeature" maxOccurs="unbounded"
         minOccurs="0">
         <complexType>
            <sequence maxOccurs="unbounded" minOccurs="1">
               <element name="CardCall"
                  type="iso:CardCallType" />
            </sequence>
         </complexType>
      </element>
      <any namespace="##any" processContents="lax" minOccurs="0"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional" />
</complexType>
```

<ATR> [optional, unbounded]

For contact-based smart cards this element may contain information to the Answer To Reset[1] (ATR) of the card, allowing the realisation of a preselection of the card type. Further details are explained below.

`<ATS> [optional]`

In an analogous way this element may contain information to the Answer To Select of a contactless smart card, allowing the realisation of a preselection of the card type. Further details are explained below.

`<CharacteristicFeature> [optional, unbounded]`

This element may contain a list of card calls, which can be used to determine the card type. The elements of the list are of type `CardTypeType`. Further details are explained below.

Furthermore there may be some additional element, which structure is defined by some other specification.

The `<ATR>` element of type `ATRType` is part of the element `<CardIdentification>`.

```
<complexType name="ATRType">
  <sequence>
    <element name="TS" type="iso:ByteMaskType" maxOccurs="1" minOccurs="1" />
    <element name="T0" type="iso:ByteMaskType" maxOccurs="1" minOccurs="1" />
    <element name="InterfaceBytes">
      <complexType>
        <sequence>
          <element name="Tx1" type="iso:ATRInterfaceBytesType" />
          <element name="Tx2" type="iso:ATRInterfaceBytesType" />
          <element name="Tx3" type="iso:ATRInterfaceBytesType" />
          <element name="Tx4" type="iso:ATRInterfaceBytesType" />
        </sequence>
      </complexType>
    </element>
    <element name="HistoricalBytes" maxOccurs="1" minOccurs="0">
      <complexType>
        <sequence maxOccurs="15" minOccurs="0">
          <element name="Ti" type="iso:ByteMaskType" />
        </sequence>
      </complexType>
    </element>
    <element name="TCK" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
  </sequence>
</complexType>
```

`<TS> [required]`

The element `<TS>` describes the first byte of the communication. This element is as many others of type `ByteMaskType` which is explained below.

`<T0> [required]`

The element `<T0>` is of type `ByteMaskType`and describes the "format character" which shall indicate the amount of historical bytes and the presence of the first interface bytes (TA1, TB1, TC1 and TD1).

`<InterfaceBytes> [required]`

[1] Because there are several protocols for contact smart cards (e.g. T=0 and T=1) which can be supported by the same card it is possible that one card has several ATRs. All of these ATR elements could be used for a preselection of the card type (using a disjunction).

This element shall contain the interface bytes which are included in the ATR. The elements <Tx*i*>, *i*∈{1,2,3,4 } are of type ATRInterfaceBytesType.. This type is explained below.

<HistoricalBytes> [optional]

This element contains the historical bytes as a sequence of at most 15 bytes. Each element <Ti> is of type ByteMaskType(see below) which also describes the significant part of the byte to identify the card type.

<TCK> [optional]

This element of type ByteMaskType may contain the check sum of all bytes of the ATR beginning with T0.

The ByteMaskType consists of a hexadecimal value and a corresponding mask which results in the significant part of the value when a logical AND is performed on value and mask.

```
<complexType name="ByteMaskType">
  <sequence>
    <element name="Value" type="hexBinary" maxOccurs="1" minOccurs="1" />
    <element name="Mask" type="hexBinary" maxOccurs="1" minOccurs="1" />
  </sequence>
</complexType>
```

If the whole byte is significant the mask 'FF' has to be used. To get the first half byte the mask has to be 'F0', for the second half byte '0F'. If only the first bit is significant the mask would be '80' and so on.

The type ATRInterfaceBytesType is used by the elements <Tx*i*> of the element <InterfaceBytes>. This type consists of four elements <Tx*i*>, *x*∈{A,B,C,D}. Each of these elements is of type ByteMaskType (see above) which also describes the significant part of the byte.

```
<complexType name="ATRInterfaceBytesType">
  <sequence>
    <element name="TAi" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
    <element name="TBi" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
    <element name="TCi" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
    <element name="TDi" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
  </sequence>
</complexType>
```

The <ATS> element of type ATSType is part of the element <CardIdentification and describes the Answer To Select of contactless smart cards.

```
<complexType name="ATSType">
  <sequence>
    <element name="TL" type="iso:ByteMaskType" maxOccurs="1" minOccurs="1" />
    <element name="T0" type="iso:ByteMaskType" maxOccurs="1" minOccurs="1" />
    <element name="InterfaceBytes" type="iso:ATSInterfaceBytesType" />
    <element name="HistoricalBytes" maxOccurs="1" minOccurs="0">
      <complexType>
        <sequence maxOccurs="15" minOccurs="0">
          <element name="Ti" type="iso:ByteMaskType"/>
        </sequence>
      </complexType>
    </element>
    <element name="CRC1" type="iso:ByteMaskType" maxOccurs="1" minOccurs="1" />
    <element name="CRC2" type="iso:ByteMaskType" />
```

```
  </sequence>
</complexType>
```

<TL> [required]

The element <TL> contains the length of the ATS including the TL byte itself but excluding CRC1 and CRC2. This element is of type ByteMaskTypewhich also describes the significant part of the byte.

<T0> [required]

The element <T0> contains the Frame Size for proximity Card Integer (FSCI) and also indicates the presence of interface bytes in the ATS. This element is of type ByteMaskType which also describes the significant part of the byte.

<InterfaceBytes> [required]

This element contains the interface bytes which could be included in the ATS. The element is of type ATSInterfaceBytesType which is explained below.

<HistoricalBytes> [optional]

This element contains the historical bytes as a sequence of at most 15 bytes. Each element <Ti> is of type ByteMaskType which also describes the significant part of the byte to identify the card type.

<CRC1>, <CRC2> [required]

These two elements of type ByteMaskType can contain the check sum of all bytes of the ATS beginning with TL.

The following type ATSInterfaceBytesType is used by the element <InterfaceBytes> of the element <ATS>. This type consists of three elements <Tx1>, $x \in \{A,B,C\}$. Each of these elements is of type ByteMaskType which also describes the significant part of the byte.

```
<complexType name="ATSInterfaceBytesType">
  <sequence>
    <element name="TA1" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
    <element name="TB1" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
    <element name="TC1" type="iso:ByteMaskType" maxOccurs="1" minOccurs="0" />
  </sequence>
</complexType>
```

A list of elements of the type CardCallType is used to describe the characteristic features of the card in the <CardIdentification> element. While the CardCallType is defined in the protocol related schema definition it is explained here to ease reading.

```
<complexType name="CardCallType">
   <sequence>
       <element name="CommandAPDU" type="hexBinary" />
       <element name="ResponseAPDU" type="iso:ResponseAPDUType"
           maxOccurs="unbounded" minOccurs="1" />
   </sequence>
</complexType>
```

<CommandAPDU> [required]

This element contains the command APDU to be sent to the card. For security reasons APDUs with CLA values '0*x*' or '1*x*', *x*∈{0,…,9, A,…,F} and INS values '20', '21', '24', '2C', and '22' SHOULD NOT be used, because these values would correspond to the commands CHANGE REFERENCE DATA, RESET RETRY COUNTER und MANAGE SECURITY ENVIRONMENT. With these commands an attacker could use the card identification to increase the retry counter of the card which could result in a denial of service attack. Such commands shall be blocked if the `<CharacteristicFeature>` element is not signed by a trustworthy authority.

`<ResponseAPDU> [required]`

This element contains the response of the smart card to the command APDU. If the response corresponds to the given value in the `CardInfo`file the actual path in the decision tree will be followed until an accepted state is reached.

The `ResponseAPDU` is of type ResponseAPDUType, which is defined as follows:

```
<complexType name="ResponseAPDUType">
   <sequence>
      <element name="Body" type="iso:DataMaskType" maxOccurs="1"
         minOccurs="0" />
      <element name="Trailer" type="hexBinary" />
   </sequence>
</complexType>
```

`<Body> [optional]`

This element MAY contain information related to the analysis of the response APDU. It is of type `DataMaskType`, which is explained below.

`<Trailer> [required]`

This element relates to the expected status of the command and is equal to '9000' in case of expected success.

The `DataMaskType` is used to describe the body of a response APDU. Because of the optional `<Tag>`-element and choice between the terminal `MatchingData`-element or the recursive `DataObject`-element, which is again of `DataMaskType` it is possible to handle arbitrarily nested TLV structures. It is defined as follows:

```
<complexType name="DataMaskType">
   <sequence>
      <element name="Tag" type="hexBinary" maxOccurs="1"
         minOccurs="0" />
      <choice>
         <element name="MatchingData" type="iso:MatchingDataType"/>
         <element name="DataObject" type="iso:DataMaskType" />
      </choice>
   </sequence>
</complexType>
```

`<Tag> [optional]`

This element MAY contain the tag where the expected value or structured data object is stored.

`<MatchingData> [choice]`

This element contains a description of the data against which the response from the card shall be matched. It is of type `MatchingDataType`, which is explained below.

`<DataObject> [choice]`

This element contains a further structured data object of type`DataMaskType`, which is returned by the smart card. This element is – similarly to the element <Body> contained in the <ResponseAPDU> element –DataMaskType, so that arbitrary deep nested TLV coded structures could be defined.

The `MatchingDataType` defines the structure of the terminal data against which the data from the card shall be matched. It is defined as follows:

```
<complexType name="MatchingDataType">
    <sequence>
        <element name="Offset" type="hexBinary"
            maxOccurs="1" minOccurs="0" />
        <element name="Length" type="hexBinary"
            maxOccurs="1" minOccurs="0" />
        <element name="MatchingValue" type="hexBinary"/>
        <element name="Mask" type="hexBinary"
            maxOccurs="1" minOccurs="0" />
    </sequence>
    <attribute name="MatchingRule" type="iso:MatchingRuleType" use="optional"
default="Equals" />
</complexType>
```

`<Offset> [optional]`

This element MAY contain an offset, which specifies a starting point at which the data from the card shall be considered for matching.

`<Length> [optional]`

This element MAY contain the length of the data considered for matching.

`<MatchingValue> [optional]`

This element contains the value, which is expected to be equal to or contained in the value returned by the smart card. If this element is missing the default value “Equals” is assumed.

`<Mask> [optional]`

This element MAY contain a mask to perform a logical AND on the data returned before it is compared to the `MatchingValue`. This way it is possible to filter out a specific part (e.g. single bits) of the data. Refer also to the definition of the`ByteMaskType`.

`<MatchingRule> [optional attribute]`

This optional attribute MAY indicate, whether the MatchingValue is expected to be equal (`Equals`) or only contained (`Contains`) in the data returned from the card. It is of type `MatchingRuleType`, which is defined as follows:

```
<simpleType name="MatchingRuleType">
    <restriction base="string">
        <enumeration value="Equals" />
        <enumeration value="Contains" />
    </restriction>
</simpleType>
```

D.3.5 CardCapabilities

The `<CardCapabilities>` element of type `CardCapabilitiesType` is an optional part of the `<CardInfo>` element and specifies the general capabilities of the card.

This information may be used to support cards which are not fully conforming to ISO/IEC 7816-4. Furthermore it is possible to specify requirements for cards and card profiles.

If a card is fully conform to ISO/IEC 7816-4 this element is not needed for mapping SAL-requests to APDUs.

```
<complexType name="CardCapabilitiesType">
   <sequence>
      <element name="Interface" maxOccurs="unbounded"
         minOccurs="0">
         <complexType>
            <complexContent>
               <extension base="iso:RequirementsType">
                  <sequence>
                     <element name="Protocol" type="anyURI" />
                  </sequence>
               </extension>
            </complexContent>
         </complexType>
      </element>
      <element name="EF.DIR" maxOccurs="1" minOccurs="0"
         type="iso:FileRefReqType" />
      <element name="EF.ATRorINFO" maxOccurs="1" minOccurs="0"
         type="iso:EFATRorINFOType" />
       <any namespace="##any" processContents="lax" minOccurs="0"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional" />
</complexType>
```

`<Interface> [optional, unbounded]`

This element MAY contain an identifier for a supported transport protocol. The element is an extension of the `RequirementsType` and therefore contains an element `<RequirementLevel>` (see below). Additionally it contains an element `<Protocol>` which holds an URI specifying the transport protocol. The following protocols MAY be used:

- urn:iso:std:iso-iec:7816:-3:tech:protocols:T-equals-0
- urn:iso:std:iso-iec:7816:-3:tech:protocols:T-equals-1
- urn:iso:std:iso-iec:10536:tech:protocols:T-equals-2
- urn:iso:std:iso-iec:14443:-2:tech:protocols:Type-A
- urn:iso:std:iso-iec:14443:-2:tech:protocols:Type-B

`<EF.DIR> [optional]`

This element MAY contain information about the EF.DIR. For details refer to the definition of`FileRefReqType`.

`<EF.ATRorINFO> [optional]`

This element MAY contain information about the EF.ATRorINFO. For details refer to the definition of`EFATRType`.

Furthermore there MAY be some additional element, which structure is defined by some other specification.

The element <`RequirementLevel`> in the type `RequirementsType` specifies, whether the given feature SHALL be supported by the platform ('PlatformMandatory'), MAY be supported by the platform ('PlatformOptional'), SHALL be personalized ('PersonalizationMandatory') or MAY be personalized ('PersonalizationOptional').

```
<complexType name="RequirementsType">
  <sequence maxOccurs="1" minOccurs="1">
    <element name="RequirementLevel"
             type="iso:BasicRequirementsType" maxOccurs="1" minOccurs="0" />
  </sequence>
</complexType>
```

The type `BasicRequirementsType` is defined as follows:

```
<simpleType name="BasicRequirementsType">
  <restriction base="string">
    <enumeration value="PlatformMandatory" />
    <enumeration value="PlatformOptional" />
    <enumeration value="PersonalizationMandatory" />
    <enumeration value="PersonalizationOptional" />
  </restriction>
</simpleType>
```

The type `FileRefReqType` extends the `RequirementsType` by an element <`Path`> which is of type `PathType`(see below):

```
<complexType name="FileRefReqType">
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="Path" type="iso:PathType" maxOccurs="1" minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The type `PathType` is based on ISO/IEC 7816-15 and used to specify the path to a file or a part of it:

```
<complexType name="PathType">
  <sequence>
    <choice>
      <element name="efIdOrPath" type="hexBinary" />
      <element name="TagRef">
        <complexType>
          <sequence>
            <element name="tag" type="hexBinary" />
            <element name="efIdOrPath"
                     type="hexBinary" maxOccurs="1" minOccurs="0" />
          </sequence>
        </complexType>
      </element>
      <element name="AppFileRef">
```

```
        <complexType>
          <sequence>
            <element name="aid" type="hexBinary" />
            <element name="efIDOrPath" type="hexBinary" />
          </sequence>
        </complexType>
      </element>
      <element name="AppTagRef">
        <complexType>
          <sequence>
            <element name="aid" type="hexBinary" />
            <element name="tag" type="string" />
            <element name="efIdOrPath"
                     type="hexBinary" maxOccurs="1" minOccurs="0" />
          </sequence>
        </complexType>
      </element>
    </choice>
    <element name="Index" type="hexBinary" maxOccurs="1" minOccurs="0" />
    <element name="Length" type="hexBinary" maxOccurs="1" minOccurs="0" />
  </sequence>
</complexType>
```

`<efIdOrPath> [choice]`

This element contains a file identifier or a path (as a sequence of file identifiers without separator) to a file on a smart card. According to ISO/IEC 7816-15 there are five cases to be differentiated:

- `<efIdOrPath>` is empty and no file is identified.
- `<efIdOrPath>` consists of one byte where the five most significant bits contain a short file identifier. The other bits b1,…,b3 are zero.
- `<efIdOrPath>` consists of exactly two bytes and contains an ordinary file identifier.
- `<efIdOrPath>` consists of an even number (>2) of bytes and contains an absolute or relative path (a sequence of file identifiers without separator) to a file.
- `<efIdOrPath>` consists of an odd number (>1) of bytes and contains a “qualified path” to a file as described in ISO/IEC 7816-4.

`<TagRef> [choice]`

This element consists of an element `<tag>` containing a tag encapsulating the addressed data and an optional element `<efIdOrPath>` as described above.

`<AppFileRef> [choice]`

With this element it is possible to reference a file of a certain application on the card. It consists of an element `<aid>` containing the identifier of the application and an element `<efIdOrPath>` as described above.

`<AppTagRef> [choice]`

With this element it is possible to reference a data object encapsulated with a tag and stored in a card application. It consists of an element `<aid>` containing the identifier of the application, an element `<tag>` containing a tag where the addressed data is stored and an optional element `<efIdOrPath>` as described above.

<Index> and <Length> [optional]

For these elements there are two cases to differentiate

- *transparent file*

  In this case the optional element <Index> contains the offset in the READ BINARY command (parameters P1/P2) and <Length> the parameter L_e.

- *record oriented file*

  In this case the element <Index> contains the record number in the READ RECORD command. The element <Length> will be ignored, if present.

The element <EF.ATR>is of type EFATRorINFOType and is part of the CardCapabilitiesType.. It contains information about the (EF.) ATR / INFO and is not necessary for the mapping of SAL-requests to APDUs, if the card is conforming to ISO/IEC 7816-4. This element may also be used to specify requirements for a card within a given profile.

```
<complexType name="EFATRorINFOType">
  <complexContent>
    <extension base="iso:FileRefReqType">
      <sequence>
        <element name="ISO7816-4-CardServiceData" maxOccurs="1"
                 type="iso:ISO7816-4-CardServiceDataType" minOccurs="0" />
        <element name="Pre-Issuing-DO"
                 type="iso:FileRefReqType" maxOccurs="1" minOccurs="0" />
        <element name="ISO7816-4-CardCapabilities" maxOccurs="1"
                 type="iso:ISO7816-4-CardCapabilitiesType" minOccurs="0" />
        <element name="ImplicitlySelectedApplication"
                 type="iso:FileRefReqType" maxOccurs="1" minOccurs="0" />
        <element name="ExtendedLengthInfo"
                 type="iso:ExtendedLengthInfoType" maxOccurs="1" minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

<ISO7816-4-CardServiceData> [optional]

This element MAY contain information about the card service data as described in Clause 8.1.1.2.3 of ISO/IEC 7816-4. Details of the type ISO7816-4-CardServiceDataTypeare described below.

<Pre-Issuing-DO> [optional]

This element MAY contain information about the path to manufacturer specific pre-issuing data. This element is not necessary if the card is conforming to ISO/IEC 7816-4. For details refer to the definition ofFileRefReqType.

<ISO7816-4-CardCapabilities> [optional]

This element MAY contain information about the card capabilities as described in ISO/IEC 7816-4. For details refer to the definition of the ISO7816-4-CardCapabilitiesType

<ImplicitlySelectedApplication> [optional]

This element MAY contain information about an implicit selected application as described in ISO/IEC 7816-4. For details refer to the definition of`FileRefReqType`.

`<ExtendedLengthInfo> [optional, unbounded]`

This element MAY contain information about the support of extended length. For details refer to the definition of `ExtendedLengthInfoType`.

The type `ISO7816-4-CardServiceDataType` is an extension of the `FileRefReqType`.. It is used by the element `ISO7816-4-CardServiceData`, which is part of the `EFATRorINFOType`.

```
<complexType name="ISO7816-4-CardServiceDataType">
  <complexContent>
    <extension base="iso:FileRefReqType">
      <sequence>
        <element name="Application-selection" maxOccurs="1" minOccurs="0">
          <complexType>
            <sequence>
              <element name="by-full-DF-name" type="iso:BitReqType" />
              <element name="by-partial-DF-name" type="iso:BitReqType" />
            </sequence>
          </complexType>
        </element>
        <element name="BER-TLV-data-objects-available"
                 maxOccurs="1" minOccurs="0">
          <complexType>
            <sequence>
              <element name="in-EF.DIR" type="iso:BitReqType" />
              <element name="in-EF.ATR" type="iso:BitReqType" />
            </sequence>
          </complexType>
        </element>
        <element name="EF.x-access-services" maxOccurs="1" minOccurs="0">
          <complexType>
            <choice>
              <element name="ReadBinary" type="iso:BasicRequirementsType" />
              <element name="ReadRecord" type="iso:BasicRequirementsType" />
              <element name="GetData" type="iso:BasicRequirementsType" />
            </choice>
          </complexType>
        </element>
        <element name="Root" maxOccurs="1" minOccurs="0">
          <complexType>
            <choice>
              <element name="Card-with-MF" type="iso:BasicRequirementsType" />
              <element name="Card-without-MF" type="iso:BasicRequirementsType" />
            </choice>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`<Application-selection> [optional]`

This element MAY contain information about the method to select a card application. It consists of two elements, each of type `BitReqType`::

- <by-full-DF-name> indicates that the card applications can be addressed by the full DF name (see Table 85 in ISO/IEC 7816-4).

- <by-partial-DF-name> indicates that the card applications can be addressed by the partial DF name (see Table 85 in ISO/IEC 7816-4).

Details of the type BitReqTypeare described below.

<BER-TLV-data-objects-available> [optional]

This element MAY contain information about the presence of BER-TLV coded data objects in EF.DIR and EF.ATR. It consists of two elements, each of type BitReqType::

- <in-EF.DIR> indicates the presence of BER-TLV coded data objects in EF.DIR.

- <in-EF.ATR> indicates the presence of BER-TLV coded data objects in EF.ATR.

Details of the type BitReqType are described below.

<EF.x-access-services> [optional]

This element MAY contain information about the method to access EF.DIR and EF.ATR. It consists of a choice between three elements, each of typeBasicRequirementsType::

- <ReadBinary> indicates that both files should be accessed via READ BINARY.

- <ReadRecord> indicates that both files should be accessed via READ RECORD.

- <GetData> indicates that both files should be accessed via GET DATA.

<Root> [optional]

This element MAY contain information about the presence of a root directory. It consists of a choice between two elements, each of typeBasicRequirementsType::

- <Card-with-MF> indicates that there is a MF on the Card.

- <Card-without-MF> indicates that there isn't a MF on the Card.

The type BitReqType expands the type RequirementsType by the element <Bit> which contains a bit represented as a boolean value.

```
<complexType name="BitReqType">
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="Bit" type="boolean" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The type ISO7816-4-CardCapabilitiesType is an extension of the FileRefReqType.. It is used by the element < ISO7816-4-CardCapabilities , which is part of theEFATRorINFOType.. The inherited element <Path> MAY contain a path to the three card capability bytes, as long as these are not part of the

(EF.) ATR (compact header tags '71', '72' and '73'). The `<Path>` element is not necessary, if the card is conforming to ISO/IEC 7816-4.

```
<complexType name="ISO7816-4-CardCapabilitiesType">
  <complexContent>
    <extension base="iso:FileRefReqType">
      <sequence>
        <element name="FirstSoftwareFunctionTable" maxOccurs="1" minOccurs="1">
          <complexType>
            <sequence>
              <element name="DF-selection" maxOccurs="1" minOccurs="0">
                <complexType>
                  <sequence>
                    <element name="by-full-DF-name" maxOccurs="1"
                             type="iso:BitReqType" minOccurs="0" />
                    <element name="by-partial-DF-name" maxOccurs="1"
                             type="iso:BitReqType" minOccurs="0" />
                    <element name="by-path" maxOccurs="1"
                             type="iso:BitReqType" minOccurs="0" />
                    <element name="by-file-identifier" maxOccurs="1"
                             type="iso:BitReqType" minOccurs="0" />
                    <element name="implicit" maxOccurs="1"
                             type="iso:BitReqType" minOccurs="0" />
                  </sequence>
                </complexType>
              </element>
              <element name="Short-EF-identifier" maxOccurs="1"
                       type="iso:BitReqType" minOccurs="0" />
              <element name="Record-number" maxOccurs="1"
                       type="iso:BitReqType" minOccurs="0" />
              <element name="Record-identifier" maxOccurs="1"
                       type="iso:BitReqType" minOccurs="0" />
            </sequence>
          </complexType>
        </element>
        <element name="SecondSoftwareFunctionTable">
          <complexType>
            <sequence>
              <element name="EFs-of-TLV-structure" maxOccurs="1"
                       type="iso:BitReqType" minOccurs="0" />
              <element name="Behaviour-of-write-functions"
                       maxOccurs="1" minOccurs="0">
                <complexType>
                  <complexContent>
                    <extension base="iso:RequirementsType">
                      <sequence>
                        <element name="Behaviour" type="iso:WriteBehaviourType" />
                      </sequence>
                    </extension>
                  </complexContent>
                </complexType>
              </element>
              <element name="Data-unit-size-in-quartets"
                       maxOccurs="1" minOccurs="0">
                <complexType>
                  <complexContent>
                    <extension base="iso:RequirementsType">
                      <sequence>
                        <element name="Exponent">
                          <simpleType>
                            <restriction base="integer">
                              <minInclusive value="1" />
                              <maxInclusive value="31" />
                            </restriction>
                          </simpleType>
```

```
                    </element>
                  </sequence>
                </extension>
              </complexContent>
            </complexType>
          </element>
          <element name="Value-FF-for-first-byte-of-BER-TLV-valid"
                   type="iso:BitReqType" maxOccurs="1" minOccurs="0" />
        </sequence>
      </complexType>
    </element>
    <element name="ThirdSoftwareFunctionTable">
      <complexType>
        <sequence>
          <element name="Command-chaining"
                   type="iso:BitReqType" maxOccurs="1" minOccurs="0" />
          <element name="Extended-Lc-and-Le-fields"
                   type="iso:BitReqType" maxOccurs="1" minOccurs="0" />
          <element name="Logical-Channel-support" maxOccurs="1" minOccurs="0">
            <complexType>
              <sequence>
                <element name="LC-Number-by-Card" type="iso:BitReqType" />
                <element name="LC-Number-by-IFD" type="iso:BitReqType" />
                <element name="Number-of-Logical-Channels"
                         maxOccurs="1" minOccurs="1">
                  <complexType>
                    <complexContent>
                      <extension base="iso:RequirementsType">
                        <sequence>
                          <element name="Maxium-Number">
                            <simpleType>
                              <restriction base="integer">
                                <minInclusive value="1" />
                                <maxInclusive value="8" />
                              </restriction>
                            </simpleType>
                          </element>
                        </sequence>
                      </extension>
                    </complexContent>
                  </complexType>
                </element>
              </sequence>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
  </sequence>
 </extension>
</complexContent>
</complexType>
```

The `WriteBehaviourType` MAY be used to specify detailed requirements of the behaviour of a smart card. See ISO/IEC 7816-4 for details.

```
<simpleType name="WriteBehaviourType">
  <restriction base="string">
    <enumeration value="One-time-write" />
    <enumeration value="Proprietary" />
    <enumeration value="Write-OR" />
    <enumeration value="Write-AND" />
  </restriction>
```

```
</simpleType>
```

The ExtendedLengthInfoType MAY be used to specify information about the support of extended length. This type is derived from the RequirementsType and defined as follows:

```
<complexType name="ExtendedLengthInfoType">
   <complexContent>
      <extension base="iso:RequirementsType">
         <sequence>
            <element name="GlobalLengthInfo" type="iso:LengthInfoType" />
            <element name="CommandSpecificLengthInfo"
                     type="iso:CommandSpecificLengthInfoType"
                     maxOccurs="unbounded" minOccurs="0" />
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

<GlobalLengthInfo> [required]

This element provides general extended length information, which is valid unless it is overruled by a CommandSpecificLengthInfo-element. It is of type LengthInfoType, which is specified below.

<CommandSpecificLengthInfo> [optional, unbounded]

This element MAY exist multiple times and is used to provide command specific extended length information. It is of type CommandSpecificLengthInfoType, which is specified below.

