

PD CEN/TS 15480-3:2014



BSI Standards Publication

# Identification card systems — European Citizen Card

Part 3: European Citizen Card Interoperability  
using an application interface

**bsi.**

...making excellence a habit.™

### **National foreword**

This Published Document is the UK implementation of CEN/TS 15480-3:2014. It supersedes DD CEN/TS 15480-3:2010 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/17, Cards and personal identification.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014.  
Published by BSI Standards Limited 2014

ISBN 978 0 580 82884 3  
ICS 35.240.15

**Compliance with a British Standard cannot confer immunity from legal obligations.**

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 30 April 2014.

### **Amendments/corrigenda issued since publication**

<b>Date</b>	<b>Text affected</b>
-------------	----------------------

---

English Version

**Identification card systems - European Citizen Card - Part 3:  
European Citizen Card Interoperability using an application  
interface**

Systèmes de carte d'identification - Carte Européenne du  
Citoyen - Partie 3 : Interopérabilité de la Carte européenne  
du Citoyen utilisant une interface applicative

Identifikationskartensysteme - Europäische Bürgerkarte -  
Teil 3: Anwendungsschnittstelle für die Interoperabilität von  
Europäischen Bürgerkarten

This Technical Specification (CEN/TS) was approved by CEN on 14 October 2013 for provisional application.

The period of validity of this CEN/TS is limited initially to three years. After two years the members of CEN will be requested to submit their comments, particularly on the question whether the CEN/TS can be converted into a European Standard.

CEN members are required to announce the existence of this CEN/TS in the same way as for an EN and to make the CEN/TS available promptly at national level in an appropriate form. It is permissible to keep conflicting national standards in force (in parallel to the CEN/TS) until the final decision about the possible conversion of the CEN/TS into an EN is reached.

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



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

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

## Contents

Page

Foreword.....	5
1 Scope .....	7
2 Normative references .....	7
3 Terms and definitions .....	8
4 Symbols and abbreviations .....	8
5 ECC fitting in ISO/IEC 24727 model .....	10
5.1 ISO/IEC 24727 main features .....	10
5.2 General security issues – Applicable ISO/IEC 24727-4 Stack Configurations for the ECC environment .....	12
5.3 ECC-3 Middleware Architecture .....	16
5.3.1 General.....	16
5.3.2 Service Access Layer (SAL) .....	17
5.3.3 Generic Card Access Layer (GCAL) .....	17
5.3.4 Interface Device Layer and API (IFD API).....	17
5.3.5 ECC-3 Stack Distribution and Connection Handling .....	17
5.3.6 Multi-stack composed configuration .....	20
5.3.7 A Web Service based architecture for ECC-3 framework.....	22
5.3.8 XML-based SAL interface .....	27
6 Card Discovery Mechanisms .....	28
6.1 General.....	28
6.2 Discovery decision tree .....	29
6.3 Migration path towards ECC and provision for legacy cards .....	29
6.3.1 General.....	29
6.3.2 Interoperable access to the Repository .....	30
6.4 Set of data for interoperability.....	30
6.5 Application and Card Capability Descriptors .....	31
6.6 ISO/IEC 7816-15 implementation.....	34
6.6.1 General.....	34
6.6.2 Profile designation within EF.DIR .....	34
6.6.3 ISO/IEC 24727-3 data structures mapping .....	35
6.6.4 ISO/IEC 24727-3 data structures storage onto the card .....	35
6.6.5 General discovery mechanism.....	37
6.7 Other data descriptor .....	39
7 Authentication protocols .....	39
7.1 General.....	39
7.2 Authentication Mechanisms based on ISO/IEC 24727 SAL-API .....	39
7.3 Asymmetric internal authentication.....	40
7.4 Asymmetric external authentication.....	40
7.5 Symmetric internal authentication.....	41
7.6 Symmetric external authentication .....	41
7.7 Mutual authentication with key establishment .....	41
7.8 Device authentication with non traceability.....	41
7.9 Key transport protocol based on RSA .....	41
7.10 Terminal Authentication.....	42
8 IFD-API Web Service Binding.....	42
9 Card-Info Structure — Introduction .....	42

<b>10</b>	<b>XML-based Service Access Layer Interface .....</b>	<b>43</b>
<b>11</b>	<b>Federative Framework-wise Authenticate API .....</b>	<b>43</b>
<b>11.1</b>	<b>General .....</b>	<b>43</b>
<b>11.2</b>	<b>Authenticate method .....</b>	<b>44</b>
<b>11.3</b>	<b>Web Service Binding for Authenticate API .....</b>	<b>47</b>
<b>11.3.1</b>	<b>General .....</b>	<b>47</b>
<b>11.3.2</b>	<b>Authenticate.XSD definition .....</b>	<b>47</b>
<b>11.3.3</b>	<b>Authenticate.WSDL definition .....</b>	<b>48</b>
<b>Annex A</b>	<b>(informative) Interface Device Layer Architecture and Management.....</b>	<b>51</b>
<b>A.1</b>	<b>Scope .....</b>	<b>51</b>
<b>A.2</b>	<b>IFD-Layer Architecture .....</b>	<b>51</b>
<b>A.3</b>	<b>Resource Manager .....</b>	<b>52</b>
<b>A.3.1</b>	<b>General .....</b>	<b>52</b>
<b>A.3.2</b>	<b>IFD-Handlers .....</b>	<b>52</b>
<b>A.3.3</b>	<b>Card transactions .....</b>	<b>52</b>
<b>A.3.4</b>	<b>Application threads .....</b>	<b>52</b>
<b>A.4</b>	<b>Administrative functions .....</b>	<b>52</b>
<b>A.4.1</b>	<b>IFD-Handler related functions .....</b>	<b>52</b>
<b>A.4.2</b>	<b>Interface Device related functions .....</b>	<b>53</b>
<b>Annex B</b>	<b>(informative) IFD-API – C Language Binding.....</b>	<b>54</b>
<b>Annex C</b>	<b>(informative) SAL-API Post-issuance personalisation requests .....</b>	<b>60</b>
<b>C.1</b>	<b>General .....</b>	<b>60</b>
<b>C.2</b>	<b>Post-issuance personalisation requests .....</b>	<b>60</b>
<b>C.3</b>	<b>Canonical protocol .....</b>	<b>60</b>
<b>C.3.1</b>	<b>General .....</b>	<b>60</b>
<b>C.3.2</b>	<b>DataSetCreate .....</b>	<b>61</b>
<b>C.3.3</b>	<b>DSICreate.....</b>	<b>68</b>
<b>C.3.4</b>	<b>DIDCreate .....</b>	<b>70</b>
<b>C.3.5</b>	<b>DIDUpdate .....</b>	<b>71</b>
<b>C.3.6</b>	<b>CardApplicationServiceCreate.....</b>	<b>72</b>
<b>C.4</b>	<b>General recommendation and conclusion.....</b>	<b>74</b>
<b>Annex D</b>	<b>(informative) Additional features versus ISO/IEC 24727 (all parts).....</b>	<b>75</b>
<b>D.1</b>	<b>General .....</b>	<b>75</b>
<b>D.2</b>	<b>Discovery Mechanism.....</b>	<b>75</b>
<b>D.3</b>	<b>General Procedures (SAL).....</b>	<b>75</b>
<b>D.4</b>	<b>Architecture .....</b>	<b>77</b>
<b>D.5</b>	<b>Differences between IFD-API in ISO/IEC 24727-4 and ECC-3 .....</b>	<b>77</b>
<b>D.5.1</b>	<b>More generale SlotCapabilityType.....</b>	<b>77</b>
<b>D.5.2</b>	<b>Transmit with support for batch processing .....</b>	<b>80</b>
<b>D.5.3</b>	<b>Additional error code for SignalEvent.....</b>	<b>82</b>
<b>Annex E</b>	<b>(informative) C-Language Binding for ExecuteSAL function .....</b>	<b>83</b>
<b>Annex F</b>	<b>(informative) Java-Language Binding for ExecuteSAL function .....</b>	<b>84</b>
<b>Annex G</b>	<b>(informative) Application Discovery Profile: card requirements to access/offer services in ISO/IEC 24727 framework .....</b>	<b>85</b>
<b>G.1</b>	<b>General .....</b>	<b>85</b>
<b>G.2</b>	<b>OID .....</b>	<b>85</b>
<b>G.3</b>	<b>General .....</b>	<b>85</b>
<b>G.4</b>	<b>interfaces / transport protocols .....</b>	<b>85</b>
<b>G.5</b>	<b>Data elements and data structures.....</b>	<b>86</b>
<b>G.6</b>	<b>Command set.....</b>	<b>88</b>
<b>G.7</b>	<b>Data structure of Card Applications.....</b>	<b>89</b>
<b>G.7.1</b>	<b>General .....</b>	<b>89</b>
<b>G.7.2</b>	<b>DF/ADF content .....</b>	<b>89</b>

<b>G.7.3</b>	<b>EF DCOD content.....</b>	<b>89</b>
<b>G.7.4</b>	<b>EF AOD content .....</b>	<b>90</b>
<b>G.7.5</b>	<b>EF SKD content.....</b>	<b>90</b>
<b>G.7.6</b>	<b>Ef PrKD content .....</b>	<b>90</b>
	<b>Bibliography .....</b>	<b>91</b>

## Foreword

This document (CEN/TS 15480-3:2014) has been prepared by Technical Committee CEN/TC 224 “Personal identification, electronic signature and cards and their related systems and operations”, the secretariat of which is held by AFNOR.

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

This document supersedes CEN/TS 15480-3:2010.