The LengthInfoType is used in the definition of the ExtendedLengthInfoType above and the definition of CommandSpecificLengthInfoType below. It is defined as follows:

```
<complexType name="LengthInfoType">
   <sequence>
      <element name="MaxNc" type="positiveInteger" />
      <element name="MaxNe" type="positiveInteger" />
   </sequence>
</complexType>
```

It contains the following elements:

<MaxNc> [required]

Indicates the maximum value of the command length (Nc), which is valid unless it is overruled by the MaxNc-element in some CommandSpecificLengthInfo-element.

<MaxNe> [required]

Indicates the maximum value of the expected response length (Ne), which is valid unless it is overruled by the MaxNe-element in some CommandSpecificLengthInfo-element.

The CommandSpecificLengthInfoType is used in the definition of the ExtendedLengthInfoType above and it is defined as follows:

```
<complexType name="CommandSpecificLengthInfoType">
   <sequence>
```

```
        <element name="Tag" type="byte"/>
        <element name="Command" type="hexBinary"/>
        <element name="LengthInfo" type="iso:LengthInfoType"/>
    </sequence>
</complexType>
```

It contains the following elements:

`<Tag> [required]`

Indicates the tag (‘81’ – ‘8F’) of the standardized data structure, which specifies what the command description in the `Command`-element includes (see ISO/IEC 7816-4).

`<Command> [required]`

Specifies the (part of the) command header, for which the extended length information in the `LengthInfo`-element is provided.

`<LengthInfo> [required]`

Provides the command specific extended length information for the command identified by `Command` and `Tag`. This element is of type `LengthInfoType` specified above.

D.3.6 ApplicationCapabilities

The `<ApplicationCapabilities>` element is part of the `CardInfoType`(cf. Clause `ApplicationCapabilitiesType`). It is of type `ApplicationCapabilitiesType` and provides detailed information about the card applications available on the card. When used for a card which contains an application capability description as defined in ISO/IEC 24727-2 and/or is fully described by an appropriate ISO/IEC 7816-15 structure this element may be omitted.

```
<complexType name="ApplicationCapabilitiesType">
    <sequence>
      <element name="ImplicitlySelectedApplication"
               type="iso:ApplicationIdentifierType" maxOccurs="1" minOccurs="0" />
      <element name="CardApplication" maxOccurs="unbounded"
               type="iso:CardApplicationType" minOccurs="0" />
      <any namespace="##any" processContents="lax" minOccurs="0"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional" />
  </complexType>
```

`<ImplicitlySelectedApplication> [optional]`

This element MAY contain information which card application is implicitly selected after initialization.

`<CardApplication> [optional, unbounded]`

This element MAY contain a description for each card application available on the card.

Furthermore there MAY be some additional element, which structure is defined by some other specification.

The `<CardApplication>` element of type `CardApplicationType` is part of the `ApplicationCapabilitiesType`. The `CardApplicationType` is an extension of the type `RequirementsType` containing the `<RequirementLevel>` element.

```
<complexType name="CardApplicationType">
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="InterfaceProtocol"
                 type="anyURI" maxOccurs="unbounded" minOccurs="0" />
        <element name="ApplicationIdentifier" maxOccurs="1"
                 type="iso:ApplicationIdentifierType" minOccurs="1" />
        <element name="ApplicationName"
                 type="string" maxOccurs="1" minOccurs="0" />
        <element name="CardApplicationServiceInfo" maxOccurs="unbounded"
                 type="iso:CardApplicationServiceType" minOccurs="0" />
        <element name="DIDInfo"
                 type="iso:DIDInfoType" maxOccurs="unbounded" minOccurs="0" />
        <element name="DataSetInfo"
                 type="iso:DataSetInfoType" maxOccurs="unbounded" minOccurs="0" />
        <any namespace="##any" processContents="lax" minOccurs="0"/>
      </sequence>
      <attribute name="Id" type="ID" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

`<InterfaceProtocol> [optional, unbounded]`

This element MAY contain a reference to a protocol for the access to the card application, which corresponds to the `Interface`-element.

`<ApplicationIdentifier>`

This element contains the application identifier of the card application.

`<ApplicationName> [optional]`

This element MAY contain an additional name of the card application.

`<CardApplicationServiceInfo> [optional, unbounded]`

This element MAY contain information about the services supported by the card application. For details refer to the definition of `CardApplicationServiceType`.

`<DIDInfo> [optional, unbounded]`

This element MAY be present for each differential identity (DID) of the card application and specifies the related details of this DID. For details refer to the definition of `DIDInfoType`.

`<DataSetInfo> [optional, unbounded]`

This element MAY be present for each data set (and embodied data structures for interoperability) of the card application. For details refer to the definition of `DataSetInfoType`.

Furthermore there MAY be some additional element, which structure is defined by some other specification.

The `<CardApplicationServiceInfo>` element of type `CardApplicationServiceType` is part of the definition of `CardApplicationServiceType`. The `CardApplicationServiceType` is an extension of the type `RequirementsType` containing the `< RequirementLevel>` element.

```
<complexType name="CardApplicationServiceType">
```

```
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="CardApplicationServiceName" type="string" />
        <element name="ServiceACL"
                 type="iso:AccessControlListType" />
        <element name="CardApplicationServiceDescription" maxOccurs="1"
                 type="iso:CardApplicationServiceDescriptionType" minOccurs="0" />
        <any namespace="##any" processContents="lax" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

<CardApplicationServiceName>

This element contains the name of the card application service.

<ServiceACL>

This element contains access control information for the card application service (the structure of the AccessControlListType is explained below).

<CardApplicationServiceDescription> [optional]

This element MAY contain an interface description of the card application service. For details refer to the definition of CardApplicationServiceDescriptionType below. If the card application service is standardized this element is not necessary.

Furthermore there MAY be some additional element, which structure is defined by some other specification.

The AccessControlListType contains a sequence of AccessRule-elements.

```
<complexType name="AccessControlListType">
    <sequence>
        <element name="AccessRule" type="iso:AccessRuleType"
            maxOccurs="unbounded" minOccurs="0"/>
    </sequence>
</complexType>
```

<AccessRule> [optional, unbounded]

This element MAY define an access rule, which is of type AccessRuleType. The AccessRuleType is defined as follows

```
<complexType name="AccessRuleType">
    <sequence>
        <element name="CardApplicationServiceName" type="string"
            maxOccurs="1" minOccurs="0" />
        <element name="Action" type="iso:ActionNameType" />
        <element name="SecurityCondition" type="iso:SecurityConditionType"/>
    </sequence>
</complexType>
```

and contains the following elements:

<CardApplicationServiceName> [optional]

This element MAY indicate the CardApplicationServiceName, if this information is not provided outside the AccessControlList as it is the case in the definition of the CardApplicationServiceType.

<Action>

This element specifies the action as defined in this standard or a loaded onto the card.

<SecurityCondition>

This element specifies the security conditions, which are necessary to perform the action specified above. As defined in this standard the security condition may be specified as a Boolean combination of DIDAuthenticationState-elements.

The CardApplicationServiceDescriptionType is defined as follows

```
<complexType name="CardApplicationServiceDescriptionType">
   <choice>
       <element name="ServiceDescriptionURL" type="anyURI" />
       <element name="ServiceDescriptionText" type="string" />
   </choice>
</complexType>
```

It supports the following alternatives

<ServiceDescriptionURL> [choice]

This element MAY contain a URL at which the ServiceDescriptionText may be found.

<ServiceDescriptionText> [choice]

This element MAY contain the description of the service.

The <DIDInfo> element of type DIDInfoType is part of the CardApplicationType. The DIDInfoType is an extension of the type RequirementsTypecontaining the < RequirementLevel> element.

```
<complexType name="DIDInfoType">
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="DifferentialIdentity"
                 type="iso:DifferentialIdentityType" />
        <element name="DIDACL" type="iso:AccessControlListType" />
        <any namespace="##any" processContents="lax" minOccurs="0"/>
      </sequence>
      <attribute name="Id" type="ID" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

<DifferentialIdentity>

According to this standard all key material used for authentication or other cryptographic operations is represented by a differential identity (DID). The DifferentialIdentityTypeis described in detail below.

<DIDACL>

This element contains the access control list defining the access rights to the differential identities. For more details refer to the definition of the AccessControlListType.

Furthermore there MAY be some additional element, which structure is defined by some other specification.

The <DifferentialIdentity> element of type DifferentialIdentityType is part of theDIDInfoType.

```
<complexType name="DifferentialIdentityType">
  <sequence>
    <element name="DIDName" type="iso:DIDNameType" />
    <element name="DIDProtocol" type="anyURI" />
    <element name="DIDMarker" type="iso:DIDMarkerType" />
    <element name="DIDScope" type="iso:DIDScopeType" minOccurs="0" />
    <element name="DIDQualifier"
             type="iso:DIDQualifierType" minOccurs="0" maxOccurs="1" />
  </sequence>
</complexType>
```

<DIDName> [required]

This element contains the name (unique in the scope) which is used in function calls to refer to the differential identity.

<DIDProtocol> [required]

This element specifies the protocol of the differential identity. Standardized protocols are defined in Annex A of this document for example.

<DIDMarker> [required]

The structure of this element depends on the < DIDProtocol> element.

<DIDScope> [optional]

This element MAY specify whether the differential identity can be accessed from all card applications of the card (global) or only within a certain card application (local).

<DIDQualifier> [optional]

This element MAY contain further information on the differential identity. The structure of this element is defined as follows:

```
<complexType name="DIDQualifierType">
    <choice>
        <element name="ApplicationIdentifier"
                 type="iso:ApplicationIdentifierType" />
        <element name="ObjectIdentifier" type="anyURI" />
        <element name="ApplicationFunction" type="iso:BitString" />
    </choice>
</complexType>
```

The <DataSetInfo> element of type DataSetInfoTypeis part of the CardApplicationType. The DataSetInfoType is an extension of the type RequirementsType containing the < RequirementLevel> element.

```
<complexType name="DataSetInfoType">
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="DataSetName" type="iso:DataSetNameType" />
        <element name="DataSetACL" type="iso:AccessControlListType" />
        <element name="DataSetPath" type="iso:PathType" />
        <element name="DSI"
                 type="iso:DSIType" maxOccurs="unbounded" minOccurs="0" />
        <any namespace="##any" processContents="lax" minOccurs="0"/>
      </sequence>
      <attribute name="Id" type="ID" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

`<DataSetName> [required]`

This element contains the name of a data set which can be used to address the data set in SAL calls.

`<DataSetACL> [required]`

This element contains the access rules on the data set.

`<DataSetPath> [required]`

This element contains the path to a data set.

`<DSI> [optional, unbounded]`

If the data set contains data structures for interoperability (DSIs) each of these elements MAY contain further information about one DSI. Details of the `DSIType` are specified below.

Furthermore there MAY be some additional element, which structure is defined by some other specification. Such an additional element MAY be used for post-issuance personalization purposes for example.

The `DSIType` is an extension of the type `RequirementsType`and is specified as follows:

```
<complexType name="DSIType">
  <complexContent>
    <extension base="iso:RequirementsType">
      <sequence>
        <element name="DSIName" type="iso:DSINameType" />
        <element name="DSIPath" type="iso:PathType" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`<DSIName> [required}`

This element contains the name of the DSI which is used for addressing the DSI in procedure calls.

`<DSIPath> [required]`

This element contains the path to the DSI on the card..

D.3.7 Signature

To defeat attacks which make use of manipulated CardInfo-structures one MAY apply a digital signature according to RFC 3275 to protect the `CardInfo`-element or chosen parts of it.

When signed CardInfo files are imported, the signature SHALL be verified according to an appropriate policy.

Furthermore it SHOULD be ensured that

— especially protected key material for which the `<Protected>` element of the `KeyRefType` is TRUE and

— potential dangerous APDUs in the `CommandAPDU` element with CLA values '0*x*' or '1*x*', *x*∈{0,…,9, A,…,F} and INS values '20', '21', '24', '2C', and '22'

are only accessible or executed, if they are protected by an appropriate signature. CardInfo-files with protected key material or dangerous APDU and without an appropriate signature SHOULD be rejected.

In a signature on a CardInfo file the signer SHOULD only use the `ds:X509Data` choice in the `KeyInfo` element (see Section 4.4 in RFC 3275).

All parts to be signed SHALL be specified with `ds:Reference` elements. Every element to be included in the signature SHALL have an `Id` attribute which is used in the `URI` attribute of the `ds:Reference` element. As the simplest case the whole `<CardInfo>` element is referenced, so that the signature is part of the signed data and therefore is excluded when calculating the hash value ("enveloped signature"). On the other hand it is possible to sign only child elements of the `<CardInfo>` element (e.g. `<CardType>`, `<CardIdentification>` and `<CardCapabilities>`) as well as existing `<CardApplication>` elements (contained in the element `<Applicationcapabilities>`) so that it is possible to add a complete card application (on the card and in the CardInfo file) without invalidating the existing signature. The tests described above assure that an attacker is not able to use an unsigned part of the CardInfofile to access sensitive key objects which are protected by a signature.

D.4 Complete XML-Schema Definition (CardInfo.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

    <include schemaLocation="ISO24727-3.xsd" />

    <include schemaLocation="ISO24727-Protocols.xsd" />

    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"
/>

    <import namespace="urn:oasis:names:tc:dss:1.0:core:schema"
        schemaLocation="http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-schema-
v1.0-os.xsd" />


    <!-- ================================
        CardInfo-element
        ================================
        The CardInfo-element contains
        the necessary information for
        the mapping of generic requests
```

```
        at the Service Access Layer to
        card-specific APDUs in case there
        is no appropriate CIA according
        to ISO/IEC 7816-15 available on
        the card. Furthermore this
        structure may be used to specify
        card profiles.
        ================================  -->

    <element name="CardInfo" type="iso:CardInfoType" />


    <complexType name="CardInfoType">
        <sequence>
            <!--
                The CardType-element contains a unique
                identifier for the card type and
                optionally further links to specification
                documents.
            -->
            <element name="CardType" type="iso:CardTypeType" />

            <!--
                The CardIdentification-element
                allows to determine the type of
                a given card by walking through
                the decision tree and checking
                whether the characteristic features
                are as expected.
            -->
            <element name="CardIdentification"
                type="iso:CardIdentificationType" />


            <!--
                The CardCapabilities-element
                specifies the general capabilities
                of the card and may be omitted, if
                the card fully complies to ISO/IEC 7816-4.
            -->
            <element name="CardCapabilities"
                type="iso:CardCapabilitiesType" maxOccurs="1" minOccurs="0" />

            <!--
                The ApplicationCapabilities-element
                provides detailled information about the
                card applications available on the card.
            -->
            <element name="ApplicationCapabilities"
                type="iso:ApplicationCapabilitiesType" maxOccurs="1" minOccurs="0"
/>

            <!--
                The Signature-elements may be used to
                protect the integrity and authenticity
                of (parts of) the CardInfo-element.
            -->
            <element name="Signature" type="ds:SignatureType"
```

```
            maxOccurs="unbounded" minOccurs="0" />
        </sequence>
        <attribute name="Id" type="ID" use="optional" />
    </complexType>


    <!-- ============================
        ApplicationInfo-element
        ============================
        The ApplicationInfo-element
        allows to specify card applications
        which may be downloaded to a card
        by a Card Application Management System.
        ============================  -->

    <element name="ApplicationInfo" type="iso:CardApplicationType" />


    <!-- ============================  -->
    <!-- Definition of basic types in  -->
    <!--       alphabetic order         -->
    <!-- ============================  -->


    <complexType name="ApplicationCapabilitiesType">
        <sequence>
            <element name="ImplicitlySelectedApplication"
                type="iso:ApplicationIdentifierType" maxOccurs="1" minOccurs="0" />

            <element name="CardApplication"
                type="iso:CardApplicationType" maxOccurs="unbounded" minOccurs="0"
/>
            <element name="Other" type="dss:AnyType" minOccurs="0" />
        </sequence>
        <attribute name="Id" type="ID" use="optional" />
    </complexType>


    <complexType name="ApplicationDataRefType">
        <complexContent>
            <extension base="iso:DataRefType">
                <sequence>
                    <element name="Application"
                        type="iso:ApplicationIdentifierType" maxOccurs="1"
minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="ATRInterfaceBytesType">
        <sequence>
            <element name="TAi" type="iso:ByteMaskType" maxOccurs="1"
                minOccurs="0" />
            <element name="TBi" type="iso:ByteMaskType" maxOccurs="1"
                minOccurs="0" />

            <element name="TCi" type="iso:ByteMaskType" maxOccurs="1"
                minOccurs="0" />
```

```
        <element name="TDi" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="0" />

    </sequence>
</complexType>

<complexType name="ATRType">
    <sequence>
        <element name="TS" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="1" />
        <element name="T0" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="1" />

        <element name="InterfaceBytes">
            <complexType>
                <sequence>
                    <element name="Tx1"
                        type="iso:ATRInterfaceBytesType" />

                    <element name="Tx2"
                        type="iso:ATRInterfaceBytesType" />

                    <element name="Tx3"
                        type="iso:ATRInterfaceBytesType" />

                    <element name="Tx4"
                        type="iso:ATRInterfaceBytesType" />

                </sequence>
            </complexType>
        </element>
        <element name="HistoricalBytes" maxOccurs="1"
            minOccurs="0">
            <complexType>
                <sequence maxOccurs="15" minOccurs="0">
                    <element name="Ti" type="iso:ByteMaskType" />

                </sequence>
            </complexType>
        </element>
        <element name="TCK" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="0" />
    </sequence>
</complexType>


<complexType name="ATSInterfaceBytesType">
    <sequence>
        <element name="TA1" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="0" />
        <element name="TB1" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="0" />

        <element name="TC1" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="0" />

    </sequence>
</complexType>

<complexType name="ATSType">
```

```
    <sequence>
        <element name="TL" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="1" />
        <element name="T0" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="1" />

        <element name="InterfaceBytes"
            type="iso:ATSInterfaceBytesType" />

        <element name="HistoricalBytes">
            <complexType>
                <sequence maxOccurs="15" minOccurs="0">
                    <element name="Ti" type="iso:ByteMaskType" />

                </sequence>
            </complexType>
        </element>
        <element name="CRC1" type="iso:ByteMaskType" maxOccurs="1"
            minOccurs="1" />

        <element name="CRC2" type="iso:ByteMaskType" />
    </sequence>
</complexType>


<simpleType name="BasicRequirementsType">
    <restriction base="string">
        <enumeration value="PlatformMandatory" />
        <enumeration value="PlatformOptional" />
        <enumeration value="PersonalizationMandatory" />
        <enumeration value="PersonalizationOptional" />
    </restriction>
</simpleType>

<complexType name="BitReqType">
    <complexContent>
        <extension base="iso:RequirementsType">
            <sequence>
                <element name="Bit" type="boolean" />
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ByteMaskType">
    <sequence>
        <element name="Value" type="hexBinary" maxOccurs="1"
            minOccurs="1" />
        <element name="Mask" type="hexBinary" maxOccurs="1"
            minOccurs="1" />

    </sequence>
</complexType>

<complexType name="CardApplicationServiceType">
    <complexContent>

        <extension base="iso:RequirementsType">
            <sequence>
                <element name="CardApplicationServiceName"
```

```
                    type="string" />
                <element name="CardApplicationServiceDescription"
                    maxOccurs="1" minOccurs="0"
                    type="iso:CardApplicationServiceDescriptionType" />
                <element name="Other" type="dss:AnyType" minOccurs="0" />
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="CardApplicationType">
    <complexContent>
        <extension base="iso:RequirementsType">
            <sequence>
                <element name="InterfaceProtocol" type="anyURI"
                    maxOccurs="unbounded" minOccurs="0" />
                <element name="ApplicationIdentifier"
                    type="iso:ApplicationIdentifierType" maxOccurs="1"
minOccurs="1" />
                <element name="ApplicationName" type="string"
                    maxOccurs="1" minOccurs="0" />
                <element name="LocalApplicationName"
type="dss:InternationalStringType" maxOccurs="unbounded" minOccurs="0">
                </element>
                <element name="CardApplicationServiceInfo"
                    type="iso:CardApplicationServiceType" maxOccurs="unbounded"
                    minOccurs="0" />
                <element name="CardApplicationACL"
type="iso:AccessControlListType" />
                <element name="DIDInfo" type="iso:DIDInfoType"
                    maxOccurs="unbounded" minOccurs="0" />
                <element name="DataSetInfo" maxOccurs="unbounded" minOccurs="0"
type="iso:DataSetInfoType" />
                <element name="Other" type="dss:AnyType" minOccurs="0" />
            </sequence>
            <attribute name="Id" type="ID" use="optional" />
        </extension>
    </complexContent>
</complexType>

<complexType name="CardCapabilitiesType">
    <sequence>
        <element name="Interface" maxOccurs="unbounded"
            minOccurs="0">
            <complexType>
                <complexContent>
                    <extension base="iso:RequirementsType">
                        <sequence>
                            <element name="Protocol" type="anyURI" />
                        </sequence>
                    </extension>
                </complexContent>
            </complexType>
        </element>
        <element name="EF.DIR" maxOccurs="1" minOccurs="0"
            type="iso:FileRefReqType" />
        <element name="EF.ATRorINFO" maxOccurs="1" minOccurs="0"
            type="iso:EFATRorINFOType" />
        <element name="Other" type="dss:AnyType" minOccurs="0" />
    </sequence>
```

```
    <attribute name="Id" type="ID" use="optional" />
</complexType>


<complexType name="CardIdentificationType">
    <sequence>
        <element name="ATR" maxOccurs="unbounded" minOccurs="0"
            type="iso:ATRType" />
        <element name="ATS" type="iso:ATSType" maxOccurs="1"
            minOccurs="0" />
        <element name="CharacteristicFeature" maxOccurs="unbounded"
            minOccurs="0">
            <complexType>
                <sequence maxOccurs="unbounded" minOccurs="1">
                    <element name="CardCall"
                        type="iso:CardCallType" />
                </sequence>
            </complexType>
        </element>
        <element name="Other" type="dss:AnyType" minOccurs="0" />
    </sequence>
    <attribute name="Id" type="ID" use="optional" />
</complexType>


<complexType name="CardTypeType">
    <sequence>
        <element name="ProfilingInfo" maxOccurs="1" minOccurs="0">
            <complexType>
                <sequence>
                    <element name="BasisSpecification"
                        type="anyURI" />

                    <element name="ProfilingRelation"
                        type="iso:ProfilingType" />
                </sequence>
            </complexType>
        </element>
        <element name="ObjectIdentifier" type="anyURI" />
        <element name="SpecificationBodyOrIssuer" type="string"
            maxOccurs="1" minOccurs="0" />
        <element name="CardTypeName"
            type="dss:InternationalStringType" maxOccurs="unbounded"
            minOccurs="0" />
        <element name="Version" maxOccurs="1" minOccurs="0">
            <complexType>
                <sequence>
                    <element name="Major" type="string" />
                    <element name="Minor" type="string"
                        maxOccurs="1" minOccurs="0" />
                    <element name="SubMinor" type="string"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </complexType>
        </element>
        <element name="Status" type="string" maxOccurs="1"
            minOccurs="0" />
        <element name="Date" type="date" maxOccurs="1"
            minOccurs="0" />
        <element name="CardInfoRepository" type="anyURI"
```

```
            maxOccurs="1" minOccurs="0" />
         <element name="Other" type="dss:AnyType" minOccurs="0" />
      </sequence>
      <attribute name="Id" type="ID" use="optional" />
   </complexType>

   <complexType name="DataSetInfoType">
      <complexContent>

         <extension base="iso:RequirementsType">
            <sequence>
               <element name="DataSetName"
                  type="iso:DataSetNameType" />
               <element name="LocalDataSetName"
type="dss:InternationalStringType" maxOccurs="unbounded" minOccurs="0">
               </element>
               <element name="DataSetACL"
                  type="iso:AccessControlListType" />
               <element name="DataSetPath" type="iso:PathType"
                  maxOccurs="1" minOccurs="1" />

               <element name="DSI" type="iso:DSIType"
                  maxOccurs="unbounded" minOccurs="0" />
               <element name="Other" type="dss:AnyType" minOccurs="0" />
            </sequence>
            <attribute name="Id" type="ID" use="optional" />
         </extension>
      </complexContent>
   </complexType>

   <complexType name="DIDInfoType">
      <complexContent>
         <extension base="iso:RequirementsType">
            <sequence>
               <element name="DifferentialIdentity"
                  type="iso:DifferentialIdentityType" />
               <element name="DIDACL"
                  type="iso:AccessControlListType" />
               <element name="Other" type="dss:AnyType" minOccurs="0" />
            </sequence>
            <attribute name="Id" type="ID" use="optional" />
         </extension>
      </complexContent>
   </complexType>


   <complexType name="DifferentialIdentityType">
      <sequence>
         <element name="DIDName" type="iso:DIDNameType" />
         <element name="LocalDIDName" type="dss:InternationalStringType"
maxOccurs="unbounded" minOccurs="0">
         </element>
         <element name="DIDProtocol" type="anyURI" />
         <element name="DIDMarker" type="iso:DIDMarkerType" />
         <element name="DIDScope" minOccurs="0"
            type="iso:DIDScopeType" />
         <element name="DIDQualifier" minOccurs="0"
            type="iso:DIDQualifierType" maxOccurs="1" />
      </sequence>
   </complexType>
```

```
    <complexType name="DSIType">
        <complexContent>
            <extension base="iso:RequirementsType">
                <sequence>
                    <element name="DSIName" type="iso:DSINameType" />
                    <element name="DSIPath" type="iso:PathType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>


    <complexType name="EFATRorINFOType">
        <complexContent>
            <extension base="iso:FileRefReqType">
                <sequence>
                    <element name="ISO7816-4-CardServiceData"
                        maxOccurs="1" minOccurs="0"
                        type="iso:ISO7816-4-CardServiceDataType" />

                    <element name="Pre-Issuing-DO"
                        type="iso:FileRefReqType" maxOccurs="1" minOccurs="0" />

                    <element name="ISO7816-4-CardCapabilities"
                        maxOccurs="1" minOccurs="0"
                        type="iso:ISO7816-4-CardCapabilitiesType" />

                    <element name="ImplicitlySelectedApplication"
                        type="iso:FileRefReqType" maxOccurs="1" minOccurs="0" />

                    <element name="ExtendedLengthInfo" maxOccurs="1"
                        minOccurs="0" type="iso:ExtendedLengthInfoType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="ExtendedLengthInfoType">
        <complexContent>
            <extension base="iso:RequirementsType">
                <sequence>
                    <element name="GlobalLengthInfo"
                        type="iso:LengthInfoType" />
                    <element name="CommandSpecificLengthInfo"
                        type="iso:CommandSpecificLengthInfoType"
maxOccurs="unbounded"
                        minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="LengthInfoType">
        <sequence>
            <element name="MaxNc" type="positiveInteger" />
            <element name="MaxNe" type="positiveInteger" />
        </sequence>
    </complexType>
```

```
    <complexType name="CommandSpecificLengthInfoType">
        <sequence>
            <element name="Tag" type="byte" />
            <element name="Command" type="hexBinary" />
            <element name="LengthInfo" type="iso:LengthInfoType" />
        </sequence>
    </complexType>


    <complexType name="FileRefReqType">
        <complexContent>
            <extension base="iso:RequirementsType">
                <sequence>
                    <element name="Path" type="iso:PathType"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>


    <complexType name="ISO7816-4-CardCapabilitiesType">
        <complexContent>
            <extension base="iso:FileRefReqType">
                <sequence>
                    <element name="FirstSoftwareFunctionTable"
                        maxOccurs="1" minOccurs="1">
                        <complexType>
                            <sequence>
                                <element name="DF-selection"
                                    maxOccurs="1" minOccurs="0">
                                    <complexType>
                                        <sequence>
                                            <element
                                                name="by-full-DF-name"
type="iso:BitReqType" maxOccurs="1"
                                                minOccurs="0" />
                                            <element
                                                name="by-partial-DF-name"
type="iso:BitReqType"
                                                maxOccurs="1" minOccurs="0" />
                                            <element name="by-path"
                                                type="iso:BitReqType" maxOccurs="1"
minOccurs="0" />
                                            <element
                                                name="by-file-identifier"
type="iso:BitReqType"
                                                maxOccurs="1" minOccurs="0" />
                                            <element name="implicit"
                                                type="iso:BitReqType" maxOccurs="1"
minOccurs="0" />
                                        </sequence>
                                    </complexType>
                                </element>
                                <element name="Short-EF-identifier"
                                    type="iso:BitReqType" maxOccurs="1" minOccurs="0"
/>
                                <element name="Record-number"
```

```
                        type="iso:BitReqType" maxOccurs="1" minOccurs="0"
/>
                    <element name="Record-identifier"
                        type="iso:BitReqType" maxOccurs="1" minOccurs="0"
/>
                </sequence>
            </complexType>
        </element>
        <element name="SecondSoftwareFunctionTable">
            <complexType>
                <sequence>
                    <element name="EFs-of-TLV-structure"
                        type="iso:BitReqType" maxOccurs="1" minOccurs="0"
/>
                    <element
                        name="Behaviour-of-write-functions" maxOccurs="1"
                        minOccurs="0">

                        <complexType>
                            <complexContent>
                                <extension
                                    base="iso:RequirementsType">
                                    <sequence>
                                        <element
                                            name="Behaviour"
type="iso:WriteBehaviourType" />

                                    </sequence>
                                </extension>
                            </complexContent>
                        </complexType>
                    </element>
                    <element
                        name="Data-unit-size-in-quartets" maxOccurs="1"
                        minOccurs="0">
                        <complexType>
                            <complexContent>
                                <extension
                                    base="iso:RequirementsType">
                                    <sequence>
                                        <element
                                            name="Exponent">
                                            <simpleType>
                                                <restriction
                                                    base="integer">
                                                    <minInclusive
                                                        value="1" />
                                                    <maxInclusive
                                                        value="31" />
                                                </restriction>
                                            </simpleType>
                                        </element>
                                    </sequence>
                                </extension>
                            </complexContent>
                        </complexType>
                    </element>
                    <element
                        name="Value-FF-for-first-byte-of-BER-TLV-valid"
```

```
                                type="iso:BitReqType" maxOccurs="1" minOccurs="0"
/>
                        </sequence>
                    </complexType>
                </element>
                <element name="ThirdSoftwareFunctionTable">
                    <complexType>
                        <sequence>
                            <element name="Command-chaining"
                                type="iso:BitReqType" maxOccurs="1" minOccurs="0"
/>
                            <element
                                name="Extended-Lc-and-Le-fields"
type="iso:BitReqType"
                                maxOccurs="1" minOccurs="0" />
                            <element name="Logical-Channel-support"
                                maxOccurs="1" minOccurs="0">
                                <complexType>
                                    <sequence>
                                        <element
                                            name="LC-Number-by-Card"
type="iso:BitReqType" />

                                        <element
                                            name="LC-Number-by-IFD"
type="iso:BitReqType" />
                                        <element
                                            name="Number-of-Logical-Channels"
maxOccurs="1"
                                            minOccurs="1">
                                            <complexType>
                                                <complexContent>
                                                    <extension
                                                        base="iso:RequirementsType">
                                                        <sequence>
                                                            <element
                                                                name="Maxium-Number">
                                                                <simpleType>
                                                                    <restriction
                                                                        base="integer">
                                                                        <minInclusive
                                                                            value="1" />
                                                                        <maxInclusive
                                                                            value="8" />
                                                                    </restriction>
                                                                </simpleType>
                                                            </element>
                                                        </sequence>
                                                    </extension>
                                                </complexContent>
                                            </complexType>
                                        </element>
                                    </sequence>
                                </complexType>
                            </element>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </extension>
```

```
    </complexContent>
</complexType>


<complexType name="ISO7816-4-CardServiceDataType">
    <complexContent>
        <extension base="iso:FileRefReqType">
            <sequence>
                <element name="Application-selection" maxOccurs="1"
                    minOccurs="0">
                    <complexType>
                        <sequence>
                            <element name="by-full-DF-name"
                                type="iso:BitReqType" />

                            <element name="by-partial-DF-name"
                                type="iso:BitReqType" />

                        </sequence>
                    </complexType>
                </element>
                <element name="BER-TLV-data-objects-available"
                    maxOccurs="1" minOccurs="0">
                    <complexType>
                        <sequence>
                            <element name="in-EF.DIR"
                                type="iso:BitReqType" />

                            <element name="in-EF.ATR"
                                type="iso:BitReqType" />

                        </sequence>
                    </complexType>
                </element>
                <element name="EF.x-access-services" maxOccurs="1"
                    minOccurs="0">
                    <complexType>
                        <choice>
                            <element name="ReadBinary"
                                type="iso:BasicRequirementsType" />

                            <element name="ReadRecord"
                                type="iso:BasicRequirementsType" />

                            <element name="GetData"
                                type="iso:BasicRequirementsType" />

                        </choice>
                    </complexType>
                </element>
                <element name="Root" maxOccurs="1" minOccurs="0">
                    <complexType>
                        <choice>
                            <element name="Card-with-MF"
                                type="iso:BasicRequirementsType" />

                            <element name="Card-without-MF"
                                type="iso:BasicRequirementsType" />

                        </choice>
```

```
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>


<simpleType name="ProfilingType">
    <restriction base="string">
        <enumeration value="extends" />
        <enumeration value="redefines" />
    </restriction>
</simpleType>


<complexType name="RequirementsType">
    <sequence maxOccurs="1" minOccurs="1">
        <element name="RequirementLevel"
            type="iso:BasicRequirementsType" maxOccurs="1" minOccurs="0" />
    </sequence>
</complexType>


<simpleType name="VariantIndicatorType">
    <restriction base="string">
        <enumeration value="CommandWithoutSM" />
        <enumeration value="ResponseWithoutSM" />
        <enumeration value="CommandWithSM" />
        <enumeration value="ResponseWithSM" />
    </restriction>
</simpleType>


<simpleType name="WriteBehaviourType">
    <restriction base="string">
        <enumeration value="One-time-write" />
        <enumeration value="Proprietary" />
        <enumeration value="Write-OR" />
        <enumeration value="Write-AND" />
    </restriction>
</simpleType>

</schema>
```

Annex E
(informative)

XML Binding for Selected Authentication Protocols

This clause contains the protocol-specific definitions of the Crypto and Differential Identity Services for a set of authentication protocols in accordance with ISO/IEC 24727-3.

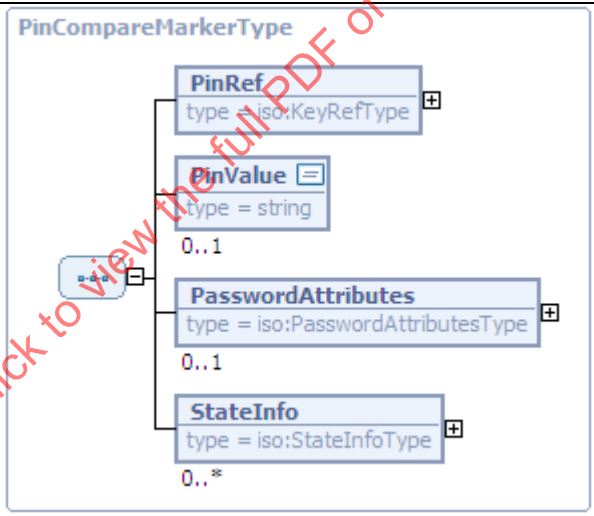
E.1 PIN Compare

Authentication of a user is performed by means of a PIN in this protocol, which is also specified in a bridged form in Annex A.9 of this standard.

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) pin-compare(9) }`.

Note: A comparable protocol is urn:oid:1.3.162.15480.3.0.9 for iso(1) identified-organization (3) CEN (162) CEN 15480 (15480) part3(3) annex-a(0) pin-compare(9). This should be deprecated in favor of the ISO OID.

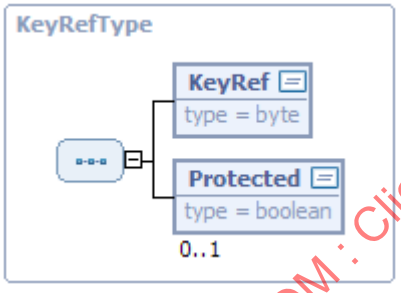
E.1.1 Marker



This type specifies the structure of the DID marker for this authentication protocol.

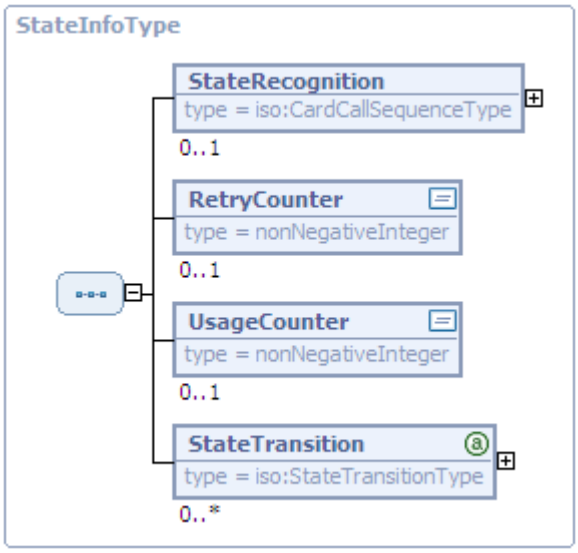
Name	Description
`PinRef`	Contains the key reference of the PIN. Details on the `KeyRefType` can be found below.
`PinValue`	MAY contain the value of the PIN. If this value is missing despite being required, it is input at the terminal. The PIN needs to be stated in the following cases:

	— When creating the PIN object with `DIDCreate`. — When changing the PIN with `DIDUpdate` (see Clause E.1.3) and — CHANGE REFERENCE DATA (i.e. `unblockingPassword` bit in `pwdFlags`) is not set) — RESET RETRY COUNTER (i.e. the `unblockingPassword` bit in `pwdFlags` is set) in the following cases: — P1='00' (indicated by the absence of `resetRetryCounter1` and `resetRetryCounter2` in the PasswordFlags bit string) or — P1='02' (indicated by the setting of `resetRetryCounter1` and the absence of `resetRetryCounter2` in the `PasswordFlags` bit string)
`PasswordAttributes`	Is an optional element which can contain the `PasswordAttributes` defined in part 6 of this standard.
`StateInfo`	MAY contain information about the designated states if more than one status is defined for the key object. There is one `StateInfo` element for each possible status and the current status corresponds to the first `StateInfo` element in the sequence. Details on the `StateInfoType` can be found below.



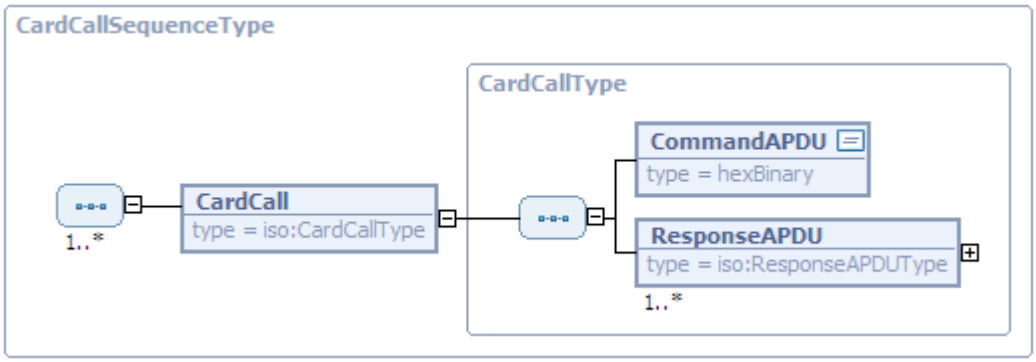
The `KeyRefType` is used for the specification of markers (e.g. in the `PinCompareMarkerType`, `MutualAuthMarkerType`, `RSAAuthMarkerType` and the `CryptoMarkerType`).

Name	Description
`KeyRef`	Contains the reference to the key object.
`Protected`	Is an optional element with which key objects can be marked as particularly sensitive (e.g. private signature key or PIN for qualified electronic signature). If such key objects exist in a `CardInfo` file, they SHOULD be covered by a signature issued by a trustworthy organisation, and any attempt to access a key object of this type from an unsigned part of a `CardInfo` file will result in an error message during import of the `CardInfo` file.



The `StateInfoType` is used to specify states. A status is clearly identified by the necessary attribute `StateName`.

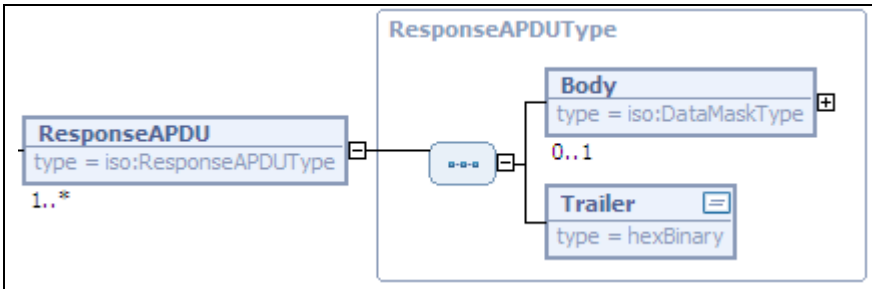
Name	Description
`StateRecognition`	MAY indicate the sequence of calls to the card through which the existing status can be determined. The `CardCallSequenceType` is explained in greater detail below. The commands in the respective `StateRecognition` elements SHOULD be selected in such a way as to permit the clear identification of the status of the key object.
`RetryCounter`	Is an optional element which MAY contain the `RetryCounter` for the key object in accordance with ISO/IEC 7816-4, Table 34. Depending on the context in which this element is used, the maximum value (`DIDCreate`, `DIDUpdate`) or the current value of the counter (`DIDGet`) is given.
`UsageCounter`	Is an optional element which MAY contain the `UsageCounter` for the key object in accordance with ISO/IEC 7816-4, Table 34. Depending on the context in which this element is used, the maximum value (`DIDCreate`, `DIDUpdate`) or the current value of the counter (`DIDGet`) is given.
`StateTransition`	Contains information on the various state transitions provided for the PIN.



The `CardCallSequenceType` defines a sequence of smart card commands in the form of elements of the `CardCallType`, which are used, for example, to identify the card type, to check whether a key object is in a certain state or, as in the `StateTransition` element, to simply execute a preset sequence of smart card commands.

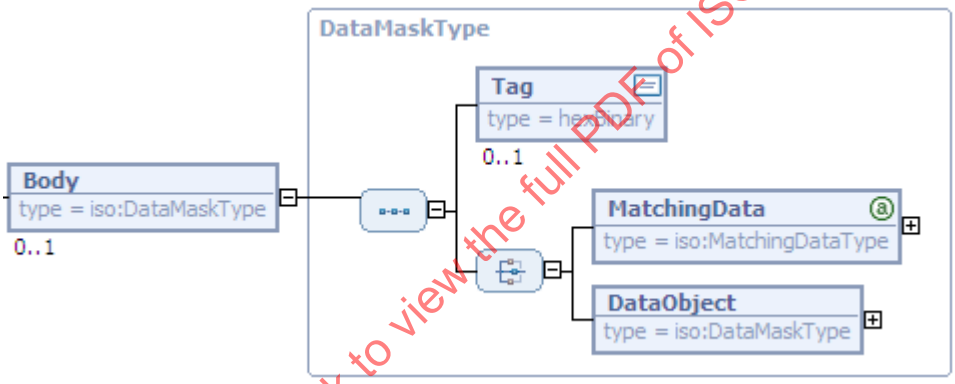
A status is considered to have been identified if for **ALL** `CommandAPDU` calls (i.e. AND operation) the reply from the smart card agrees with **ONE** reply specified in the sequence of ResponseAPDU elements (i.e. OR operation).

Name	Description
`CardCall`	Defines a call to the smart card which is given by a `CommandAPDU` element and a sequence of possible `ResponseAPDU` elements. The sequence of the `CardCall` elements is to be understood as an AND operation when identifying a status or card type.
`CommandAPDU`	Contains the APDU command which is to be sent to the card (cf. ISO/IEC 7816-4, ISO/IEC 7816-8 and ISO/IEC 7816-9). For security reasons no APDUs with CLA values '0x' or '1x' SHOULD be used, where x is any half byte, nor should the INS values '20', '21', '24', '2C' and '22' be used, as this would correspond to the smart card commands VERIFY, CHANGE REFERENCE DATA, RESET RETRY COUNTER and MANAGE SECURITY ENVIRONMENT (see ISO/IEC 7816-4), which a hacker could use to increment the error operation counter in the event of a malicious "card or status detection" and thereby provoke a denial-of-service attack under some circumstances. In each case such APDUs SHOULD be denied if the `CardCall` element is not signed by a trustworthy body .
`ResponseAPDU`	Defines a sequence of valid replies from the smart card for the identification of a certain status or smart card type. The sequence of the `ResponseAPDU` elements is to be understood as an OR operation when identifying a status or card type. The structure of the `ResponseAPDUType` is explained in detail below.



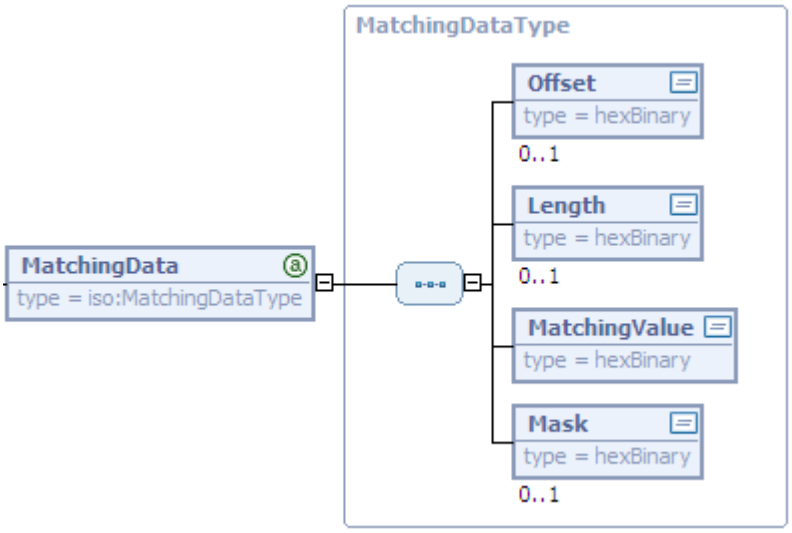
The `ResponseAPDU` element is part of a `CardCall` element and specifies a possible reply from a card when invoking a `CommandAPDU`. The `ResponseAPDU` comprises an (optional) `body` and a `trailer` (if successful equal to '9000'). The structure of the `DataMaskType` defined below allows any parts to be filtered out of the `body` element and used to check for matching data.

Name	Description
`Body`	MAY contain information indicating which part of the data returned by the smart card is relevant to check for matching data. Further details on the `DataMaskType` can be found below.
`Trailer`	Contains the expected status of the invocation (if successful '9000').



The `body` element is of the type `DataMaskType` and MAY be part of the `ResponseAPDU` element and contains a `tag` element and either `MatchingData` or a structured `DataObject` of the type `DataMaskType`.

Name	Description
`Tag`	MAY contain the tag under which the expected value (or another structured data object) is filed.
`MatchingData`	Specifies which parts of the returned data are relevant in terms of checking for matching data. The detailed structure of the `MatchingData` is explained below.
`DataObject`	Is of the type `DataMaskType` and therefore again contains information on the selective filtering of a complex data object, thus allowing analysis of TLV-encoded structures irrespective of how they are nested.



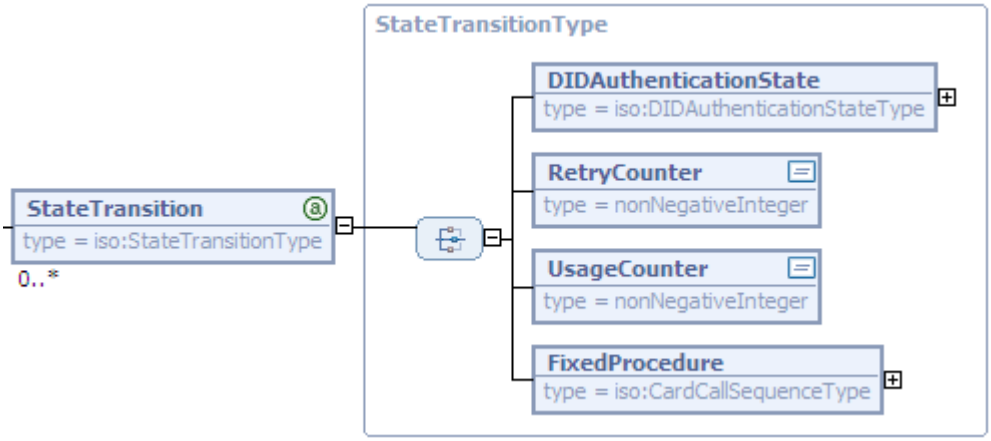
The `MatchingData` structure makes it possible to specify which parts of a data object returned in the body are relevant for the comparison.

Name	Description
`Offset`	MAY optionally contain an offset which is taken into account when determining the relevant value. For example, an offset of '03' and a mask *xy* would have the same effect as a mask '00 00 00 *xy*'.
`Length`	MAY contain the length of the value relevant for the comparison.

`MatchingValue`	Contains the value which is either identical to the value returned by the card or contained therein.
	
	The optional attribute `MatchingRule` of the type `MatchingRuleType` in the `MatchingDataType` determines whether to check if the values are equal or contained. This type is defined as follows:
	
	`<simpleType name="MatchingRuleType">`
	`<restriction base="string">`
	`<enumeration value="Equals" />`
	`<enumeration value="Contains" />`
	`</restriction>`
	`</simpleType>`
	
	If this attribute is omitted then `Equals` is the default setting.

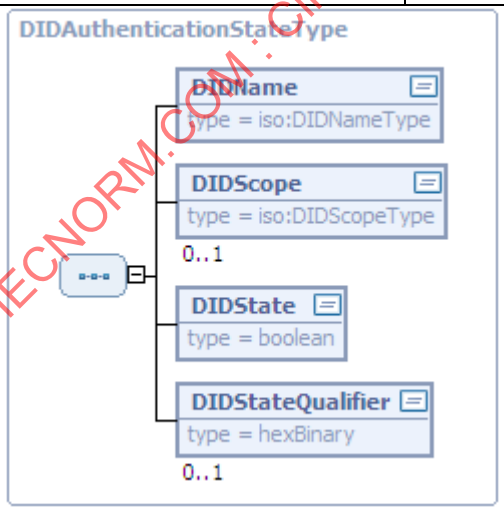
`Mask`	The OPTIONAL `Mask` element, which is linked conjunctively with the `MatchingValue` element and the data returned by the card, makes it

	possible to filter out the significant parts of the data.



Elsewhere there is a description of the `StateInfoType` which contains an arbitrary number of `StateTransition` elements, used to describe the possible state transitions which can be reached from the state described by the `StateInfoType`. The `StateTransition` element has a mandatory `TargetState` attribute, used to refer to the `StateName` attribute of an existing state.

Name	Description
`DIDAuthenticationState`	Indicates that the transition from one state to another can proceed by means of authentication with the DID stated in this element. Details on the `DIDAuthenticationStateType` can be found below.
`RetryCounter`	Indicates that the transition from one state to another is made by reaching the value stated for the `RetryCounter` of the key object.
`UsageCounter`	Indicates that the transition from one state to another is made by reaching the value stated for the `UsageCounter` of the key object.
`FixedProcedure`	Indicates that the transition from one state to another is made by executing a fixed sequence of smart card commands.



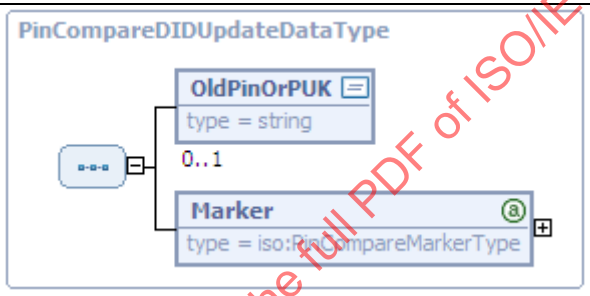
The `DIDAuthenticationStateType` is used in the specification of access rights and in the

specification of state transitions (cf. `StateTransition` above).	
Name	**Description**
`DIDName`	Specifies the name of the DID required for authentication on transition from one state to another.
`DIDScope`	MAY, if required, resolve ambiguities between local and global DIDs.
`DIDState`	Specifies the required authentication status and SHOULD be allocated the value `True` when specifying state transitions.
`DIDStateQualifier`	MAY contain further information which is evaluated when using certificate-based authentication procedures.

E.1.2 DIDCreate

With DIDCreate use is made of DIDUpdateData of the type PinCompareMarkerType.

E.1.3 DIDUpdate



This type specifies the structure of the `DIDUpdateDataType` for the PIN Compare protocol.

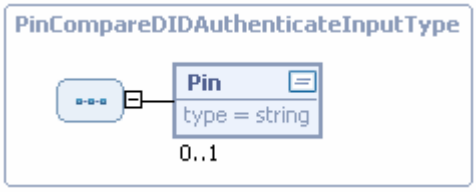
Name	**Description**
`OldPinOrPUK`	MAY contain the old PIN or the Personal Unblocking Key (PUK). If this information is missing despite being required then it is input at the terminal. This information is needed if the settings in the `pwdFlags` element are as follows: 1. The `exchangeRefData` bit is set but not the `unblockingPassword` bit (i.e. it is a normal PIN which can only be changed by entering the old PIN (CHANGE REFERENCE DATA with P1='00', cf. ISO/IEC 7816-4)) or 2. The `unblockingPassword` bit is set but not the `resetRetryCounter1` bit (i.e. it is a PUK which SHOULD be presented upon the RESET RETRY COUNTER command with P1='00' or P1='01' (cf. ISO/IEC 7816-4)).
`Marker`	Contains the new marker for the PinCompare protocol.

E.1.4 DIDGet

In the case of `DIDGet` there is a return of `DIDStructure` containing data of the type `PinCompareMarkerType`, although the `PinValue` element is omitted, and the current reading of any `RetryCounter` or `UsageCounter` which may be defined is stated in the first `StateInfo` element which describes the current status.

E.1.5 DIDAuthenticate

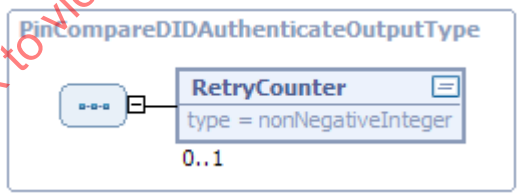
The protocol is processed by a single invocation of `DIDAuthenticate` with the following entry:



This type specifies the structure of the `DIDAuthenticationDataType` for the PIN Compare protocol when `DIDAuthenticate` is called up.

Name	Description
`Pin`	MAY contain the value of the PIN. If this element is missing, it is input at the terminal.

The return to `DIDAuthenticateResponse` is as follows:



This type specifies the structure of the `DIDAuthenticationDataType` for the PIN Compare protocol when `DIDAuthenticate` is returned.

Name	Description
`RetryCounter`	If user verification failed, this contains the current value of the `RetryCounter`.

E.1.6 Non-supported functions

The following functions are not supported with this protocol and, when called up, relay an error message to this effect /resultminor/sal#inappropriateProtocolForAction:

— CardApplicationStartSession

— Encipher

— Decipher

— GetRandom

— Hash

— Sign

— VerifySignature

— VerifyCertificate

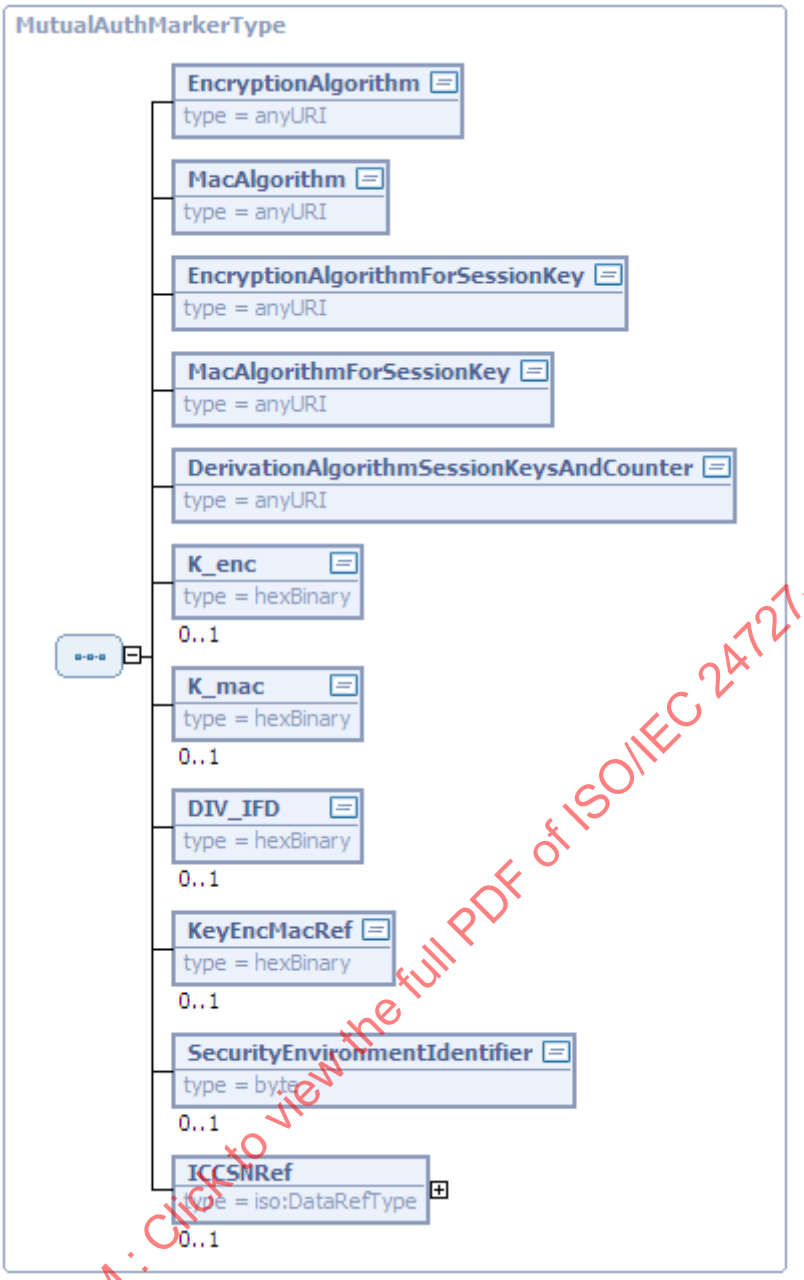
E.2 Mutual authentication

This protocol is specified in similar form in Annex A.12 of this standard and Section 8.8 of EN14890-1 and provides the framework for mutual authentication with the exchange of keys using symmetrical algorithms.

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) client-application-mutual-authentication-wke(12) }`.

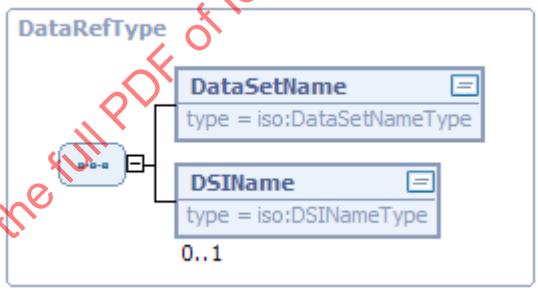
Note: A similar protocol is urn:oid:1.3.162.15480.3.0.12 for iso(1) identified-organization (3) CEN (162) CEN 15480 (15480) part3(3) annex-a(0) client-application-mutual-authentication-wke(12). This protocol should be deprecated in favor of the ISO OID.

E.2.1 Marker



This type specifies the structure of the DID marker for this authentication protocol.

Name	Description
`EncryptionAlgorithm`	Specifies the encryption algorithm to be used on establishing connection.
`MacAlgorithm`	Specifies the MAC algorithm to be used on establishing connection.
`EncryptionAlgorithm ForSessionKey`	Specifies the encryption algorithm to be used for secure messaging.
`MacAlgorithmForSessionKey`	Specifies the MAC algorithm to be used for secure messaging.
`DerivationAlgorithm`	Specifies the algorithm required to identify the session key and

SessionKeysAndCounter	counters.
K_enc	MAY contain the encryption key to be used on establishing connection.
K_mac	MAY contain the MAC key to be used on establishing connection.
DIV_IFD	MAY contain an initialisation vector for the employment of the symmetrical algorithms. If there is no entry, a sequence of 0x00 bytes which is suitable for the respective algorithm is used as an initialisation vector.
KeyEncMacRef	Contains the key reference to the symmetrical key pair (for encryption and MAC formation).
SecurityEnvironment Identifier	Is an optional element by means of which a security environment which deviates from the standard MAY be specified.
ICCSNRef	MAY contain a reference to the serial number of the card. If this reference is known elsewhere, the entry MAY be omitted here. The DataRefType is as follows:  If it is a transparent file, the DSIName MAY be omitted.

E.2.2 DIDCreate

With DIDCreate use is made of DIDUpdateData of the type MutualAuthMarkerType.

E.2.3 DIDUpdate

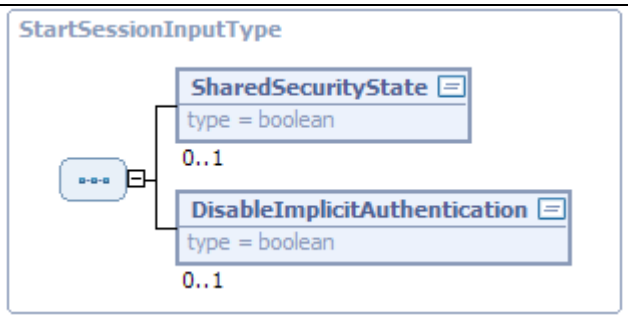
With DIDUpdate use is made of DIDUpdateData of the type MutualAuthMarkerType.

E.2.4 DIDGet

In the case of DIDGet there is a return of DIDStructure containing data of the type MutualAuthMarkerType

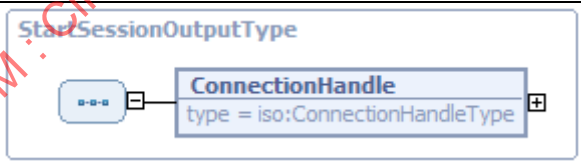
E.2.5 CardApplicationStartSession

A session to the ICC is established when `CardApplicationStartSession` is invoked with the optional parameter of the `StartSessionInputType`, which can be subsequently used with the `ConnectionHandle` returned in the `StartSessionOutputType`.



This type specifies the structure of the `DIDAuthenticationDataType` when `CardApplicationStartSession` is called up.

Name	Description
`SharedSecurityState`	If this flag is set to `True`, any channel which may have been established between the client application (or an `SAM assigned to it`) and the `ICC` is also used with the returned `ConnectionHandle`. If no such channel exists then one is established. If this element is missing or if it is `False`, a new logical channel is always established to the `ICC`. If the card does not support any logical channels, an error /resultminor/sal#functionNotSupported is returned.
`DisableImplicitAuth`	If authentication is necessary for access to the specified DIDs, this MAY be implicitly initiated by default (if this element is missing or is `False`). If this flag is set to `True`, the necessary authentication is not implicitly initiated. Therefore, it is possible that an error /resultminor/sal#securityConditionsNotSatisfied MAY occur.



This type specifies the structure of the `DIDAuthenticationDataType` in `CardApplicationStartSessionResponse`.

Name	Description
`ConnectionHandle`	Enables use of the session opened within `CardApplicationStartSession` in future function calls.

The procedure for setting up a session is approximately as follows:

1. Identify DID information for ICC by means of `DIDGet` and read out ICCSN by means of `DataSetSelect` and `DSIRead`.
2. Request a random number from ICC, form the challenge from the random number and ICCSN and invoke `DIDAuthenticate` for InternalAuthenticate on SAM.
3. Invocation of `DIDAuthenticate` for MutualAuthenticate on ICC with result from step 2.
4. Invocation of `DIDAuthenticate` for ExternalAuthenticate on SAM with result from Step 3.

E.2.6 DIDAuthenticate

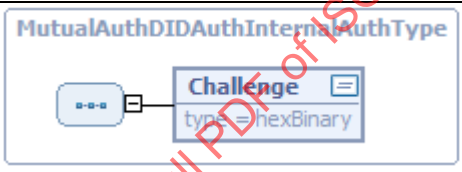
DIDAuthenticate is used in this protocol for the following purposes:

— To invoke InternalAuthenticate

— To invoke MutualAuthenticate

— To invoke ExternalAuthenticate

E.2.6.1 Invocation of InternalAuthenticate



This type specifies the structure of the `DIDAuthenticationDataType` for the Mutual Authentication protocol when `DIDAuthenticate` is invoked to request INTERNAL AUTHENTICATE.

Name	Description
`Challenge`	Contains the challenge of the communication partner which is to be encrypted and assigned a MAC when INTERNAL AUTHENTICATE is requested.

E.2.6.2 Invocation of MutualAuthenticate

The return to `DIDAuthenticateResponse` after INTERNAL AUTHENTICATE, which also serves as the input for the next step (`DIDAuthenticate` for MUTUAL AUTHENTICATE), is as follows in this case:

This type specifies the structure of the `DIDAuthenticationDataType` for the Mutual Authentication

protocol when DIDAuthenticate is returned after INTERNAL AUTHENTICATE or before MUTUAL AUTHENTICATE.	
Name	**Description**
InternalCryptogram	Contains the cryptogram generated in the previous step.

E.2.6.3 Invocation of ExternalAuthenticate

The return to DIDAuthenticateResponse after MUTUAL AUTHENTICATE, which also serves as the input for the next step (DIDAuthenticate for EXTERNAL AUTHENTICATE), is as follows in this case:

This type specifies the structure of the DIDAuthenticationDataType for the Mutual Authentication protocol when DIDAuthenticate is called up to request EXTERNAL AUTHENTICATE.

Name	**Description**
MutualCryptogram	Contains the cryptogram generated in the previous step.

E.2.7 Non-supported functions

The following functions are not supported with this protocol and return a corresponding error message /resultminor/sal#inappropriateProtocolForAction when called up:

— Encipher

— Decipher

— Hash

— Sign

— VerifySignature

— VerifyCertificate

E.2.8 Minimum requirements in terms of algorithms

This authentication protocol can fundamentally be used with various cryptographic algorithms, but the following algorithms SHOULD be supported as a minimum:

— EncryptionAlgorithm

  - urn:oid:1.2.840.113549.3.7 for des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) des-ede3-cbc(7)}

- MacAlgorithm
  - urn:oid:1.2.840.113549.3.7 for des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) des-ede3-cbc(7)}

- DerivationAlgorithmSessionKeysAndCounter
  - urn:oid:1.2.840.63.0 for x9-63-scheme ::= { iso(1) member-body(2) US(840) ansi-x9-63(63) schemes(0) }

- MacAlgorithmForSessionKey
  - urn:oid:1.2.840.113549.3.7 for des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) des-ede3-cbc(7)}

- EncryptionAlgorithmForSessionKey
  - urn:oid:1.2.840.113549.3.7 for des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) des-ede3-cbc(7)}

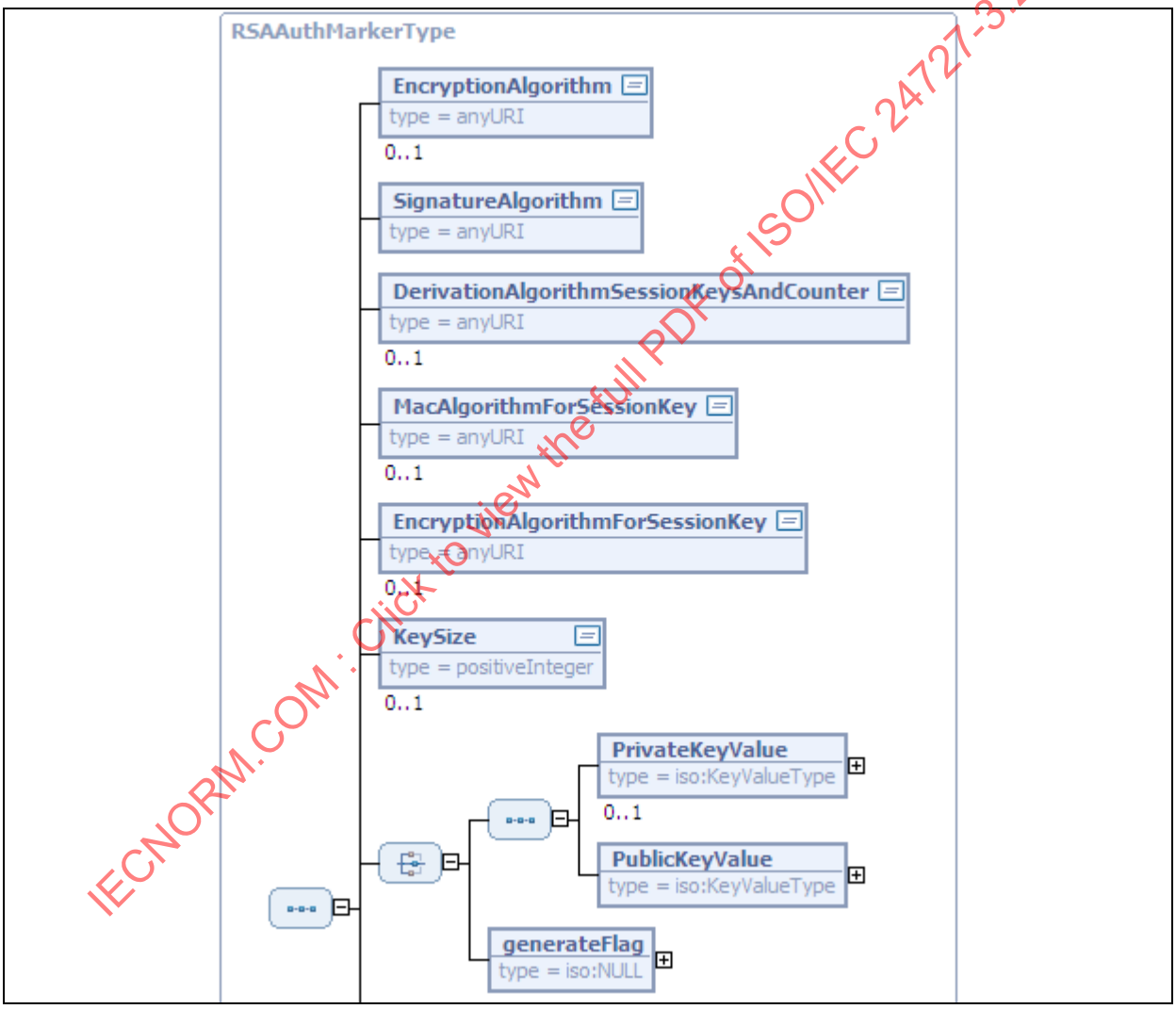
E.3 RSA Authentication

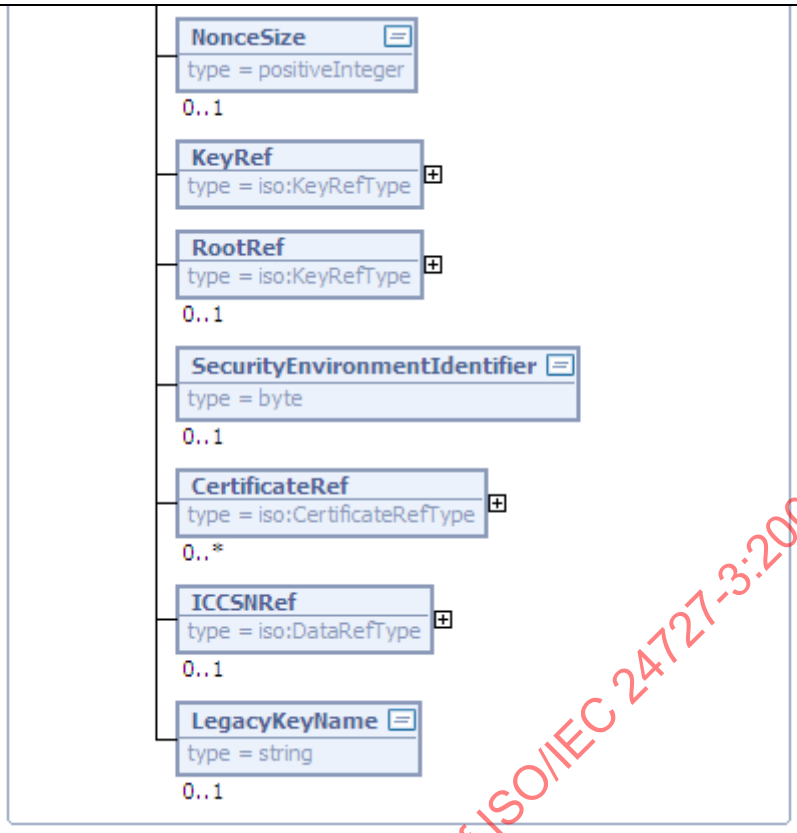
This protocol is specified in similar form in Annex A.15 of this standard as *Key Transport with mutual authentication based on RSA* and Section 8.4 of EN14890-1 and provides the framework for mutual authentication with an optional exchange of keys using the RSA algorithm.

This authentication protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) key-transport-with-mutual-authentication(15) }`.

Note: A similar protocol is urn:oid:1.3.162.15480.3.0.15 for iso(1) identified-organization (3) CEN (162) CEN 15480 (15480) part3(3) annex-a(0) key-transport-with-mutual-authentication(15). This protocol should be deprecated in favor of the ISO protocol.

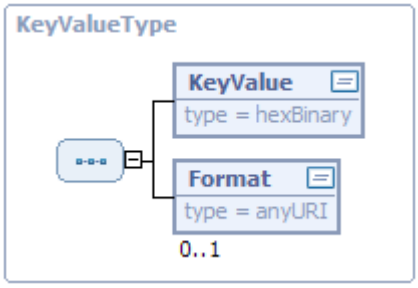
E.3.1 Marker



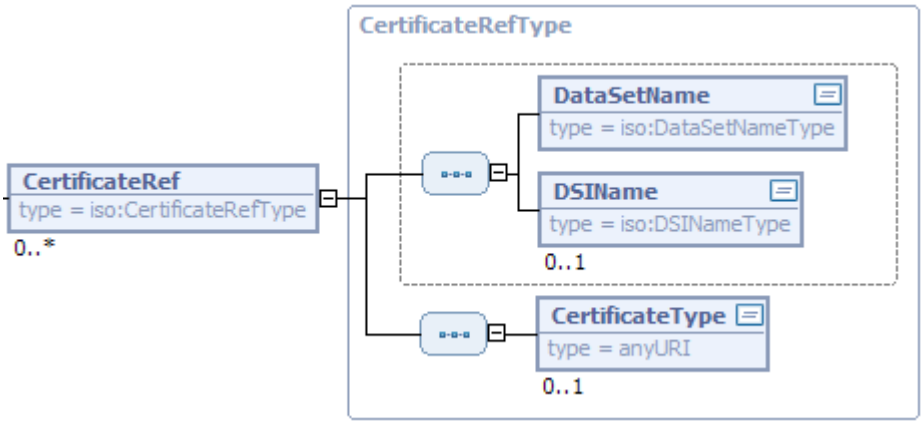


This type specifies the structure of the DID marker for this authentication protocol.

Name	Description
`EncryptionAlgorithm`	MAY contain the encryption algorithm to be used in the scope of the optional key exchange. If this element is omitted then mutual authentication takes place without a key agreement.
`SignatureAlgorithm`	Specifies the signature algorithm to be used for mutual authentication.
`DerivationAlgorithmSessionKeysAndCounter`	Specifies the algorithm required to identify the session key and counters.
`MacAlgorithmForSessionKey`	Specifies the MAC algorithm to be used for secure messaging.
`EncryptionAlgorithmForSessionKey`	Specifies the encryption algorithm to be used for secure messaging.
`KeySize`	MAY contain the bit length of the RSA module.
`PrivateKeyValue`	MAY contain the private key. The structure of the `KeyValueType` is as follows:

<table>
<tr><td></td><td>The KeyValueType contains key information and an optional format specification.
 

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>KeyValue</td><td>Contains the value of the key (in the specified format).</td></tr>
<tr><td>Format</td><td>MAY contain the URI of the format employed.</td></tr>
</table></td></tr>
<tr><td>PublicKeyValue</td><td>Contains the public key of the DID if the generateFlag alternative has not been selected.</td></tr>
<tr><td>generateFlag</td><td>Specifies that the key pair belonging to the DID is to be generated on the card.</td></tr>
<tr><td>NonceSize</td><td>MAY contain the byte length of the challenges to be used in this authentication protocol.</td></tr>
<tr><td>KeyRef</td><td>SHOULD contain the reference of the private key of the DID.</td></tr>
<tr><td>RootRef</td><td>MAY contain the reference of the trustworthy root key of the DID. If this element is omitted, the reference to the root key SHOULD come from the certificate requiring verification.</td></tr>
<tr><td>SecurityEnvironment Identifier</td><td>Is an optional element by means of which a security environment which deviates from the standard MAY be specified.</td></tr>
<tr><td>CertificateRef</td><td>MAY contain a sequence of references to certificates which are stored on the card. If these are card-verifiable certificates which are to be used to verify the certificate path with the VERIFY CERTIFICATE command (ISO/IEC 7816-8), they SHOULD be specified in the sequence required to verify the certificate path, i.e. starting from a trustworthy root. Further details on the CertificateRefType can be found below.</td></tr>
<tr><td>ICCSNRef</td><td>MAY contain a reference to the serial number of the card.. If this reference is known elsewhere, e.g. by a CardInfo file, the element MAY be dispensed with here.</td></tr>
<tr><td>LegacyKeyName</td><td>MAY contain a name which can be used to address the DID in a legacy application, e.g. a Microsoft Cryptographic API provider.</td></tr>
</table>



`CertificateRef` is part of the `RSAAuthQualifierType` and contains a reference to a certificate saved on the card. This type extends the `DataRefType` by adding the `CertificateType` element.

Name	Description
`CertificateType`	MAY specify the type of certificate. The following certificate types are notably supported: — urn:ietf:rfc:3280 - for X.509 public key certificates in accordance with RFC3280 — urn:ietf:rfc:3281 - for X.509-based attribute certificates in accordance with RFC3281 — urn:iso:std:iso-iec:7816:-8:tech:certificate:<*aid*>:<*cpi*> for a card-variable certificate, whereby <*aid*> contains the application identifier of a registered card application and <*cpi*> contains a corresponding certificate profile identifier (tag '5F29').

E.3.2 DIDCreate

With `DIDCreate` use is made of `DIDCreationData` of the `RSAAuthMarkerType`.

E.3.3 DIDUpdate

With `DIDUpdate` use is made of `DIDUpdateData` of the `RSAAuthMarkerType`.

E.3.4 DIDGet

With `DIDGet` use is made of `DIDDiscoveryData` of the `RSAAuthMarkerType`.

E.3.5 CardApplicationStartSession

The procedure for setting up a session is approximately as follows:

1. Determine DID information for ICC by means of `DIDGet` and read out corresponding certificates using `DataSetSelect` and `DSIRead`.
2. Determine DID information for SAM by means of `DIDGet` and read out corresponding certificates using `DataSetSelect` and `DSIRead`.

3. Invoke `DIDAuthenticate` to verify certificate path on ICC by successive invocations of `VerifyCertificate` with the certificates determined in step 1.
4. Invoke `DIDAuthenticate` to verify certificate path on SAM by successive invocations of `VerifyCertificate` with the certificates determined in step 2.
5. Request a Challenge from SAM and invoke `DIDAuthenticate` for InternalAuthenticate on ICC.
6. Invoke `DIDAuthenticate` for ExternalAuthenticate on SAM with result from step 5.
7. Request a Challenge from ICC and invoke `DIDAuthenticate` for InternalAuthenticate on SAM.
8. Invoke `DIDAuthenticate` for ExternalAuthenticate on ICC with result from step 7.

E.3.6 DIDAuthenticate

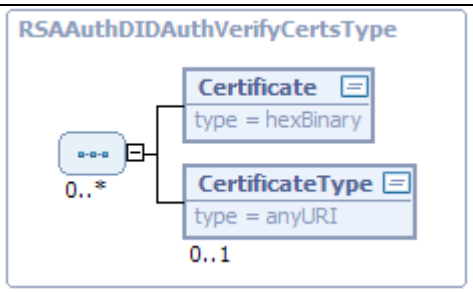
DIDAuthenticate is used in this protocol for the following purposes:

— To verify the certificate path

— To invoke InternalAuthenticate

— To invoke ExternalAuthenticate

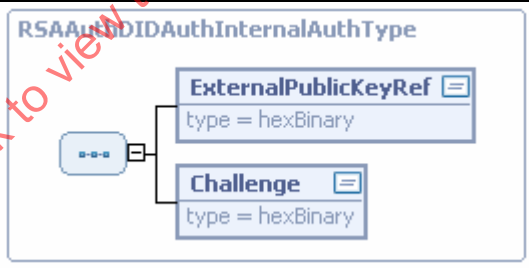
E.3.6.1 Verification of the certificate path



This type specifies the structure of the `DIDAuthenticationDataType` for the RSA Key Transport protocol on invocation of `DIDAuthenticate` for verification of the certificate path and contains a sequence of certificates which are used in the specified sequence to verify the certificate path — starting from a common root.

Name	Description
`Certificate`	Contains a certificate.
`CertificateType`	MAY specify which type of certificate is involved.

E.3.6.2 Invocation of InternalAuthenticate



This type specifies the structure of the `DIDAuthenticationDataType` for the RSA Authentication protocol when `DIDAuthenticate` is invoked to invoke INTERNAL AUTHENTICATE.

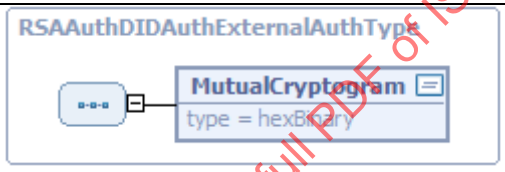
Name	Description
`ExternalPublicKeyRef`	Contains the key reference of the public key of the message recipient. In the case of the electronic health insurance card, for example, this key reference consists of two zero bytes and the counterpart ICCSN which is stored in EF.GDO.
`Challenge`	Contains the challenge of the communication partner which is to be signed by invoking INTERNAL AUTHENTICATE.

In this case the return to DIDAuthenticateResponse is as follows:

This type specifies the structure of the DIDAuthenticationDataType for the RSA Authentication protocol when DIDAuthenticate is returned after INTERNAL AUTHENTICATE.

Name	Description
Signature	Contains the signature generated during this request.

E.3.6.3 Invocation of ExternalAuthenticate



This type specifies the structure of the DIDAuthenticationDataType for the RSA Authentication protocol when DIDAuthenticate is invoked to invoke EXTERNAL AUTHENTICATE.

Name	Description
MutualCryptogram	Contains the signature to be verified by means of EXTERNAL AUTHENTICATE.

E.3.6.4 VerifyCertificate

The certificate which has been transmitted is checked against the public key of the trustworthy root referenced in RootCert.

The optional element DIDQualifier is not present when VerifyCertificate is invoked.

E.3.6.5 Non-supported functions

The following functions are not supported with this protocol and return a corresponding error message /resultminor/sal#inappropriateProtocolForAction when called up:

— Encipher

— Decipher

— Hash

— Sign

— VerifySignature

E.3.6.6 Minimum requirements in terms of algorithms

This authentication protocol can basically be used with various cryptographic algorithms but the following algorithms SHOULD be supported as a minimum:

— EncryptionAlgorithm

  — urn:oid: 1.2.840.113549.1.1.1 for RSA encryption ::= {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) rsa-encryption(1)}

— SignatureAlgorithm

  — urn:oid:1.3.36.3.4.3.2.1 for sigS-ISO9796-2rndWithsha1 ::= {iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureScheme(4) sigS-ISO9796-2rnd(3) sigS-ISO9796-2rndWithrsa(2) sigS-ISO9796-2rndWithsha1(1)}

  — urn:oid:1.3.36.3.4.3.2.4 for sigS-ISO9796-2rndWithsha256 ::= {iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureScheme(4) sigS-ISO9796-2rnd(3) sigS-ISO9796-2rndWithrsa(2) sigS-ISO9796-2rndWithsha256(4)}

— DerivationAlgorithmSessionKeysAndCounter

  — urn:oid:1.3.162.14890.1.x.1 for CEN14890-KDF-Simple::= {iso(1) identified-organization (3) CEN (162) CEN 14890 (15480) part-1 (1) key-derivation (x) simple-scheme (1)} in accordance with Section 8.4.2 from EN14890-1.

— MacAlgorithmForSessionKey

  — urn:oid:1.2.840.113549.3.7 for des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) des-ede3-cbc(7)}

— EncryptionAlgorithmForSessionKey

  — urn:oid:1.2.840.113549.3.7 für des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) des-ede3-cbc(7)}

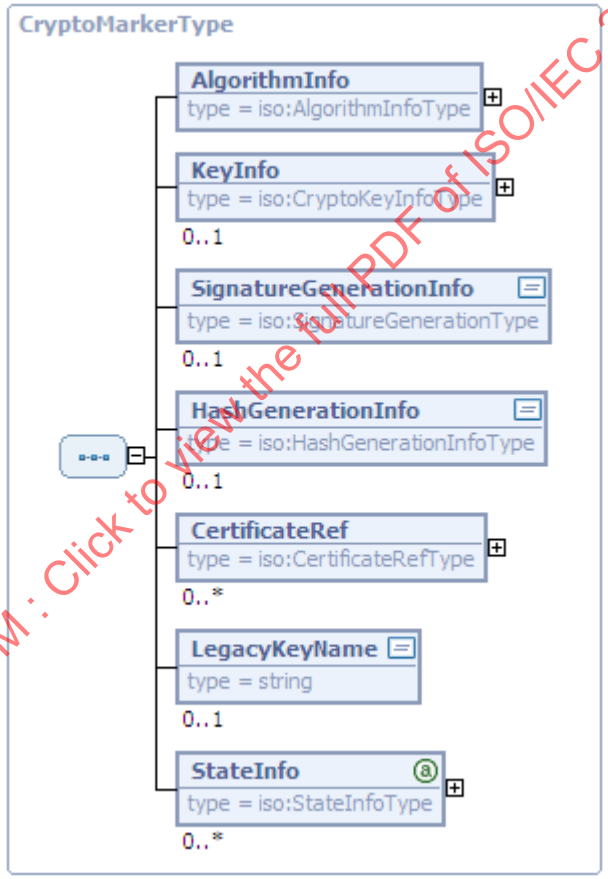
E.4 XML binding for generic digital signature operation

Cryptographic operations can be used independently from specific authentication procedures under this generic protocol. It allows for referencing and using DID for digital signature according EN14890 scheme.

The proposed identifier for this protocol (to be confirmed) is urn:oid:1.3.162.15480.3.0.25 for iso(1) identified-organization (3) CEN (162) CEN 15480 (15480) part3(3 annex-a(0) generic-cryptography (x) with x=25.

This clause provides a generic description of the differential-identities for digital signature operations involved in some authentication protocols as the Asymmetric Internal Authenticate (ISO/IEC 24727-3:2008). Regardless of the authentication protocol, generic cryptographic operations (i.e. Sign, Encipher, Decipher) resorting to differential-identities and executed through the SAL API with XML binding require a comprehensive definition of their attributes as XML Elements. These Elements include the CryptoMarkerType and all card-application content management for differential-identities (e.g. DIDCreate, DIDUpdate, DIDGet, …) through the SAL API. The following XML Elements are provided for this purpose.

E.4.1 Marker



This type specifies the structure of the DID qualifier for this generic protocol.

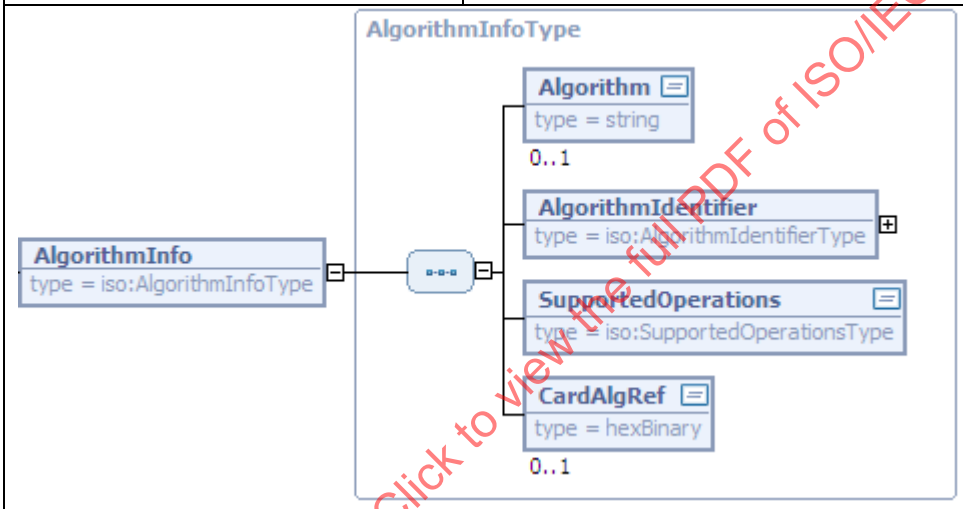
Name	Description
AlgorithmInfo	Contains information about the cryptographic algorithms supported by the DID. See below for details.

KeyInfo	MAY contain information on the DID key material. See below for details.

SignatureGenerationInfo	MAY contain information about the sequence of the smart card commands required for the DID to generate signatures, if the DID can function in this way. The SignatureGenerationType is defined as follows:
	
	`<simpleType name="SignatureGenerationType">`
	`  <list>`
	`  <simpleType>`
	`  <restriction base="token">`
	`  <enumeration value="MSE_RESTORE" />`
	`  <enumeration value="MSE_HASH" />`
	`  <enumeration value="PSO_HASH" />`
	`  <enumeration value="MSE_KEY" />`
	`  <enumeration value="MSE_DS" />`
	`  <enumeration value="MSE_KEY_DS" />`
	`  <enumeration value="PSO_CDS" />`
	`  <enumeration value="INT_AUTH" />`
	`  </restriction>`
	`  </simpleType>`
	`  </list>`
	`</simpleType>`
	
	The definition of this type as a list of the smart card commands relevant for the signature allows precise specification of the commands by means of which a signature can be generated and the order of these commands.

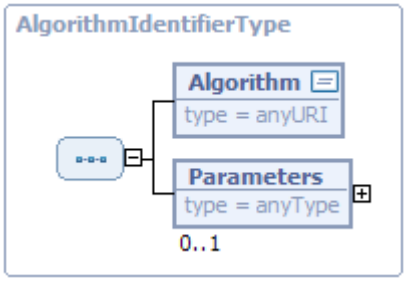
HashGenerationInfo	MAY contain information about the details required for the DID to calculate hash values, if the DID can function in this manner. The HashGenerationInfoType is defined as follows: `<simpleType name="HashGenerationInfoType">`

	<restriction base="string"> <enumeration value="NotOnCard" /> <enumeration value="CompletelyOnCard" /> <enumeration value="LastRoundOnCard" /> </restriction> </simpleType>
CertificateRef	MAY contain a sequence of references to certificates which are stored on the card.
LegacyKeyName	MAY contain a name which can be used to address the DID in a legacy application, e.g. a Microsoft Cryptographic API provider.
StateInfo	MAY contain information about the designated states iif more than one state is defined for the key object. There is one StateInfo element for each possible state and the current state corresponds to the first StateInfo element in the sequence.



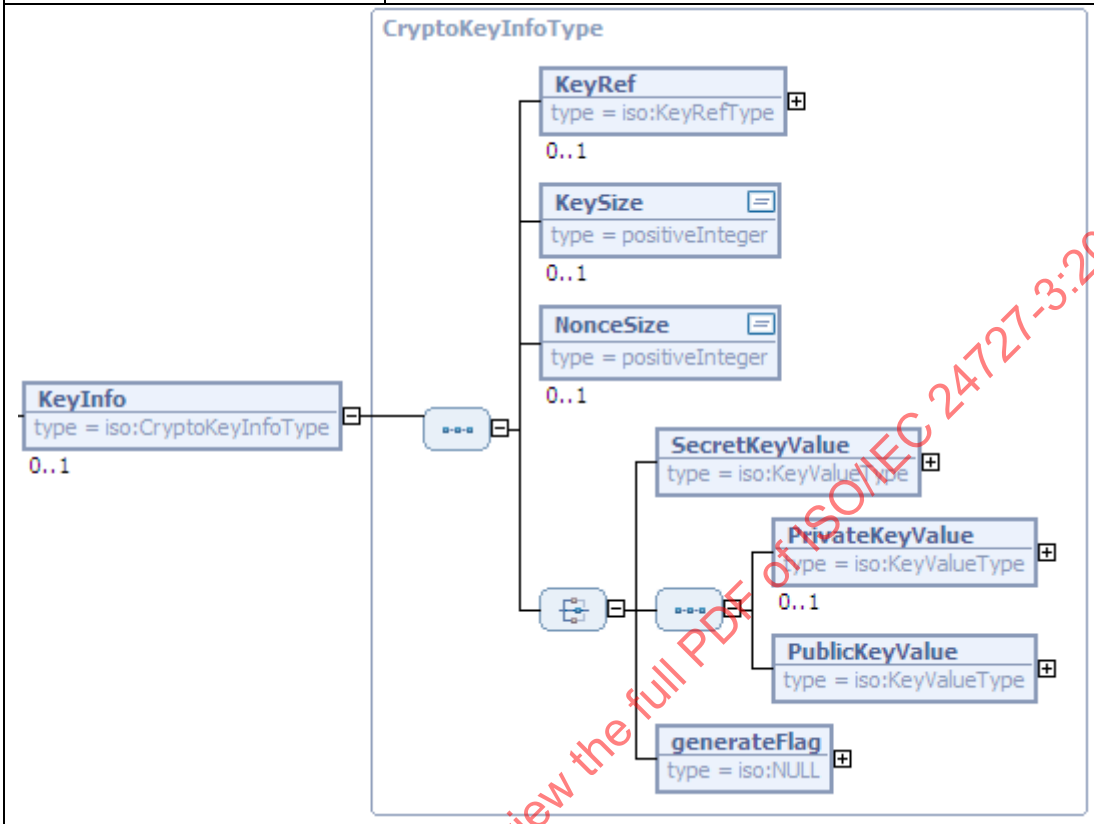
The AlgorithmInfo element is part of the CryptoMarkerType and contains information about the cryptographic algorithm supported by the DID. The AlgorithmInfoType is based on the AlgorithmInfo structure.

Name	Description
Algorithm	MAY contain a textual descriptor for the algorithm. For example, the descriptors from PKCS#11 MAY be used here.

AlgorithmIdentifier	Contains the unambiguous descriptor for the cryptographic algorithm in the form of a URI and, if required, further parameters for the algorithm. AlgorithmIdentifierType is as follows:  The AlgorithmIdentifierType is used for the definition of the CADomainParameterInfo element.

SupportedOperations	Specifies the cryptographic operations for which the DID can be used. The SupportedOperationsType is defined as follows:
	
	`<simpleType name="SupportedOperationsType">`
	`<union memberTypes="iso:BitString">`
	`<simpleType>`
	`<list>`
	`<simpleType>`
	`<restriction base="token">`
	`<enumeration value="Compute-checksum" />`
	`<enumeration value="Compute-signature" />`
	`<enumeration value="Verify-checksum" />`
	`<enumeration value="Verify-signature" />`
	`<enumeration value="Encipher" />`
	`<enumeration value="Decipher" />`
	`<enumeration value="Hash" />`
	`<enumeration value="Derive-key" />`
	`</restriction>`
	`</simpleType>`
	`</list>`
	`</simpleType>`

	</union> </simpleType>
CardAlgRef	MAY contain the card-specific reference for the cryptographic algorithm on the card.



The KeyInfo element is part of the CryptoMarkerType and contains information about the DID key material.

Name	Description
KeyRef	MAY contain the key reference of the DID on the card.
KeySize	MAY contain the relevant bit length of the cryptographic key.
NonceSize	MAY contain the length of the random numbers in bytes which can be requested via GetRandom.
SecretKeyValue	Contains the value of a secret key.
PrivateKeyValue	MAY contain the private key.
PublicKeyValue	Contains the public key of the DID if the generateFlag alternative has not been selected.
generateFlag	Specifies that the key material belonging to the DID is to be generated on the card. If this alternative is not selected, either the

	SecretKeyValue element is present or at least the PublicKeyValue element and, where applicable, (if the DID is to be used for signature generation or decryption) the PrivateKeyValue element as well.

E.4.2 DIDCreate

With DIDCreate use is made of DIDCreationData of the CryptoMarkerType.

E.4.3 DIDUpdate

With DIDUpdate use is made of DIDUpdateData of the CryptoMarkerType.

E.4.4 DIDGet

With DIDGet use is made of DIDDiscoveryData of the CryptoMarkerType.

E.4.5 Encipher

Insofar as this operation is supported by the DID the text transmitted is encrypted with the public or secret key assigned to the DID.

E.4.6 Decipher

Insofar as this operation is supported by the DID the key text transmitted is decrypted with the private or secret key assigned to the DID.

E.4.7 GetRandom

Generates a random number of the required length (NonceSize bytes) and returns it:

```
random = RNG (NonceSize)
```

E.4.8 Hash

If this operation is supported by the DID, a hash value is generated from the message transmitted using the relevant algorithm. In the process, the calculation is either entirely on the card, partly on the card or not on the card.

Hash values for data streams can also be calculated using DIDAuthenticate.

E.4.9 Sign

If this operation is supported by the DID (cf. SupportedOperations element) a signature is generated for the message transmitted using the DID specified. The series of smart card commands specified in the SignatureGenerationInfo element is transmitted in the process.

E.4.10 VerifySignature

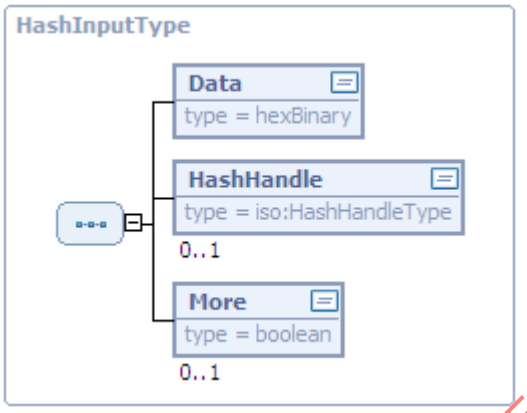
The transmitted signature is then verified on the basis of the public key referenced with DIDName.

E.4.11 VerifyCertificate

The certificate which has been transmitted is checked against the public key of the trustworthy root referenced in `RootCert`.

E.4.12 DIDAuthenticate

The `DIDAuthenticate` function can be used with the Generic Crypto protocol to calculate hash values iteratively for data streams or large volumes of data.

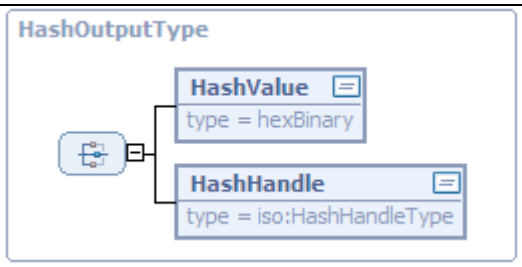


This type specifies the structure of the `HashInputTypes` which is used when invoking `DIDAuthenticate`.

Name	Description
`Data`	Contains the data which are to be supplied for the calculation of the hash value.
`HashHandle`	Is an optional element which is used to be able to link several requests of `DIDAuthenticate`. This might be needed, for example, to calculate hash values for large volumes of data or data streams with several requests from `DIDAuthenticate`. It is necessary to differentiate between the following four cases: — No `HashHandle` and `More=FALSE` (or not available) - the `HashValue` calculated from the data is the only one to be returned. — No `HashHandle` and `More=TRUE` - a new `HashHandle` is returned. — `HashHandle` exists and `More=FALSE` (or not available) - the data transmitted are also included in the calculation of the hash value and the fully calculated `HashValue` is returned. — `HashHandle` exists and `More=TRUE` - the data transmitted are also included in the calculation of the hash value and the transmitted `HashHandle` is returned.
`More`	`More=TRUE` specifies that the hash value is not to be returned yet in this request but that further requests are to follow from `DIDAuthenticate` and therefore a `HashHandle` is to be returned.

	If this element is missing, this is equivalent to the default value `More=FALSE` and, as a result, any existing `HashHandle` will lose its validity and a `HashValue` will be returned.

The `DIDAuthenticateResponse` returns data of the `HashOutputType` in the `DIDAuthenticationData` element:



This type specifies the structure of the `HashOutputType` which is returned in the `DIDAuthenticationData` element of the `DIDAuthenticateResponse`.

Name	Description
`HashValue`	Is the calculated hash value.
`HashHandle`	Is a handle which can be transmitted with the next invocation of `DIDAuthenticate` so that the hash value can be calculated iteratively.

E.4.13 Non-supported functions

The following functions are not supported with this protocol and return a corresponding error message /resultminor/sal#inappropriateProtocolForAction when invoked:

— CardApplicationStartSession

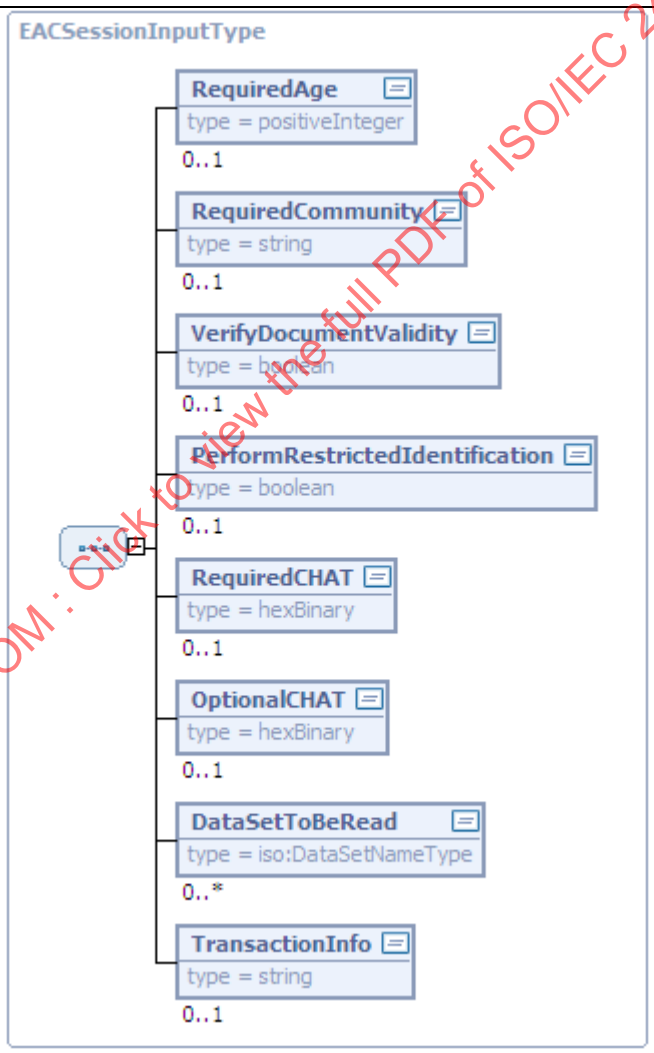
E.5 Modular Extended Access Control Protocol (M-EAC)

This protocol forms the framework for mutual authentication with keys exchanged using the Modular Extended Access Control protocol according to ISO/IEC 24727-3 A.14.

E.5.1 Call and return of CardApplicationStartSession

The protected channel to the card by means of EAC is established by requesting `CardApplicationStartSession` with a corresponding DID for the EAC protocol. In this context the `DIDName` refers to the DID on the ICC with the marker structure defined A.14. The `AuthenticationProtocolData` are of type `EACSessionInputType` explained in more detail below, through which the optional test sequences for age verification, document validity and municipality citizenship MAY be specified and / or the generation of a sector-specific pseudonym MAY be requested.

If required, a differentiation MAY also be made between different eService keys (with different certificates and authorisations) using the `SAMConnectionHandle`. Handles for the keys / certificates, which are currently available to the eService SAL are returned without additional parameters when `CardApplicationPath` is called.

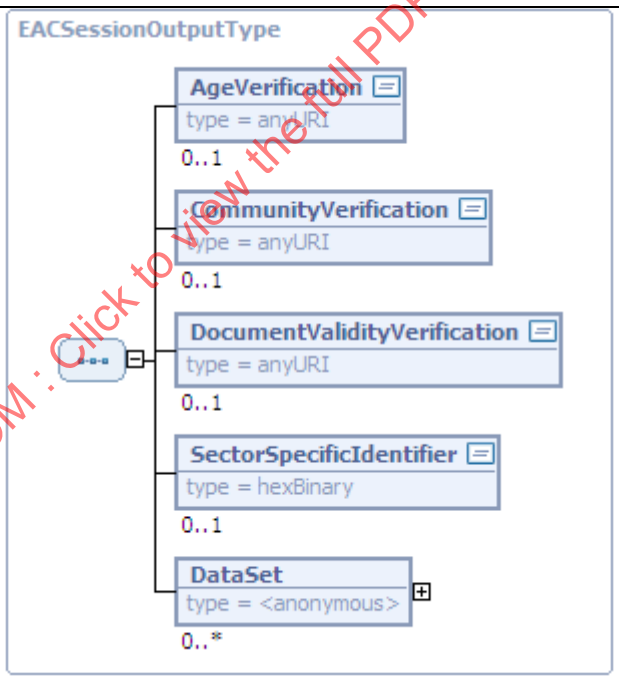


This type specifies the structure of the `AuthenticationProtocolData` when

CardApplicationStartSession is called up with the EAC protocol.	
Name	**Description**
RequiredAge	MAY be used for age verification with a specific minimum age. If this element is missing, the age is not verified. This element is converted by the eService SAL into the format required for the ICC (cf. Authenticated-AuxiliaryData in TADIDAuthInputType). If the age verification process fails, a warning is returned in the AgeVerification element explained below (…/sal/mEAC#AgeVerificationFailedWarning). If the specified DID is provided in EAC Version 1, the warning returned is …/sal/Functionality ByCurrentProtocolVersionNotSupportedWarning.
RequiredCommunity	MAY be used to check whether the citizen is affiliated with a certain municipality. If the community affiliation process fails, a warning is returned in the CommunityVerification element explained below (…/sal/mEAC#CommunityVerificationFailedWarning). If the specified DID is provided for EAC Version 1, the warning returned is …/sal/Functionality ByCurrentProtocolVersionNotSupportedWarning. If this element is missing, the citizenship is not checked.
VerifyDocumentValidity	MAY specify whether the current document validity will be checked. If this element is missing or FALSE, the document validity is not checked. This element is converted by the eService SAL into the format required for the ICC (cf. Authenticated-AuxiliaryData in TADIDAuthInputType). If the document validity check fails, the warning (…/sal/mEAC#DocumentValidityVerificationFailed) is returned. If the specified DID is provided for EAC Version 1, the warning returned is …/sal/Functionality ByCurrentProtocolVersionNotSupportedWarning.
PerformRestricted Identification	MAY specify whether the sector-specific pseudonym is to be calculated once the trustworthy channel has been established between the eService and ICC with the Restricted Identification protocol. If the specified DID is provided for EAC Version 1, the warning returned is …/sal/Functionality ByCurrentProtocolVersionNotSupportedWarning.
RequiredCHAT	If the eService does not want to use the full access rights provided by the CHAT of the addressed certificate or leave it up to the configuration of the eService-SAL, it MAY explicitly specify the required CHAT here. If the user does not allow the

	required rights, the connection establishment fails.
`OptionalCHAT`	In a similar manner the eService MAY specify optional access rights, which are not strictly necessary to establish the connection.
`DataSetToBeRead`	In order to minimise the number of messages which SHOULD be sent via the network, it is possible to send the necessary APDUs for the Restricted Identification protocol together with the APDUs for the data readout in one single `Transmit` request. To allow this, the data groups which are to be read out SHOULD be specified when invoking `CardApplicationStartSession`, with one `DataSetToBeRead` element available for each data set to be read out.
`TransactionInfo`	This element MAY contain transaction-related information, which SHOULD be displayed in the eID-PIN dialogue before the PACE-protocol is performed.

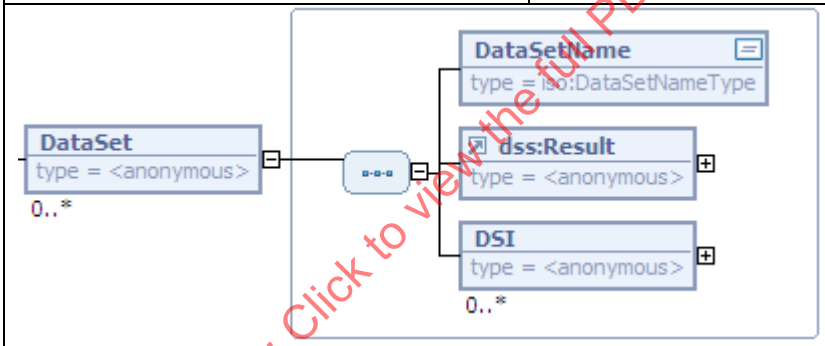
In response to the `CardApplicationStartSession` request, a `CardApplicationStartSessionResponse` is returned with `AuthenticationProtocolData` of the `EACSessionOutputType`.



This type specifies the structure of the `AuthenticationProtocolData` in `CardApplicationStartSessionResponse` with the EAC protocol.

Name	Description
`AgeVerification`	If a `RequiredAge` element was transferred on invocation, the result of the age verification is returned in this element.

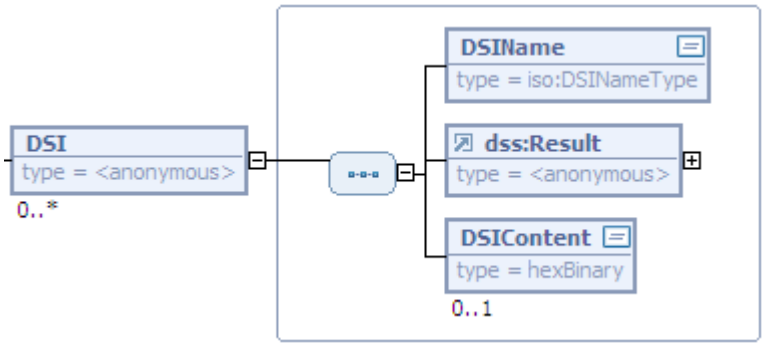
	If the check was successful, the returned URI is …/resultmajor#ok, whereas the error code returned if the age verification failed is …/sal/EAC#AgeVerificationFailedWarning.
CommunityVerification	If a RequiredCommunity element was transferred on invocation, the result of the citizenship check is returned in this element. If the check was successful, the returned URI is …/resultmajor#ok, whereas the error code returned if the citizenship check failed is …/sal/EAC#CommunityVerificationFailedWarning.
DocumentValidityVerification	If the document validity check was requested with the VerifyDocumentValidity element, which is assigned the status True, the result of this check is returned in this element. If the check is successful, the returned URI is …/resultmajor#ok, whereas the error code …/sal/EAC#DocumentValidityVerificationFailed is returned if the document validity check fails.
SectorSpecificIdentifier	If the PerformRestrictedIdentification element was used to request the calculation of the sector-specific pseudonym, this is returned here.
DataSet	In response to each DataSet request either its content or an error message, indicating that readout was unsuccessful is returned. The precise structure of this element is explained in detail below.



For each DataSetToBeRead element in the request, a corresponding DataSet element is returned.

Name	Description
DataSetName	Contains the name of the DataSet.
dss:Result	Contains the result of the request. If the DataSet readout was successful, the URI returned in the ResultMajor element is …/resultmajor#ok. If the process fails, the URI returned in the ResultMajor element is …/resultmajor#error and, in addition, further details are returned in the ResultMinor element as to the cause of the error, distinguishing between the following cases: • /resultminor/sal#unknownDataSetName • /resultminor/sal#securityConditionsNotSatisfied

	• /resultminor/sal#prerequisitesNotSatisfied
DSI	Is available once for each Data Structure for Interoperability contained in the DataSet. See below for details.



The DSI element is part of DataSet.

Name	Description
DSIName	Contains the name of the Data Structure for Interoperability (DSI).
dss:Result	Contains the result of the request. If the DSI readout was successful, the message returned in the ResultMajor element is …/resultmajor#ok . If the process fails, the message returned in the ResultMajor element is …/resultmajor#error and, in addition, further details are returned in the ResultMinor element as to the cause of the error, distinguishing between the following cases: • /resultminor/sal#unknownDSIName • /resultminor/sal#prerequisitesNotSatisfied • /resultminor/sal#securityConditionsNotSatisfied
DSIContent	The content of the DSI is returned here if the process is successful.

E.5.2 Overview of EAC protocol sequence

The sequence between both SAL instances after invocation of CardApplicationStartSession on the eService SAL is shown in the following diagram:

Note: the PACE protocol is defined in Supplementary Access Control – EN 14890-1 .

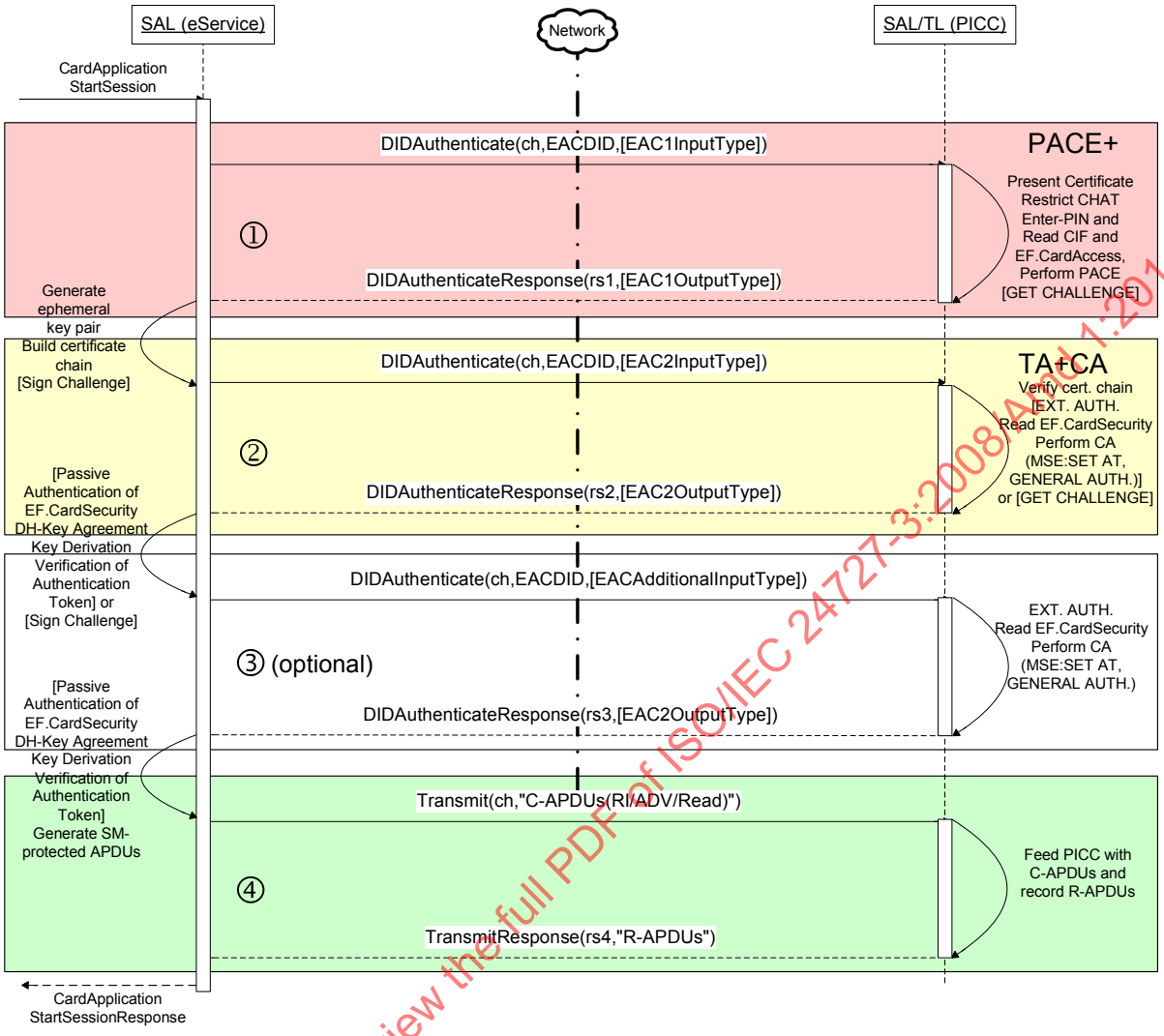


Figure E.1: Message sequence between SAL's after CardApplicationStartSession (M-EAC)

The protected channel between the eService SAL and the ICC is established in the following steps:

1. The eService-SAL invokes `DIDAuthenticate` with the `DIDName` provided for PACE and `AuthenticationProtocolData` of the `EAC1InputType` explained in more detail below. The eService certificate intended for use is transferred in this process or, if necessary, an alternative `CardHolderAuthorizationTemplate` (CHAT), corresponding certificate descriptions if required and the `AuthenticatedAuxiliaryData` prepared for the chip. The contents of the certificate, any existing certificate descriptions (`CertificateDescription`) and the `TransactionInfo`-element are duly displayed to the user. The user is also given the opportunity to limit the effective eService access rights before entering the password and thereby signalling consent to the access. The file `EF.CardAccess` is read out and, following the PACE protocol process, the challenge for the Terminal Authentication is requested by the chip if the ICC is able to check the terminal certificate chain between the challenge request and the signature verification. If this is not possible, the Challenge element is missing from the `EAC1OutputType` returned and an additional message is required (see step 3). The data described are returned in the `AuthenticationProtocolData` of type `EAC1OutputType`, which is explained in more detail below. If this process is successful, in addition to the Challenge which may be present, the ASN.1-coded `SecurityInfo` structure from `EF.CardAccess`, and the `CertificateHolderAuthorizationTemplate` (CHAT), which is limited by the user, there will be up

to two CertificationAuthorityReference elements which specify the root keys that are available for the certificate verification on the ICC. The SecurityInfo structure from the EF.CardAccess file notably contains the domain parameters which are used in the next step to generate a fresh key pair. The eService SAL selects a certificate chain appropriate for the currently connected ICC using the CertificationAuthorityReference elements and sends it to the ICC SAL in the next step.
If the password entry fails, this SHOULD be duly displayed to the user on the client system so that the user MAY re-enter the password and, if necessary, reset the retry counter or cancel the entry.

In the process, the user SHOULD also be informed of the remaining number of attempts allowed to enter the password and, where applicable, be given the opportunity to reset the retry counter. If the password entry process is cancelled after an incorrect entry and the number of remaining attempts to enter the password correctly is therefore known in the ICC SAL, this SHOULD be returned in the RetryCounter element.

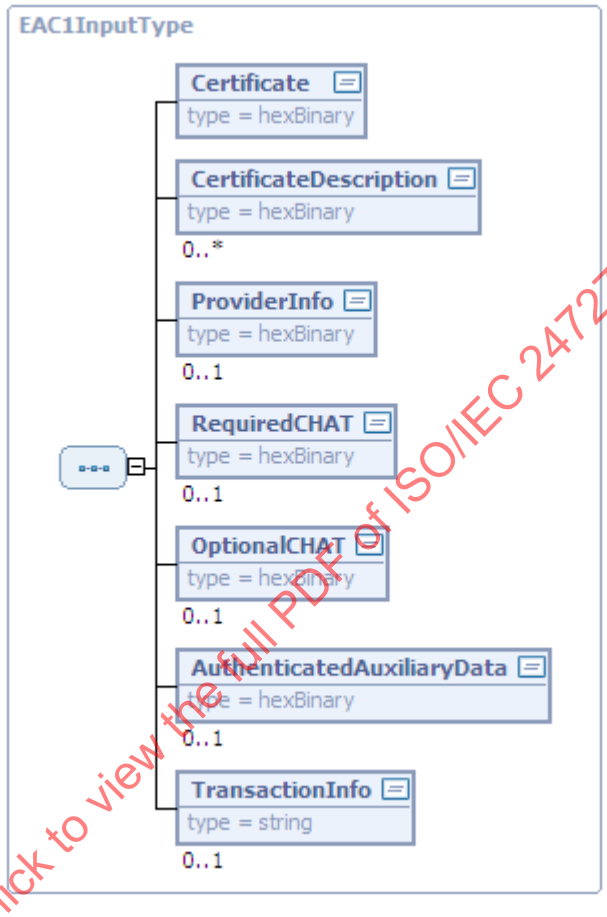
2. Using the Chip Authentication domain parameters (see SecurityInfo structure above), the eService SAL generates a fresh key pair in the next step, forms an appropriate chain of additionally required certificates and finally, where required, signs the Challenge which has been transmitted. The eService SAL then invokes DIDAuthenticate for the CADID and relays AuthenticationProtocolData of type EAC2InputType, which is described in more detail below, to the ICC SAL. In addition to the certificate chain applicable to the ICC, this element notably contains the newly generated public key EphemeralPublicKey and, where generated, the signature (Signature). The certificate chain is verified by the ICC. Moreover, either the signature is checked by means of EXTERNAL AUTHENTICATE or the Challenge is requested — given that the Challenge could not be requested before the certificate check. If it is already possible at this stage, the file EF.CardSecurity is read out and finally the Chip Authentication is executed by invoking MSE:SET AT and GENERAL AUTHENTICATE. Finally, the result of these actions is returned to the eService SAL in AuthenticationProtocolData of type EAC2OutputType, which is explained in more detail below. This includes either the Challenge-element or the content of the file EF.CardSecurity (EFCardSecurity), the authentication token (AuthenticationToken) and the random number required to check the authentication token (Nonce).

3. (Optional) If the Challenge has not already been returned with the first message, thus preventing the terminal signature from being generated until this time, an additional invocation of DIDAuthenticate with AuthenticationProtocolData of type EACAdditionalMessageType is required to transmit the terminal signature to the ICC where it is checked by invoking EXTERNAL AUTHENTICATE. In this case the file EF.CardSecurity is read out first and finally the Chip Authentication is executed by invoking MSE:SET AT and GENERAL AUTHENTICATE. The result of these actions is then returned to the eService SAL in AuthenticationProtocolData of type EAC2OutputType, which is explained in more detail below. This comprises the content of the file EF.CardSecurity (EFCardSecurity), the authentication token (AuthenticationToken) and the random number required to check the authentication token (Nonce).

4. If the signature extracted from EFCardSecurity (Passive Authentication) and the authentication token generated in the Chip Authentication process are verified, the eService SAL MAY then communicate with the ICC via APDUs protected by secure messaging in order to — according to the information requested by means of CardApplicationStartSession — request the generation of the sector-specific pseudonym, perform additional checks or read out certain data stored on the ICC. The APDUs required for this MAY be calculated in advance by the eService SAL and transferred as a batch using the Transmit function via the network to the IFD-Layer on the side of the ICC. The IFD-Layer on the side of the ICC in turn sends the APDUs prepared by the eService SAL to the ICC in sequence and logs the respective response APDUs, which are ultimately sent back to the eService SAL as a collective batch in the TransmitResponse.

E.5.3 Phase 1 - Extended PACE protocol

The "Extended PACE protocol" is realized by one single request of `DIDAuthenticate` with `AuthenticationProtocolData` of type `EAC1InputType`:

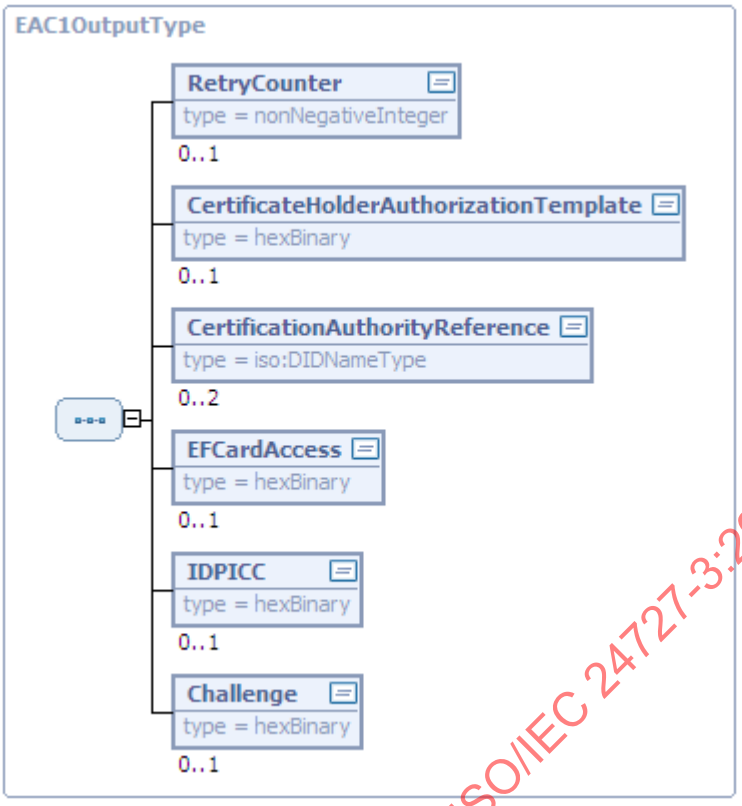


The `EAC1InputType` specifies the structure of the `DIDAuthenticationDataType` for the "Extended PACE protocol" on invoking `DIDAuthenticate` within the EAC protocol (EAC).

Name	Description
`Certificate`	Contains the eService certificate. The relevant contents of the certificate (at least the Certification Authority Reference, Certificate Holder Reference, Certificate Holder Authorization Template (CHAT) including user-friendly display of rights, Certificate Effective Date, Certificate Expiration Date and, where present, Certificate Extensions) SHOULD be duly displayed to the user before the password is entered. The user SHOULD also be given the opportunity at this point to impose further restrictions on the CHAT and therefore on the effective access rights to the terminal or to cancel the operation without entering a password.

CertificateDescription	While the present schema allows an arbitrary number of CertificateDescription-elements there SHOULD exactly be one such element and the citizen client SHOULD display the content of this element in a suitable manner before capturing the PIN and performing the PACE-protocol. The CertificateDescription-element contains a data object commCertificates, which contains a set of hash values of admissible X.509 certificates, which can be used for the establishment of TLS-channels over which the EAC-protocol is performed. The citizen client SHOULD verify the authenticity of the X.509-certificates used during the different TLS-handshakes by inspecting this commCertificates data object.
ProviderInfo	While the present schema allows that there is a ProviderInfo-element, this element is deprecated and is ignored if present.
RequiredCHAT	Specifies the data, which are required by the eService. If the full rights specified in the certificate are not supposed to be used, a CHAT already restricted by the eService MAY be transferred to the user. It SHOULD be possible, applying the principle of data-thrift, to configure the eService SAL to dictate which CHAT is transferred with which certificates and in which cases.
OptionalCHAT	Specifies the data, which are requested by the eService, but which transmission may be surpressed by the user.
AuthenticatedAuxiliaryData	MAY contain additional data which are used to check the validity of the card, verify the age or check municipality citizenship. For each piece of data transmitted for additional verification after successful Terminal Authentication, a Verify command is requested (e.g. with OID 0.4.0.127.0.7.3.1.4.1 (id-auxiliaryData-1) for age verification, with 0.4.0.127.0.7.3.1.4.2 (id-auxiliaryData-2) for the document validity check or with 0.4.0.127.0.7.3.1.4.3 (id-auxiliaryData-3) to check municipality citizenship.
TransactionInfo	This element MAY contain transaction-related information, which SHOULD be displayed in the eID-PIN dialogue before the PACE-protocol is performed.

A DIDAuthenticateResponse element with AuthenticationProtocolData of type EAC1OutputType is returned in response to this request:



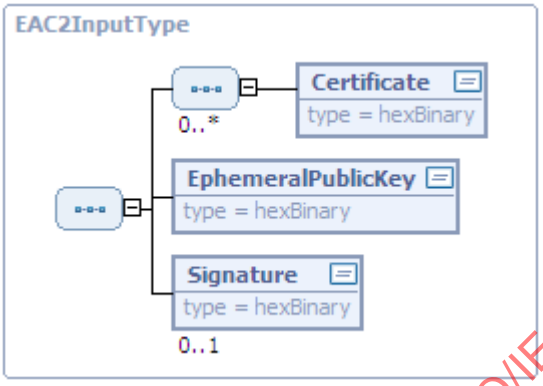
This type specifies the structure of the `DIDAuthenticationDataType` for the PACE protocol when `DIDAuthenticate` is returned.

Name	Description
`RetryCounter`	If the user verification process failed, this element contains the current value of the `RetryCounter`.
`CertificateHolder AuthorizationTemplate`	If the user has imposed further restrictions on the CHAT transmitted by the eService, such that the actual access rights do not correspond with the access rights which might potentially ensue from the certificate, the eService SAL SHOULD be informed of the CHAT restricted by the user in this manner.
`CertificationAuthority Reference`	Contains up to two references to the certification authority which MAY be used in the context of the Terminal Authentication to verify the terminal certificate. If two references are returned, the first reference is the more current of the two.
`EFCardAccess`	If no errors have occurred, the ASN.1-coded `SecurityInfos` from the `EF.CardAccess` file are returned at this point.
`IDICC`	Contains the "card identity" ID_{ICC}. This involves the Doc# from the MRZ in case of BAC or the compressed ephemeral public key of the ICC in case of PACE.
`Challenge`	MAY contain the random number generated by the ICC, $r_{ICC,TA}$, which is signed by the eService SAL during the

	Terminal Authentication, if and only if the ICC allows the verification of certificates between requesting the Challenge and checking the signature.

E.5.4 Phase 2 - Combination of Terminal and Chip Authentication

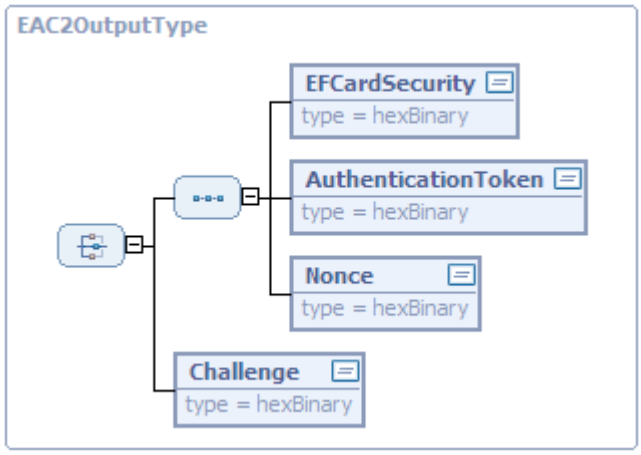
The next step involves — if the Challenge was produced in the first phase — a combination of Terminal and Chip Authentication, which is initiated by an invocation of `DIDAuthenticate` with `AuthenticationProtocolData` of type `EAC2InputType`.



This type specifies the structure of the `EAC2InputType` which is used in the EAC protocol on the second request of `DIDAuthenticate`.

Name	Description
`Certificate`	The `Certificate` element MAY occur any number of times and contains a certificate in each case so that, together with the eService certificate already transmitted, the resulting overall chain is one which MAY be verified by the ICC.
`EphemeralPublicKey`	Contains the public key of the key pair newly generated by the eService.
`Signature`	Contains, where applicable, the signature generated by the eService SAL during Terminal Authentication.

`AuthenticationProtocolData` of type `EAC2OutputType` are returned in the subsequent `DIDAuthenticateResponse`.

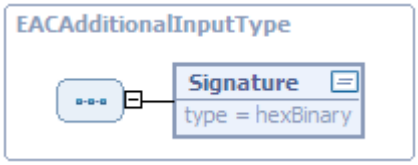


This type specifies the structure of the `EAC2OutputType` which is used in the EAC protocol on the second request of `DIDAuthenticate`. If the `Challenge` has already been returned in the previous message (cf. `EAC1OutputType`), the elements `EFCardSecurity`, `AuthenticationToken` and `Nonce` are returned. Otherwise the `Challenge` element is returned at this point.

Name	Description
`EFCardSecurity`	Contains a `SignedData` structure in accordance with RFC 3852 which contains the full `SecurityInfo` structure in the content data (`EncapsulatedContentInfo`). This signature is checked by the eService SAL during Passive Authentication.
`AuthenticationToken`	Contains the authentication token (T_{ICC}).
`Nonce`	Contains the random number ($r_{ICC,CA}$).
`Challenge`	Contains the Challenge from the ICC. If this element is returned at this point, the additional message SHOULD be sent with the `Signature` element in the next step.

E.5.5 Phase 3 - Optional additional message with signature forwarded separately

If the Challenge was produced in Phase 2 instead of Phase 1, an additional invocation of `DIDAuthenticate` with `AuthenticationProtocolData` of type `EACAdditionalInputType` occurs in this optional phase.



This type specifies the structure of the `EACAdditionalInputType` which is used in the optional additional message that is required if the `Challenge` was not included in the first phase.

Name	Description

Signature	Contains the signature generated by the eService SAL during Terminal Authentication.

AuthenticationProtocolData of type EAC2OutputType are returned in the subsequent DIDAuthenticateResponse and in this case the elements EFCardSecurity, AuthenticationToken and Nonce SHOULD be included.

E.5.6 Phase 4 - Secure messaging with APDU batches

If the Passive Authentication process (verification of the signature from EFCardSecurity) and the verification of the authentication token generated during the Chip Authentication process are successful, the eService SAL MAY now communicate with the ICC with APDUs protected with secure messaging. The APDUs required for this MAY be calculated in advance by the eService SAL and transferred as a batch using the Transmit function via the network to the IFD-Layer on the ICC. The IFD-Layer on the ICC sides sends the APDUs prepared by the eService SAL to the ICC in sequence and logs the respective response APDUs which are ultimately sent back to the eService SAL as a collective batch in TransmitResponse.

E.5.7 DIDCreate, DIDUpdate and DIDGet

The requests of DIDCreate, DIDUpdate and DIDGet each use an element of the EACMarkerType as the input parameter (with DIDCreate and DIDUpdate) or the output parameter (with DIDGet), respectively.

E.5.8 Non-supported functions

The following functions are not supported with this protocol and return a corresponding error message /resultminor/sal#inappropriateProtocolForAction when invoked:

- DIDAuthenticate
- Encipher
- Decipher
- GetRandom
- Hash
- Sign
- VerifySignature
- VerifyCertificate

Annex F
(normative)

XML Service Access Layer Interface

F.1 XML-based Service Access Layer Interface

The Web Service Binding for the SAL-API is defined in the following clauses:

— **F.1.1 Basic types (ISOCommon.xsd)** – contains the basic types, which are imported in other schema files such as ISO24727-3.xsd and ISOIFD.xsd for example.

— **F.2 XML-Schema definitions for Service Access Layer functions (ISO24727-3.xsd)** - contains XML-elements for each SAL-API request and response.

— **F.3 WSDL definitions for Service Access Layer functions (ISO24727-3.wsdl)** – includes ISO24727-3.xsd and specifies the SOAP-based Web Service binding for the SAL-API.

— **F.4 XML-Schema definitions for selected authentication protocols (ISO24727-Protocols.xsd)** – contains the XML-specifications for the selected authentication protocols defined in this document.

— **F.5 WSDL definitions for connection establishment (ISO24727-Protocols.wsdl)** – contains web service specifications for TC_API_Open and StartPAOS messages, which are used for the connection establishment.

F.1.1 Basic types (ISOCommon.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <import namespace="urn:oasis:names:tc:dss:1.0:core:schema"
        schemaLocation="http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-schema-
v1.0-os.xsd">
    </import>

    <!-- Definition of Basic Types -->

    <simpleType name="SlotHandleType">
        <restriction base="hexBinary"></restriction>
    </simpleType>

    <complexType name="ChannelHandleType">
        <sequence>
            <element name="ProtocolTerminationPoint" type="anyURI"
                maxOccurs="1" minOccurs="0">
            </element>
            <element name="SessionIdentifier" type="string"
                maxOccurs="1" minOccurs="0">
            </element>
            <element name="Binding" type="anyURI" maxOccurs="1"
                minOccurs="0" default="http://schemas.xmlsoap.org/soap/http">
            </element>
```

```
            <element name="PathSecurity" type="iso:PathSecurityType"
                maxOccurs="1" minOccurs="0">
            </element>
        </sequence>
    </complexType>

    <complexType name="PathSecurityType">
        <sequence>
            <element name="Protocol" type="anyURI"></element>
            <element name="Parameters" type="anyType" maxOccurs="1"
                minOccurs="0">
            </element>
        </sequence>
    </complexType>

    <simpleType name="ContextHandleType">
        <restriction base="hexBinary"></restriction>
    </simpleType>




    <!-- Define Response Type -->


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

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

</schema>
```

F.2 XML-Schema definitions for Service Access Layer functions (ISO24727-3.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

    <include schemaLocation="ISOCommon.xsd" />


    <include schemaLocation="ISOIFD.xsd"></include>


    <!-- Define names of card application services -->


    <!-- Define actions within card application services -->


    <complexType name="ActionNameType">
        <choice>
            <element name="APIAccessEntryPoint"
                type="iso:APIAccessEntryPointName" />
            <element name="ConnectionServiceAction"
                type="iso:ConnectionServiceActionName" />
            <element name="CardApplicationServiceAction"
                type="iso:CardApplicationServiceActionName" />
            <element name="NamedDataServiceAction"
                type="iso:NamedDataServiceActionName" />
            <element name="CryptographicServiceAction"
                type="iso:CryptographicServiceActionName" />
            <element name="DifferentialIdentityServiceAction"
                type="iso:DifferentialIdentityServiceActionName" />
            <element name="AuthorizationServiceAction"
                type="iso:AuthorizationServiceActionName" />
            <element name="LoadedAction" type="string" />
        </choice>
    </complexType>


    <simpleType name="APIAccessEntryPointName">
        <restriction base="string">
            <enumeration value="Initialize" />
            <enumeration value="Terminate" />
            <enumeration value="CardApplicationPath" />
        </restriction>
    </simpleType>


    <simpleType name="ConnectionServiceActionName">
        <restriction base="string">
            <enumeration value="CardApplicationConnect" />
            <enumeration value="CardApplicationDisconnect" />
            <enumeration value="CardApplicationStartSession" />
            <enumeration value="CardApplicationEndSession" />
        </restriction>
    </simpleType>
```

```
<simpleType name="CardApplicationServiceActionName">
    <restriction base="string">
        <enumeration value="CardApplicationList" />
        <enumeration value="CardApplicationCreate" />
        <enumeration value="CardApplicationDelete" />
        <enumeration value="CardApplicationServiceList" />
        <enumeration value="CardApplicationServiceCreate" />
        <enumeration value="CardApplicationServiceLoad" />
        <enumeration value="CardApplicationServiceDelete" />
        <enumeration value="CardApplicationServiceDescribe" />
        <enumeration value="ExecuteAction" />
    </restriction>
</simpleType>


<simpleType name="NamedDataServiceActionName">
    <restriction base="string">
        <enumeration value="DataSetList" />
        <enumeration value="DataSetCreate" />
        <enumeration value="DataSetSelect" />
        <enumeration value="DataSetDelete" />
        <enumeration value="DSIList" />
        <enumeration value="DSICreate" />
        <enumeration value="DSIDelete" />
        <enumeration value="DSIRead" />
        <enumeration value="DSIWrite" />
    </restriction>
</simpleType>


<simpleType name="CryptographicServiceActionName">
    <restriction base="string">
        <enumeration value="Encipher" />
        <enumeration value="Decipher" />
        <enumeration value="GetRandom" />
        <enumeration value="Hash" />
        <enumeration value="Sign" />
        <enumeration value="VerifySignature" />
        <enumeration value="VerifyCertificate" />
    </restriction>
</simpleType>


<simpleType name="DifferentialIdentityServiceActionName">
    <restriction base="string">
        <enumeration value="DIDList" />
        <enumeration value="DIDCreate" />
        <enumeration value="DIDGet" />
        <enumeration value="DIDUpdate" />
        <enumeration value="DIDDelete" />
        <enumeration value="DIDAuthenticate" />
    </restriction>
</simpleType>


<simpleType name="AuthorizationServiceActionName">
    <restriction base="string">
        <enumeration value="ACLList" />
        <enumeration value="ACLModify" />
    </restriction>
```

```
</simpleType>


<!-- Define other basic types -->


<complexType name="AccessControlListType">
    <sequence minOccurs="1" maxOccurs="1">
        <element name="AccessRule" type="iso:AccessRuleType"
            maxOccurs="unbounded" minOccurs="0" />
    </sequence>
</complexType>

<complexType name="AccessRuleType">
    <sequence>
        <element name="CardApplicationServiceName" type="string" />
        <element name="Action" type="iso:ActionNameType"></element>
        <element name="SecurityCondition"
            type="iso:SecurityConditionType" />
    </sequence>
</complexType>

<simpleType name="ApplicationIdentifierType">
    <restriction base="hexBinary"></restriction>
</simpleType>

<simpleType name="CardApplicationServiceLoadPackageType">
    <restriction base="hexBinary"></restriction>
</simpleType>

<complexType name="CardApplicationPathType">
    <sequence>
        <element name="ChannelHandle" type="iso:ChannelHandleType"
            maxOccurs="1" minOccurs="0" />
        <element name="ContextHandle" type="iso:ContextHandleType"
            maxOccurs="1" minOccurs="0" />
        <element name="IFDName" maxOccurs="1" minOccurs="0"
            type="string">
        </element>
        <element name="SlotIndex" type="nonNegativeInteger"
            maxOccurs="1" minOccurs="0">
        </element>
        <element name="CardApplication" maxOccurs="1" minOccurs="0"
            type="iso:ApplicationIdentifierType">
        </element>
    </sequence>
</complexType>

<complexType name="CardApplicationServiceDescriptionType">
    <choice>
        <element name="ServiceDescriptionText" type="string" />
        <element name="ServiceDescriptionURL" type="anyURI" />
    </choice>
</complexType>

<complexType name="ConnectionHandleType">
    <complexContent>
        <extension base="iso:CardApplicationPathType">
            <sequence>
                <element name="SlotHandle" type="iso:SlotHandleType"
```

```
                maxOccurs="1" minOccurs="0">
            </element>
            <element name="RecognitionInfo" maxOccurs="1"
                minOccurs="0">
                <complexType>
                    <sequence>
                        <element name="CardType" type="anyURI"
                            maxOccurs="1" minOccurs="0">
                        </element>
                        <element name="CardIdentifier"
                            type="hexBinary" maxOccurs="1" minOccurs="0">
                        </element>
                        <element name="CaptureTime"
                            type="dateTime" maxOccurs="1" minOccurs="0">
                        </element>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </extension>
</complexContent>
</complexType>


<complexType name="DataSetNameListType">
    <sequence>
        <element name="DataSetName" type="iso:DataSetNameType"
            maxOccurs="unbounded" minOccurs="0">
        </element>
    </sequence>
</complexType>

<simpleType name="DataSetNameType">
    <restriction base="iso:NameType"></restriction>
</simpleType>


<complexType name="DIDAbstractMarkerType" abstract="true">
    <complexContent>
        <extension base="anyType">
            <attribute name="Protocol" type="anyURI" use="required" />
        </extension>
    </complexContent>
</complexType>


<complexType name="DIDAuthenticationDataType" abstract="true">
    <complexContent>
        <extension base="anyType">
            <attribute name="Protocol" type="anyURI" use="required" />
        </extension>
    </complexContent>
</complexType>

<complexType name="DIDAuthenticationStateType">
    <sequence>
        <element name="DIDName" type="iso:NameType" />
        <element name="DIDScope" type="iso:DIDScopeType"
            maxOccurs="1" minOccurs="0" />
        <element name="DIDState" type="boolean"></element>
```

```
        <element name="DIDStateQualifier" type="hexBinary"
            maxOccurs="1" minOccurs="0">
        </element>
    </sequence>
</complexType>

<simpleType name="DIDNameType">
    <restriction base="iso:NameType"></restriction>
</simpleType>

<complexType name="DIDNameListType">
    <sequence>
        <element name="DIDName" type="iso:DIDNameType"
            maxOccurs="unbounded" minOccurs="0">
        </element>
    </sequence>
</complexType>


<complexType name="DIDQualifierType">
    <choice>
        <element name="ApplicationIdentifier"
            type="iso:ApplicationIdentifierType" />
        <element name="ObjectIdentifier" type="anyURI" />
        <element name="ApplicationFunction" type="iso:BitString" />
    </choice>
</complexType>


<complexType name="DIDStructureType">
    <sequence>
        <element name="DIDName" type="iso:DIDNameType" />
        <element name="DIDScope" type="iso:DIDScopeType" />
        <element name="Authenticated" type="boolean" />
        <element name="DIDMarker" type="iso:DIDAbstractMarkerType" />
        <element name="DIDQualifier" type="iso:DIDQualifierType"
            minOccurs="0" maxOccurs="1" />
    </sequence>
</complexType>


<simpleType name="DIDScopeType">
    <restriction base="string">
        <enumeration value="local" />
        <enumeration value="global" />
    </restriction>
</simpleType>


<complexType name="DIDUpdateDataType" abstract="true">
    <complexContent>
        <extension base="anyType">
            <attribute name="Protocol" type="anyURI" use="required" />
        </extension>
    </complexContent>
</complexType>

<simpleType name="DSINameType">
    <restriction base="iso:NameType"></restriction>
```

```
</simpleType>

<complexType name="DSINameListType">
   <sequence>
      <element name="DSIName" type="iso:DSINameType"
         maxOccurs="unbounded" minOccurs="0">
      </element>
   </sequence>
</complexType>


<simpleType name="ExecuteActionConfirmationType">
   <restriction base="hexBinary"></restriction>
</simpleType>


<simpleType name="ExecuteActionRequestType">
   <restriction base="hexBinary"></restriction>
</simpleType>


<simpleType name="FalseType">
   <restriction base="boolean">
      <pattern value='false' />
   </restriction>
</simpleType>

<simpleType name="NameType">
   <restriction base="string">
      <minLength value="1" />
      <maxLength value="255" />
   </restriction>
</simpleType>

<complexType name="NULL" final="#all" />


<complexType name="PathType">
   <sequence>
      <choice>
         <element name="efIdOrPath" type="hexBinary" />
         <element name="TagRef">
            <complexType>
               <sequence>
                  <element name="tag" type="hexBinary" />
                  <element name="efIdOrPath" type="hexBinary"
                     maxOccurs="1" minOccurs="0" />

               </sequence>
            </complexType>
         </element>
         <element name="AppFileRef">
            <complexType>
               <sequence>
                  <element name="aid" type="hexBinary" />
                  <element name="efIDOrPath" type="hexBinary" />

               </sequence>
            </complexType>
         </element>
```

```
            <element name="AppTagRef">
                <complexType>
                    <sequence>
                        <element name="aid" type="hexBinary" />
                        <element name="tag" type="string" />
                        <element name="efIdOrPath" type="hexBinary"
                            maxOccurs="1" minOccurs="0" />

                    </sequence>
                </complexType>
            </element>
        </choice>
        <element name="Index" type="hexBinary" maxOccurs="1"
            minOccurs="0" />
        <element name="Length" type="hexBinary" maxOccurs="1"
            minOccurs="0" />
    </sequence>
</complexType>


<complexType name="SecurityConditionType">
    <choice>
        <element name="DIDAuthentication"
            type="iso:DIDAuthenticationStateType">
        </element>
        <element name="always" type="iso:TrueType" />
        <element name="never" type="iso:FalseType" />
        <element name="not" type="iso:SecurityConditionType" />
        <element name="and">
            <complexType>
                <sequence minOccurs="1" maxOccurs="255">
                    <element name="SecurityCondition"
                        type="iso:SecurityConditionType" />
                </sequence>
            </complexType>
        </element>
        <element name="or">
            <complexType>
                <sequence minOccurs="1" maxOccurs="255">
                    <element name="SecurityCondition"
                        type="iso:SecurityConditionType" />
                </sequence>
            </complexType>
        </element>
    </choice>
</complexType>


<complexType name="TargetNameType">
    <choice>
        <element name="DataSetName" type="iso:DataSetNameType"></element>
        <element name="DIDName" type="iso:DIDNameType"></element>
        <element name="CardApplicationName"
            type="iso:ApplicationIdentifierType">
        </element>
    </choice>
</complexType>


<simpleType name="TrueType">
```

```
        <restriction base="boolean">
            <pattern value='true' />
        </restriction>
    </simpleType>


    <!-- ============================== -->
    <!-- Card-application-service access -->
    <!-- ============================== -->
    <!--            Section 6           -->
    <!-- ============================== -->

    <!-- 6.1 Initialize -->

    <element name="Initialize" type="iso:RequestType"></element>

    <element name="InitializeResponse" type="iso:ResponseType" />

    <!-- 6.2 Terminate -->

    <element name="Terminate" type="iso:RequestType"></element>

    <element name="TerminateResponse" type="iso:ResponseType" />


    <!-- 6.3 CardApplicationPath -->


    <element name="CardApplicationPath">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="CardAppPathRequest"
                            type="iso:CardApplicationPathType" maxOccurs="1"
minOccurs="1" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="CardApplicationPathResponse">
        <complexType>
            <complexContent>
                <extension base="iso:ResponseType">
                    <sequence maxOccurs="1" minOccurs="1">
                        <element name="CardAppPathResultSet">
                            <complexType>
                                <sequence maxOccurs="unbounded"
                                    minOccurs="0">
                                    <element
                                        name="CardApplicationPathResult"
                                        type="iso:CardApplicationPathType">
                                    </element>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </extension>
```

```
            </complexContent>
        </complexType>
    </element>

    <!-- ================== -->
    <!-- Connection Service -->
    <!-- ================== -->
    <!--     Section 7      -->
    <!-- ================== -->

    <!-- 7.2 CardApplicationConnect -->

    <element name="CardApplicationConnect">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="CardApplicationPath"
                            type="iso:CardApplicationPathType" maxOccurs="1"
minOccurs="1" />
                        <element name="Output" type="iso:OutputInfoType"
                            maxOccurs="1" minOccurs="0">
                        </element>
                        <element name="ExclusiveUse" type="boolean"
                            maxOccurs="1" minOccurs="0" default="false" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="CardApplicationConnectResponse">
        <complexType>
            <complexContent>
                <extension base="iso:ResponseType">
                    <sequence maxOccurs="1" minOccurs="0">
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType">
                        </element>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <!-- 7.3 CardApplicationDisconnect -->

    <element name="CardApplicationDisconnect">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                        <element name="Action" type="iso:ActionType"
                            maxOccurs="1" minOccurs="0">
                        </element>
                    </sequence>
                </extension>
            </complexContent>
```

```
        </complexType>
    </element>

    <element name="CardApplicationDisconnectResponse"
        type="iso:ResponseType" />

    <!-- 7.4 CardApplicationStartSession -->

    <element name="CardApplicationStartSession">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                        <element name="DIDScope" type="iso:DIDScopeType"
                            maxOccurs="1" minOccurs="0" />
                        <element name="DIDName" type="iso:NameType" />
                        <element name="AuthenticationProtocolData"
                            type="iso:DIDAuthenticationDataType" />
                        <element name="SAMConnectionHandle"
                            type="iso:ConnectionHandleType" maxOccurs="1"
minOccurs="0" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="CardApplicationStartSessionResponse">
        <complexType>
            <complexContent>
                <extension base="iso:ResponseType">
                    <sequence>
                        <element name="AuthenticationProtocolData"
                            type="iso:DIDAuthenticationDataType">
                        </element>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <!-- 7.5 CardApplicationEndSession -->

    <element name="CardApplicationEndSession">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="CardApplicationEndSessionResponse"
        type="iso:ResponseType" />
```

```
    <!-- ======================== -->
    <!-- Card-application service -->
    <!-- ======================== -->
    <!--          Section 8       -->
    <!-- ======================== -->

    <!-- 8.2 CardApplicationList -->


    <element name="CardApplicationList">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="CardApplicationListResponse">
        <complexType>
            <complexContent>
                <extension base="iso:ResponseType">
                    <sequence maxOccurs="1" minOccurs="1">
                        <element name="CardApplicationNameList">
                            <complexType>
                                <sequence maxOccurs="unbounded"
                                    minOccurs="1">
                                    <element name="CardApplicationName"
                                    type="iso:ApplicationIdentifierType"
maxOccurs="1"
                                        minOccurs="1">
                                    </element>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <!-- 8.3 CardApplicationCreate -->

    <element name="CardApplicationCreate">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                        <element name="CardApplicationName"
                            type="iso:ApplicationIdentifierType" />
                        <element name="CardApplicationACL"
                            type="iso:AccessControlListType">
                        </element>
```

```
            </sequence>
        </extension>
    </complexContent>
</complexType>
</element>

<element name="CardApplicationCreateResponse"
    type="iso:ResponseType" />

<!-- 8.4 CardApplicationDelete -->

<element name="CardApplicationDelete">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="CardApplicationName"
                        type="iso:ApplicationIdentifierType">
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="CardApplicationDeleteResponse"
    type="iso:ResponseType" />

<!-- 8.5 CardApplicationServiceList -->

<element name="CardApplicationServiceList">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="CardApplicationServiceListResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="CardApplicationServiceNameList"
                        maxOccurs="1" minOccurs="1">
                        <complexType>
                            <sequence>
                                <element
                                    name="CardApplicationServiceName" type="string"
                                    maxOccurs="unbounded" minOccurs="0">
                                </element>
                            </sequence>
                        </complexType>
```

```
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
</element>

<!-- 8.6 CardApplicationServiceCreate -->

<element name="CardApplicationServiceCreate">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="CardApplicationServiceName"
                        type="string" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="CardApplicationServiceCreateResponse"
    type="iso:ResponseType" />

<!-- 8.7 CardApplicationServiceLoad -->

<element name="CardApplicationServiceLoad">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="CardApplicationServiceName"
                        type="string" />
                    <element name="Code"
                        type="iso:CardApplicationServiceLoadPackageType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="CardApplicationServiceLoadResponse"
    type="iso:ResponseType" />

<!-- 8.8 CardApplicationServiceDelete -->


<element name="CardApplicationServiceDelete">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="CardApplicationServiceName"
```

```
                    type="string" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
</element>

<element name="CardApplicationServiceDeleteResponse"
    type="iso:ResponseType" />

<!-- 8.9 CardApplicationServiceDescribe -->

<element name="CardApplicationServiceDescribe">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="CardApplicationServiceName"
                        type="string" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="CardApplicationServiceDescribeResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="ServiceDescription"
                        type="iso:CardApplicationServiceDescriptionType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<!-- 8.10 ExecuteAction -->

<element name="ExecuteAction">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="CardApplicationServiceName"
                        type="string">
                    </element>
                    <element name="ActionName"
                        type="iso:ActionNameType" />
                    <element name="Request"
                        type="iso:ExecuteActionRequestType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

```
</element>

<element name="ExecuteActionResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="0">
                    <element name="Confirmation"
                        type="iso:ExecuteActionConfirmationType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<!-- ======================== -->
<!--    Named data service    -->
<!-- ======================== -->
<!--          Section 9       -->
<!-- ======================== -->

<!-- 9.2 DataSetList -->

<element name="DataSetList">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DataSetListResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="DataSetNameList"
                        type="iso:DataSetNameListType">
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>


<!-- 9.3 DataSetCreate -->

<element name="DataSetCreate">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
```

```
                <element name="DataSetName"
                    type="iso:DataSetNameType" />
                <element name="DataSetACL"
                    type="iso:AccessControlListType" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
</element>

<element name="DataSetCreateResponse" type="iso:ResponseType" />

<!-- 9.4 DataSetSelect -->

<element name="DataSetSelect">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DataSetName"
                        type="iso:DataSetNameType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DataSetSelectResponse" type="iso:ResponseType" />

<!-- 9.5 DataSetDelete -->

<element name="DataSetDelete">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DataSetName"
                        type="iso:DataSetNameType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DataSetDeleteResponse" type="iso:ResponseType" />

<!-- 9.6 DSIList -->

<element name="DSIList">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                </sequence>
```

```
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DSIListResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="DSINameList"
                        type="iso:DSINameListType">
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<!-- 9.7 DSICreate -->

<element name="DSICreate">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DSIName" type="iso:DSINameType" />
                    <element name="DSIContent" type="hexBinary"></element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DSICreateResponse" type="iso:ResponseType" />

<!-- 9.8 DSIDelete -->

<element name="DSIDelete">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DSIName" type="iso:DSINameType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DSIDeleteResponse" type="iso:ResponseType" />

<!-- 9.9 DSIWrite -->

<element name="DSIWrite">
    <complexType>
```

```
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DSIName" type="iso:DSINameType" />
                    <element name="DSIContent" type="hexBinary" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DSIWriteResponse" type="iso:ResponseType" />

<!-- 9.10 DSIRead -->

<element name="DSIRead">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DSIName" type="iso:DSINameType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DSIReadResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence>
                    <element name="DSIContent" type="hexBinary"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>



<!-- ============================ -->
<!--    Cryptographic service     -->
<!-- ============================ -->
<!--           Section 10         -->
<!-- ============================ -->

<!-- 10.2 Encipher -->

<element name="Encipher">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
```

```
                    type="iso:ConnectionHandleType" />
                <element name="DIDScope" type="iso:DIDScopeType"
                    maxOccurs="1" minOccurs="0" />
                <element name="DIDName" type="iso:NameType" />
                <element name="PlainText" type="hexBinary" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
</element>

<element name="EncipherResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="CipherText" type="hexBinary"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<!-- 10.3 Decipher -->

<element name="Decipher">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDScope" type="iso:DIDScopeType"
                        maxOccurs="1" minOccurs="0" />
                    <element name="DIDName" type="iso:NameType" />
                    <element name="CipherText" type="hexBinary" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DecipherResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="PlainText" type="hexBinary"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<!-- 10.4 GetRandom -->

<element name="GetRandom">
    <complexType>
```

```
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDScope" type="iso:DIDScopeType"
                        maxOccurs="1" minOccurs="0" />
                    <element name="DIDName" type="iso:NameType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="GetRandomResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="Random" type="hexBinary"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<!-- 10.5 Hash -->

<element name="Hash">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDScope" type="iso:DIDScopeType"
                        maxOccurs="1" minOccurs="0" />
                    <element name="DIDName" type="iso:NameType" />
                    <element name="Message" type="hexBinary"
                        maxOccurs="1" minOccurs="1">
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="HashResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence>
                    <element name="Hash" type="hexBinary"
                        maxOccurs="1" minOccurs="0">
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

```
</element>

<!-- 10.6 Sign -->


<element name="Sign">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDScope" type="iso:DIDScopeType"
                        maxOccurs="1" minOccurs="0" />
                    <element name="DIDName" type="iso:NameType" />
                    <element name="Message" type="hexBinary" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="SignResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="Signature" type="hexBinary"
                        maxOccurs="1" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>


<!-- 10.7 VerifySignature -->

<element name="VerifySignature">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDScope" type="iso:DIDScopeType"
                        maxOccurs="1" minOccurs="0" />
                    <element name="DIDName" type="iso:NameType" />
                    <element name="Signature" type="hexBinary" />
                    <element name="Message" type="hexBinary"
                        maxOccurs="1" minOccurs="0">
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="VerifySignatureResponse" type="iso:ResponseType" />
```

```
<!-- 10.8 VerifyCertificate -->

<element name="VerifyCertificate">
   <complexType>
      <complexContent>
         <extension base="iso:RequestType">
            <sequence>
               <element name="ConnectionHandle"
                  type="iso:ConnectionHandleType" />
               <element name="DIDScope" type="iso:DIDScopeType"
                  maxOccurs="1" minOccurs="0">
               </element>
               <element name="RootCert" type="iso:DIDNameType"
                  maxOccurs="1" minOccurs="0" />
               <element name="CertificateType" type="anyURI"
                  maxOccurs="1" minOccurs="0">
               </element>
               <element name="Certificate" type="hexBinary"
                  maxOccurs="1" minOccurs="1">
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
</element>

<element name="VerifyCertificateResponse" type="iso:ResponseType" />



<!-- ================================== -->
<!--    Differential-Identity service   -->
<!-- ================================== -->
<!--             Section 11             -->
<!-- ================================== -->

<!-- 11.2 DIDList -->

<element name="DIDList">
   <complexType>
      <complexContent>
         <extension base="iso:RequestType">
            <sequence>
               <element name="ConnectionHandle"
                  type="iso:ConnectionHandleType" />
               <element name="Filter"
                  type="iso:DIDQualifierType" maxOccurs="1" minOccurs="0">
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
</element>

<element name="DIDListResponse">
   <complexType>
      <complexContent>
         <extension base="iso:ResponseType">
            <sequence>
               <element name="DIDNameList"
```

```
                    type="iso:DIDNameListType" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
</element>

<!-- 11.3 DIDCreate -->

<element name="DIDCreate">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDName" type="iso:DIDNameType" />
                    <element name="DIDUpdateData"
                        type="iso:DIDUpdateDataType" />
                    <element name="DIDACL"
                        type="iso:AccessControlListType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>


<element name="DIDCreateResponse" type="iso:ResponseType" />

<!-- 11.4 DIDGet -->

<element name="DIDGet">
    <complexType>
        <complexContent>
            <extension base="iso:RequestType">
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="DIDScope" type="iso:DIDScopeType"
                        maxOccurs="1" minOccurs="0" />
                    <element name="DIDName" type="iso:NameType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="DIDGetResponse">
    <complexType>
        <complexContent>
            <extension base="iso:ResponseType">
                <sequence maxOccurs="1" minOccurs="1">
                    <element name="DIDStructure"
                        type="iso:DIDStructureType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>
```

```
    <!-- 11.5 DIDUpdate -->

    <element name="DIDUpdate">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                        <element name="DIDName" type="iso:NameType" />
                        <element name="DIDUpdateData"
                            type="iso:DIDUpdateDataType">
                        </element>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="DIDUpdateResponse" type="iso:ResponseType" />

    <!-- 11.6 DIDDelete -->

    <element name="DIDDelete">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                        <element name="DIDName" type="iso:NameType" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="DIDDeleteResponse" type="iso:ResponseType" />

    <!-- 11.7 DIDAuthenticate -->

    <element name="DIDAuthenticate">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" />
                        <element name="DIDScope" type="iso:DIDScopeType"
                            maxOccurs="1" minOccurs="0" />
                        <element name="DIDName" type="iso:NameType" />
                        <element name="AuthenticationProtocolData"
                            type="iso:DIDAuthenticationDataType" />
                        <element name="SAMConnectionHandle"
                            type="iso:ConnectionHandleType" maxOccurs="1"
minOccurs="0">
                        </element>
                    </sequence>
                </extension>
```

```
            </complexContent>
        </complexType>
    </element>

    <element name="DIDAuthenticateResponse">
        <complexType>
            <complexContent>
                <extension base="iso:ResponseType">
                    <sequence maxOccurs="1" minOccurs="1">
                        <element name="AuthenticationProtocolData"
                            type="iso:DIDAuthenticationDataType" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <!-- ================================== -->
    <!--        Authorization service       -->
    <!-- ================================== -->
    <!--               Section 12           -->
    <!-- ================================== -->

    <!-- 12.2 ACLList -->

    <element name="ACLList">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
                    <sequence>
                        <element name="ConnectionHandle"
                            type="iso:ConnectionHandleType" maxOccurs="1"
minOccurs="1" />
                        <element name="TargetName"
                            type="iso:TargetNameType" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="ACLListResponse">
        <complexType>
            <complexContent>
                <extension base="iso:ResponseType">
                    <sequence>
                        <element name="TargetACL"
                            type="iso:AccessControlListType" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <!-- 12.3 ACLModify -->

    <element name="ACLModify">
        <complexType>
            <complexContent>
                <extension base="iso:RequestType">
```

```
                <sequence>
                    <element name="ConnectionHandle"
                        type="iso:ConnectionHandleType" />
                    <element name="TargetName"
                        type="iso:TargetNameType" />
                    <element name="CardApplicationServiceName"
                        type="string" maxOccurs="1" minOccurs="1" />
                    <element name="ActionName"
                        type="iso:ActionNameType" />
                    <element name="SecurityCondition"
                        type="iso:SecurityConditionType" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>
<element name="ACLModifyResponse" type="iso:ResponseType" />
</schema>
```

F.3 WSDL definitions for Service Access Layer functions (ISO24727-3.wsdl)

```
<wsdl:definitions
    targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

    <!-- ================== -->
    <!-- Definition of types -->
    <!-- (only include XSDs) -->
    <!-- ================== -->

    <wsdl:types>
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified"
            targetNamespace="urn:iso:std:iso-iec:24727:tech:schema">
            <xsd:include schemaLocation="ISO24727-3.xsd" />
        </xsd:schema>
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        </xsd:schema>
    </wsdl:types>

    <!-- ====================== -->
    <!-- Definition of messages -->
    <!-- ====================== -->

    <!-- ================================== -->
    <!-- 6. Card-application-service access -->
    <!-- (messages)                         -->
    <!-- ================================== -->

    <!-- 6.2 Initialize -->

    <wsdl:message name="InitializeRequest">
        <wsdl:part name="parameters" element="iso:Initialize" />
    </wsdl:message>
    <wsdl:message name="InitializeResponse">
        <wsdl:part name="parameters" element="iso:InitializeResponse" />
    </wsdl:message>

    <!-- 6.3 Terminate -->

    <wsdl:message name="TerminateRequest">
        <wsdl:part name="parameters" element="iso:Terminate" />
    </wsdl:message>
    <wsdl:message name="TerminateResponse">
        <wsdl:part name="parameters" element="iso:TerminateResponse" />
    </wsdl:message>

    <!-- 6.4 CardApplicationPath -->

    <wsdl:message name="CardApplicationPathRequest">
        <wsdl:part name="parameters" element="iso:CardApplicationPath" />
    </wsdl:message>
    <wsdl:message name="CardApplicationPathResponse">
        <wsdl:part name="parameters"
            element="iso:CardApplicationPathResponse" />
    </wsdl:message>
```

```
<!-- ====================== -->
<!-- 7. Connection service -->
<!-- (messages)            -->
<!-- ====================== -->

<!-- 7.2 CardApplicationConnect -->

<wsdl:message name="CardApplicationConnectRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationConnect">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationConnectResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationConnectResponse">
    </wsdl:part>
</wsdl:message>

<!-- 7.3 CardApplicationDisconnect -->

<wsdl:message name="CardApplicationDisconnectRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationDisconnect">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationDisconnectResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationDisconnectResponse">
    </wsdl:part>
</wsdl:message>

<!-- 7.4 CardApplicationStartSession -->

<wsdl:message name="CardApplicationStartSessionRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationStartSession">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationStartSessionResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationStartSessionResponse">
    </wsdl:part>
</wsdl:message>

<!-- 7.5 CardApplicationEndSession -->

<wsdl:message name="CardApplicationEndSessionRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationEndSession">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationEndSessionResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationEndSessionResponse">
    </wsdl:part>
</wsdl:message>

<!-- ========================== -->
<!-- 8. Card-application service -->
```

```
<!-- (messages)                    -->
<!-- ============================ -->

<!-- 8.2 CardApplicationList -->

<wsdl:message name="CardApplicationListRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationList">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationListResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationListResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.3 CardApplicationCreate -->

<wsdl:message name="CardApplicationCreateRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationCreate">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationCreateResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationCreateResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.4 CardApplicationDelete -->

<wsdl:message name="CardApplicationDeleteRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationDelete">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationDeleteResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationDeleteResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.5 CardApplicationServiceList -->

<wsdl:message name="CardApplicationServiceListRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceList">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationServiceListResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceListResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.6 CardApplicationServiceCreate -->

<wsdl:message name="CardApplicationServiceCreateRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceCreate">
```

```
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationServiceCreateResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceCreateResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.7 CardApplicationServiceLoad -->

<wsdl:message name="CardApplicationServiceLoadRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceLoad">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationServiceLoadResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceLoadResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.8 CardApplicationServiceDelete -->

<wsdl:message name="CardApplicationServiceDeleteRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceDelete">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationServiceDeleteResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceDeleteResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.9 CardApplicationServiceDescribe -->

<wsdl:message name="CardApplicationServiceDescribeRequest">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceDescribe">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="CardApplicationServiceDescribeResponse">
    <wsdl:part name="parameters"
        element="iso:CardApplicationServiceDescribeResponse">
    </wsdl:part>
</wsdl:message>

<!-- 8.10 ExecuteAction -->

<wsdl:message name="ExecuteActionRequest">
    <wsdl:part name="parameters" element="iso:ExecuteAction"></wsdl:part>
</wsdl:message>
<wsdl:message name="ExecuteActionResponse">
    <wsdl:part name="parameters"
        element="iso:ExecuteActionResponse">
    </wsdl:part>
</wsdl:message>

<!-- ====================== -->
<!-- 9. Named data service -->
```

```
<!-- (messages)             -->
<!-- ====================== -->

<!-- 9.2 DataSetList -->

<wsdl:message name="DataSetListRequest">
    <wsdl:part name="parameters" element="iso:DataSetList"></wsdl:part>
</wsdl:message>
<wsdl:message name="DataSetListResponse">
    <wsdl:part name="parameters"
        element="iso:DataSetListResponse">
    </wsdl:part>
</wsdl:message>

<!-- 9.3 DataSetCreate -->

<wsdl:message name="DataSetCreateRequest">
    <wsdl:part name="parameters" element="iso:DataSetCreate"></wsdl:part>
</wsdl:message>
<wsdl:message name="DataSetCreateResponse">
    <wsdl:part name="parameters"
        element="iso:DataSetCreateResponse">
    </wsdl:part>
</wsdl:message>

<!-- 9.4 DataSetSelect -->

<wsdl:message name="DataSetSelectRequest">
    <wsdl:part name="parameters" element="iso:DataSetSelect"></wsdl:part>
</wsdl:message>
<wsdl:message name="DataSetSelectResponse">
    <wsdl:part name="parameters"
        element="iso:DataSetSelectResponse">
    </wsdl:part>
</wsdl:message>

<!-- 9.5 DataSetDelete -->

<wsdl:message name="DataSetDeleteRequest">
    <wsdl:part name="parameters" element="iso:DataSetDelete"></wsdl:part>
</wsdl:message>
<wsdl:message name="DataSetDeleteResponse">
    <wsdl:part name="parameters"
        element="iso:DataSetDeleteResponse">
    </wsdl:part>
</wsdl:message>

<!-- 9.6 DSIList -->

<wsdl:message name="DSIListRequest">
    <wsdl:part name="parameters" element="iso:DSIList"></wsdl:part>
</wsdl:message>
<wsdl:message name="DSIListResponse">
    <wsdl:part name="parameters" element="iso:DSIListResponse"></wsdl:part>
</wsdl:message>

<!-- 9.7 DSICreate -->

<wsdl:message name="DSICreateRequest">
    <wsdl:part name="parameters" element="iso:DSICreate"></wsdl:part>
```

```
</wsdl:message>
<wsdl:message name="DSICreateResponse">
    <wsdl:part name="parameters" element="iso:DSICreateResponse"></wsdl:part>
</wsdl:message>

<!-- 9.8 DSIDelete -->

<wsdl:message name="DSIDeleteRequest">
    <wsdl:part name="parameters" element="iso:DSIDelete"></wsdl:part>
</wsdl:message>
<wsdl:message name="DSIDeleteResponse">
    <wsdl:part name="parameters" element="iso:DSIDeleteResponse"></wsdl:part>
</wsdl:message>

<!-- 9.9 DSIWrite -->

<wsdl:message name="DSIWriteRequest">
    <wsdl:part name="parameters" element="iso:DSIWrite"></wsdl:part>
</wsdl:message>
<wsdl:message name="DSIWriteResponse">
    <wsdl:part name="parameters" element="iso:DSIWriteResponse"></wsdl:part>
</wsdl:message>

<!-- 9.10 DSIRead -->

<wsdl:message name="DSIReadRequest">
    <wsdl:part name="parameters" element="iso:DSIRead"></wsdl:part>
</wsdl:message>
<wsdl:message name="DSIReadResponse">
    <wsdl:part name="parameters" element="iso:DSIReadResponse"></wsdl:part>
</wsdl:message>

<!-- ========================== -->
<!-- 10. Cryptographic service  -->
<!-- (messages)                 -->
<!-- ========================== -->

<!-- 10.2 Encipher -->

<wsdl:message name="EncipherRequest">
    <wsdl:part name="parameters" element="iso:Encipher"></wsdl:part>
</wsdl:message>
<wsdl:message name="EncipherResponse">
    <wsdl:part name="parameters" element="iso:EncipherResponse"></wsdl:part>
</wsdl:message>

<!-- 10.3 Decipher -->

<wsdl:message name="DecipherRequest">
    <wsdl:part name="parameters" element="iso:Decipher"></wsdl:part>
</wsdl:message>
<wsdl:message name="DecipherResponse">
    <wsdl:part name="parameters" element="iso:DecipherResponse"></wsdl:part>
</wsdl:message>

<!-- 10.4 GetRandom -->

<wsdl:message name="GetRandomRequest">
    <wsdl:part name="parameters" element="iso:GetRandom"></wsdl:part>
</wsdl:message>
```

```
<wsdl:message name="GetRandomResponse">
    <wsdl:part name="parameters" element="iso:GetRandomResponse"></wsdl:part>
</wsdl:message>

<!-- 10.5 Hash -->

<wsdl:message name="HashRequest">
    <wsdl:part name="parameters" element="iso:Hash"></wsdl:part>
</wsdl:message>
<wsdl:message name="HashResponse">
    <wsdl:part name="parameters" element="iso:HashResponse"></wsdl:part>
</wsdl:message>

<!-- 10.6 Sign -->

<wsdl:message name="SignRequest">
    <wsdl:part name="parameters" element="iso:Sign"></wsdl:part>
</wsdl:message>
<wsdl:message name="SignResponse">
    <wsdl:part name="parameters" element="iso:SignResponse"></wsdl:part>
</wsdl:message>

<!-- 10.7 Sign -->

<wsdl:message name="VerifySignatureRequest">
    <wsdl:part name="parameters" element="iso:VerifySignature"></wsdl:part>
</wsdl:message>
<wsdl:message name="VerifySignatureResponse">
    <wsdl:part name="parameters"
        element="iso:VerifySignatureResponse">
    </wsdl:part>
</wsdl:message>

<!-- 10.8 Sign -->

<wsdl:message name="VerifyCertificateRequest">
    <wsdl:part name="parameters" element="iso:VerifyCertificate"></wsdl:part>
</wsdl:message>
<wsdl:message name="VerifyCertificateResponse">
    <wsdl:part name="parameters"
        element="iso:VerifyCertificateResponse">
    </wsdl:part>
</wsdl:message>

<!-- ================================= -->
<!-- 11. Differential-identity service -->
<!--     (messages)                    -->
<!-- ================================= -->

<!-- 11.2 DIDList -->

<wsdl:message name="DIDListRequest">
    <wsdl:part name="parameters" element="iso:DIDList"></wsdl:part>
</wsdl:message>
<wsdl:message name="DIDListResponse">
    <wsdl:part name="parameters" element="iso:DIDListResponse"></wsdl:part>
</wsdl:message>

<!-- 11.3 DIDCreate -->
```

```
<wsdl:message name="DIDCreateRequest">
    <wsdl:part name="parameters" element="iso:DIDCreate"></wsdl:part>
</wsdl:message>
<wsdl:message name="DIDCreateResponse">
    <wsdl:part name="parameters" element="iso:DIDCreateResponse"></wsdl:part>
</wsdl:message>

<!-- 11.4 DIDGet -->

<wsdl:message name="DIDGetRequest">
    <wsdl:part name="parameters" element="iso:DIDGet"></wsdl:part>
</wsdl:message>
<wsdl:message name="DIDGetResponse">
    <wsdl:part name="parameters" element="iso:DIDGetResponse"></wsdl:part>
</wsdl:message>

<!-- 11.5 DIDUpdate -->

<wsdl:message name="DIDUpdateRequest">
    <wsdl:part name="parameters" element="iso:DIDUpdate"></wsdl:part>
</wsdl:message>
<wsdl:message name="DIDUpdateResponse">
    <wsdl:part name="parameters" element="iso:DIDUpdateResponse"></wsdl:part>
</wsdl:message>

<!-- 11.6 DIDDelete -->

<wsdl:message name="DIDDeleteRequest">
    <wsdl:part name="parameters" element="iso:DIDDelete"></wsdl:part>
</wsdl:message>
<wsdl:message name="DIDDeleteResponse">
    <wsdl:part name="parameters" element="iso:DIDDeleteResponse"></wsdl:part>
</wsdl:message>

<!-- 11.7 DIDAuthenticate -->

<wsdl:message name="DIDAuthenticateRequest">
    <wsdl:part name="parameters" element="iso:DIDAuthenticate"></wsdl:part>
</wsdl:message>
<wsdl:message name="DIDAuthenticateResponse">
    <wsdl:part name="parameters"
        element="iso:DIDAuthenticateResponse">
    </wsdl:part>
</wsdl:message>

<!-- ========================== -->
<!-- 12. Authorization service -->
<!-- (messages)                 -->
<!-- ========================== -->

<!-- 12.2 ACLList -->

<wsdl:message name="ACLListRequest">
    <wsdl:part name="parameters" element="iso:ACLList"></wsdl:part>
</wsdl:message>
<wsdl:message name="ACLListResponse">
    <wsdl:part name="parameters" element="iso:ACLListResponse"></wsdl:part>
</wsdl:message>

<!-- 12.3 ACLModify -->
```

```
    <wsdl:message name="ACLModifyRequest">
        <wsdl:part name="parameters" element="iso:ACLModify"></wsdl:part>
    </wsdl:message>
    <wsdl:message name="ACLModifyResponse">
        <wsdl:part name="parameters" element="iso:ACLModifyResponse"></wsdl:part>
    </wsdl:message>

    <!-- ====================== -->
    <!-- Definition of portType -->
    <!-- ====================== -->


    <wsdl:portType name="SAL">

        <!-- ================================== -->
        <!-- 6. Card-application-service access -->
        <!-- (portType)                         -->
        <!-- ================================== -->

        <!-- 6.2 Initialize -->

        <wsdl:operation name="Initialize">
            <wsdl:input message="iso:InitializeRequest" />
            <wsdl:output message="iso:InitializeResponse" />
        </wsdl:operation>

        <!-- 6.3 Terminate -->

        <wsdl:operation name="Terminate">
            <wsdl:input message="iso:TerminateRequest" />
            <wsdl:output message="iso:TerminateResponse" />
        </wsdl:operation>

        <!-- 6.4 CardApplicationPath -->

        <wsdl:operation name="CardApplicationPath">
            <wsdl:input message="iso:CardApplicationPathRequest" />
            <wsdl:output message="iso:CardApplicationPathResponse" />
        </wsdl:operation>

        <!-- ==================== -->
        <!-- 7. Connection service -->
        <!-- (portType)            -->
        <!-- ==================== -->

        <!-- 7.2 CardApplicationConnect -->

        <wsdl:operation name="CardApplicationConnect">
            <wsdl:input message="iso:CardApplicationConnectRequest"></wsdl:input>
            <wsdl:output
message="iso:CardApplicationConnectResponse"></wsdl:output>
        </wsdl:operation>

        <!-- 7.3 CardApplicationDisconnect -->

        <wsdl:operation name="CardApplicationDisconnect">
            <wsdl:input
                message="iso:CardApplicationDisconnectRequest">
            </wsdl:input>
```

```
    <wsdl:output
        message="iso:CardApplicationDisconnectResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 7.4 CardApplicationStartSession -->

<wsdl:operation name="CardApplicationStartSession">
    <wsdl:input
        message="iso:CardApplicationStartSessionRequest">
    </wsdl:input>
    <wsdl:output
        message="iso:CardApplicationStartSessionResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 7.5 CardApplicationEndSession -->

<wsdl:operation name="CardApplicationEndSession">
    <wsdl:input
        message="iso:CardApplicationEndSessionRequest">
    </wsdl:input>
    <wsdl:output
        message="iso:CardApplicationEndSessionResponse">
    </wsdl:output>
</wsdl:operation>


<!-- ============================ -->
<!-- 8. Card-application service  -->
<!-- (portType)                   -->
<!-- ============================ -->

<!-- 8.2 CardApplicationList -->

<wsdl:operation name="CardApplicationList">
    <wsdl:input message="iso:CardApplicationListRequest"></wsdl:input>
    <wsdl:output message="iso:CardApplicationListResponse"></wsdl:output>
</wsdl:operation>

<!-- 8.3 CardApplicationCreate -->

<wsdl:operation name="CardApplicationCreate">
    <wsdl:input message="iso:CardApplicationCreateRequest"></wsdl:input>
    <wsdl:output message="iso:CardApplicationCreateResponse"></wsdl:output>
</wsdl:operation>

<!-- 8.4 CardApplicationDelete -->

<wsdl:operation name="CardApplicationDelete">
    <wsdl:input message="iso:CardApplicationDeleteRequest"></wsdl:input>
    <wsdl:output message="iso:CardApplicationDeleteResponse"></wsdl:output>
</wsdl:operation>

<!-- 8.5 CardApplicationServiceList -->

<wsdl:operation name="CardApplicationServiceList">
    <wsdl:input
        message="iso:CardApplicationServiceListRequest">
    </wsdl:input>
```

```
    <wsdl:output
        message="iso:CardApplicationServiceListResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 8.6 CardApplicationServiceCreate -->

<wsdl:operation name="CardApplicationServiceCreate">
    <wsdl:input
        message="iso:CardApplicationServiceCreateRequest">
    </wsdl:input>
    <wsdl:output
        message="iso:CardApplicationServiceCreateResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 8.7 CardApplicationServiceLoad -->

<wsdl:operation name="CardApplicationServiceLoad">
    <wsdl:input
        message="iso:CardApplicationServiceLoadRequest">
    </wsdl:input>
    <wsdl:output
        message="iso:CardApplicationServiceLoadResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 8.8 CardApplicationServiceDelete -->

<wsdl:operation name="CardApplicationServiceDelete">
    <wsdl:input
        message="iso:CardApplicationServiceDeleteRequest">
    </wsdl:input>
    <wsdl:output
        message="iso:CardApplicationServiceDeleteResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 8.9 CardApplicationServiceDescribe -->

<wsdl:operation name="CardApplicationServiceDescribe">
    <wsdl:input
        message="iso:CardApplicationServiceDescribeRequest">
    </wsdl:input>
    <wsdl:output
        message="iso:CardApplicationServiceDescribeResponse">
    </wsdl:output>
</wsdl:operation>

<!-- 8.10 ExecuteAction -->

<wsdl:operation name="ExecuteAction">
    <wsdl:input message="iso:ExecuteActionRequest"></wsdl:input>
    <wsdl:output message="iso:ExecuteActionResponse"></wsdl:output>
</wsdl:operation>


<!-- ====================== -->
<!-- 9. Named data service  -->
<!-- (portType)             -->
```

```
<!-- ====================== -->

<!-- 9.2 DataSetList -->

<wsdl:operation name="DataSetList">
    <wsdl:input message="iso:DataSetListRequest"></wsdl:input>
    <wsdl:output message="iso:DataSetListResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.3 DataSetCreate -->

<wsdl:operation name="DataSetCreate">
    <wsdl:input message="iso:DataSetCreateRequest"></wsdl:input>
    <wsdl:output message="iso:DataSetCreateResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.4 DataSetSelect -->

<wsdl:operation name="DataSetSelect">
    <wsdl:input message="iso:DataSetSelectRequest"></wsdl:input>
    <wsdl:output message="iso:DataSetSelectResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.5 DataSetDelete -->

<wsdl:operation name="DataSetDelete">
    <wsdl:input message="iso:DataSetDeleteRequest"></wsdl:input>
    <wsdl:output message="iso:DataSetDeleteResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.6 DSIList -->

<wsdl:operation name="DSIList">
    <wsdl:input message="iso:DSIListRequest"></wsdl:input>
    <wsdl:output message="iso:DSIListResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.7 DSICreate -->

<wsdl:operation name="DSICreate">
    <wsdl:input message="iso:DSICreateRequest"></wsdl:input>
    <wsdl:output message="iso:DSICreateResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.8 DSIDelete -->

<wsdl:operation name="DSIDelete">
    <wsdl:input message="iso:DSIDeleteRequest"></wsdl:input>
    <wsdl:output message="iso:DSIDeleteResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.9 DSIWrite -->

<wsdl:operation name="DSIWrite">
    <wsdl:input message="iso:DSIWriteRequest"></wsdl:input>
    <wsdl:output message="iso:DSIWriteResponse"></wsdl:output>
</wsdl:operation>

<!-- 9.10 DSIRead -->
```

```
<wsdl:operation name="DSIRead">
    <wsdl:input message="iso:DSIReadRequest"></wsdl:input>
    <wsdl:output message="iso:DSIReadResponse"></wsdl:output>
</wsdl:operation>

<!-- ========================== -->
<!-- 10. Cryptographic service  -->
<!-- (portType)                 -->
<!-- ========================== -->

<!-- 10.2 Encipher -->

<wsdl:operation name="Encipher">
    <wsdl:input message="iso:EncipherRequest"></wsdl:input>
    <wsdl:output message="iso:EncipherResponse"></wsdl:output>
</wsdl:operation>

<!-- 10.3 Decipher -->

<wsdl:operation name="Decipher">
    <wsdl:input message="iso:DecipherRequest"></wsdl:input>
    <wsdl:output message="iso:DecipherResponse"></wsdl:output>
</wsdl:operation>

<!-- 10.4 GetRandom -->

<wsdl:operation name="GetRandom">
    <wsdl:input message="iso:GetRandomRequest"></wsdl:input>
    <wsdl:output message="iso:GetRandomResponse"></wsdl:output>
</wsdl:operation>

<!-- 10.5 Hash -->

<wsdl:operation name="Hash">
    <wsdl:input message="iso:HashRequest"></wsdl:input>
    <wsdl:output message="iso:HashResponse"></wsdl:output>
</wsdl:operation>

<!-- 10.6 Sign -->

<wsdl:operation name="Sign">
    <wsdl:input message="iso:SignRequest"></wsdl:input>
    <wsdl:output message="iso:SignResponse"></wsdl:output>
</wsdl:operation>

<!-- 10.7 VerifySignature -->

<wsdl:operation name="VerifySignature">
    <wsdl:input message="iso:VerifySignatureRequest"></wsdl:input>
    <wsdl:output message="iso:VerifySignatureResponse"></wsdl:output>
</wsdl:operation>

<!-- 10.8 VerifyCertificate -->

<wsdl:operation name="VerifyCertificate">
    <wsdl:input message="iso:VerifyCertificateRequest"></wsdl:input>
    <wsdl:output message="iso:VerifyCertificateResponse"></wsdl:output>
</wsdl:operation>

<!-- ================================== -->
```

```
<!-- 11. Differential-identity service  -->
<!-- (portType)                         -->
<!-- ================================== -->

<!-- 11.2 DIDList -->

<wsdl:operation name="DIDList">
    <wsdl:input message="iso:DIDListRequest"></wsdl:input>
    <wsdl:output message="iso:DIDListResponse"></wsdl:output>
</wsdl:operation>

<!-- 11.3 DIDCreate -->

<wsdl:operation name="DIDCreate">
    <wsdl:input message="iso:DIDCreateRequest"></wsdl:input>
    <wsdl:output message="iso:DIDCreateResponse"></wsdl:output>
</wsdl:operation>

<!-- 11.4 DIDGet -->

<wsdl:operation name="DIDGet">
    <wsdl:input message="iso:DIDGetRequest"></wsdl:input>
    <wsdl:output message="iso:DIDGetResponse"></wsdl:output>
</wsdl:operation>

<!-- 11.5 DIDUpdate -->

<wsdl:operation name="DIDUpdate">
    <wsdl:input message="iso:DIDUpdateRequest"></wsdl:input>
    <wsdl:output message="iso:DIDUpdateResponse"></wsdl:output>
</wsdl:operation>

<!-- 11.6 DIDDelete -->

<wsdl:operation name="DIDDelete">
    <wsdl:input message="iso:DIDDeleteRequest"></wsdl:input>
    <wsdl:output message="iso:DIDDeleteResponse"></wsdl:output>
</wsdl:operation>

<!-- 11.7 DIDAuthenticate -->

<wsdl:operation name="DIDAuthenticate">
    <wsdl:input message="iso:DIDAuthenticateRequest"></wsdl:input>
    <wsdl:output message="iso:DIDAuthenticateResponse"></wsdl:output>
</wsdl:operation>

<!-- ============================ -->
<!-- 12. Authorization service    -->
<!-- (portType)                   -->
<!-- ============================ -->

<!-- 12.2 DIDAuthenticate -->

<wsdl:operation name="ACLList">
    <wsdl:input message="iso:ACLListRequest"></wsdl:input>
    <wsdl:output message="iso:ACLListResponse"></wsdl:output>
</wsdl:operation>

<!-- 12.3 ACLModify -->
```

```
        <wsdl:operation name="ACLModify">
            <wsdl:input message="iso:ACLModifyRequest"></wsdl:input>
            <wsdl:output message="iso:ACLModifyResponse"></wsdl:output>
        </wsdl:operation>
    </wsdl:portType>

    <!-- ===================== -->
    <!-- Definition of Binding -->
    <!-- ===================== -->


    <wsdl:binding name="SAL" type="iso:SAL">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />

        <!-- ================================== -->
        <!-- 6. Card-application-service access -->
        <!-- (binding)                          -->
        <!-- ================================== -->

        <!-- 6.2 Initialize -->

        <wsdl:operation name="Initialize">
            <soap:operation
                soapAction="urn:iso:std:iso-iec:24727:tech:schema:Initialize" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 6.3 Terminate -->

        <wsdl:operation name="Terminate">
            <soap:operation
                soapAction="urn:iso:std:iso-iec:24727:tech:schema:Terminate" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 6.4 CardApplicationPath -->

        <wsdl:operation name="CardApplicationPath">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationPath" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
```

```
        <!-- ==================== -->
        <!-- 7. Connection service -->
        <!-- (binding)             -->
        <!-- ==================== -->

        <!-- 7.2 CardApplicationConnect -->

        <wsdl:operation name="CardApplicationConnect">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationConnect" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 7.3 CardApplicationDisconnect -->

        <wsdl:operation name="CardApplicationDisconnect">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationDisconnect" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 7.4 CardApplicationStartSession -->

        <wsdl:operation name="CardApplicationStartSession">
            <soap:operation
                soapAction="urn:iso:std:iso-iec:24727:tech:schema:StartSession" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 7.5 CardApplicationEndSession -->

        <wsdl:operation name="CardApplicationEndSession">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationEndSession" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
```

```
        <!-- ============================ -->
        <!-- 8. Card-application service  -->
        <!-- (binding)                    -->
        <!-- ============================ -->

        <!-- 8.2 CardApplicationList -->

        <wsdl:operation name="CardApplicationList">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationList" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.3 CardApplicationCreate -->

        <wsdl:operation name="CardApplicationCreate">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationCreate" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.4 CardApplicationDelete -->

        <wsdl:operation name="CardApplicationDelete">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationDelete" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.5 CardApplicationServiceList -->

        <wsdl:operation name="CardApplicationServiceList">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationServiceList" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
```

```
        <!-- 8.6 CardApplicationServiceCreate -->

        <wsdl:operation name="CardApplicationServiceCreate">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationServiceCreate" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.7 CardApplicationServiceLoad -->

        <wsdl:operation name="CardApplicationServiceLoad">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationServiceLoad" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.8 CardApplicationServiceDelete -->

        <wsdl:operation name="CardApplicationServiceDelete">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationServiceDelete" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.9 CardApplicationServiceDescribe -->

        <wsdl:operation name="CardApplicationServiceDescribe">
            <soap:operation
                soapAction="urn:iso:std:iso-
iec:24727:tech:schema:CardApplicationServiceDescribe" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>

        <!-- 8.10 ExecuteAction -->

        <wsdl:operation name="ExecuteAction">
```

```
    <soap:operation
        soapAction="urn:iso:std:iso-iec:24727:tech:schema:ExecuteAction" />
    <wsdl:input>
        <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal" />
    </wsdl:output>
</wsdl:operation>

<!-- ====================== -->
<!-- 9. Named data service  -->
<!-- (binding)              -->
<!-- ====================== -->

<!-- 9.2 DataSetList -->

<wsdl:operation name="DataSetList">
    <soap:operation
        soapAction="urn:iso:std:iso-iec:24727:tech:schema:DataSetList" />
    <wsdl:input>
        <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal" />
    </wsdl:output>
</wsdl:operation>

<!-- 9.3 DataSetCreate -->

<wsdl:operation name="DataSetCreate">
    <soap:operation
        soapAction="urn:iso:std:iso-iec:24727:tech:schema:DataSetCreate" />
    <wsdl:input>
        <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal" />
    </wsdl:output>
</wsdl:operation>

<!-- 9.4 DataSetSelect -->

<wsdl:operation name="DataSetSelect">
    <soap:operation
        soapAction="urn:iso:std:iso-iec:24727:tech:schema:DataSetSelect" />
    <wsdl:input>
        <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal" />
    </wsdl:output>
</wsdl:operation>

<!-- 9.5 DataSetDelete -->

<wsdl:operation name="DataSetDelete">
    <soap:operation
        soapAction="urn:iso:std:iso-iec:24727:tech:schema:DataSetDelete" />
    <wsdl:input>
```