CEN/TS 15480, *Identification card systems — European Citizen Card*, is composed of the following parts:

- *Part 1: Physical, electrical and transport protocol characteristics;*
- *Part 2: Logical data structures and security services;*
- *Part 3: European Citizen Card Interoperability using an application interface* (the present document);
- *Part 4: Recommendations for European Citizen Card issuance, operation and use;*
- *Part 5: General Introduction.*

The following technical changes have been made in this new edition of CEN/TS 15480-3:

- addition of mention of SAL Lite component, abstraction of GCI and GCAL through Registry processed at SAL level, decision tree update, scope update, etc (5.3.5.3);
- removal of all subclauses under 6.6.3 (Data structures mapping) that were already incorporated in ISO/IEC 24727-4;
- removal of Annex J dedicated to ECC-3 API (handling ISO/IEC 7816-15 objects) considered not appropriate in ECC-3 because implementation-specific and not fundamental to interoperability;
- removal of XML Binding details for SAL API from Clause 10 and Annex G (removal of Annex G); it was incorporated in ISO/IEC 24727-3:2008/DAmD 1, Annex F;
- maintainance of the annex investigating SAL post-issuance personalisation;
- removal of Annex H describing XML binding for Authentication protocols since these protocols are now part of ISO/IEC 24727-3:2008/DAmD 1, i.e. EACv2 protocol binding doesn't need to be reflected in ECC-3 since it is incorporated in ISO/IEC 24727-3:2008, Annex E;
- removal of Annex D “example of CIA implementation for Card –Application Service description” since it is updated and incorporated in ISO/IEC 24727-4:2008/DAmD 1;
- removal of XML-based CardInfo Types (XML Registry) since it is incorporated in ISO/IEC 24727-3:2008/DAmD 1, Annex D, Clause D.3;
- IFD-API shows enhancements in comparison with ISO/IEC 24727 (e.g. SlotCapabilityType with support of transmission protocol descriptor, Transmit command with support of batch APDU, SignalEvent error coding with additional error code), therefore IFD API Annex B are removed from ECC-3 and the clauses describing enhancements are reflected in ECC-3, Annex D amongst the differences with ISO/IEC 24727;

- addition of Annex D, Additional features versus ISO/IEC 24727 (all parts), to incorporate the description of IFD API extensions in terms of API definition and binding;
- removal of 6.2.1.1, Definition for CardInfoRepository.XSD, and 6.2.1.2, Definition for CardInfoRepository.WSDL, since these binding descriptions are now part of ISO/IEC 24727-4:2008/DAMd, 1;
- addition of a new Clause 11 dedicated to Authenticate API: the Authenticate() call makes the service layer module transparent to the Service Provider, it occurs above SAL layer;
- provision of an introductory text describing the layout where Authenticate API fits;
- IFD API C-Language Binding remains in ECC-3 till its endorsement in ISO/IEC 24727 if deemed useful;
- maintenance of ExecuteSAL API in ECC-3 (both C-language binding and java binding);
- incorporation under Annex G of “Application Discovery Profile” for the purposes of integration in ISO/IEC 24727 framework.

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to announce this Technical Specification: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.



## 1 Scope

This Technical Specification provides an Interoperability Model, which will enable an eService compliant with technical requirements, to interoperate with different implementations of the European Citizen Card.

This Interoperability model will be developed as follows:

- starting from the ECC Part 2, Part 3 of the ECC series provides additional technical specifications for a middleware architecture based on ISO/IEC 24727 (all parts); this middleware will provide an API to an eService as per ISO/IEC 24727-3.
- a set of additional API provides the middleware stack with means to facilitate ECC services.
- a standard mechanism for the validation of the e-ID credential is stored in the ECC and retrieved by the eService.

In order to support the ECC services over an ISO/IEC 24727 middleware configuration, this part of the standard specifies the following:

- a set of mandatory requests to be supported by the middleware implementation based on ISO/IEC 24727 (all parts).
- data set content for interoperability to be personalised in the ECC.
- three middleware architecture solutions: one based on a stack of combined ISO/IEC 24727 configurations and the other based on Web Service configuration whereas the third one is relying on a SAL Lite component.
- an Application DiscoveryProfile featuring the guidelines for card-applications to fit in ISO/IEC 24727 framework.

## 2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

CEN/TS 15480-2:2012, *Identification card systems — European Citizen Card — Part 2: Logical data structures and security services*

CEN/TS 15480-4, *Identification card systems — European Citizen Card — Part 4: Recommendations for European Citizen Card issuance, operation and use*

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

ISO/IEC 7816-15, *Identification cards — Integrated circuit cards — Part 15: Cryptographic information application*

ISO/IEC 24727-1, *Identification cards — Integrated circuit card programming interfaces — Part 1: Architecture*

ISO/IEC 24727-2:2008<sup>1)</sup>, *Identification cards — Integrated circuit card programming interfaces — Part 2: Generic card interface*

ISO/IEC 24727-3:2008<sup>2)</sup>, *Identification cards — Integrated circuit card programming interfaces — Part 3: Application interface*

ISO/IEC 24727-4:2008<sup>3)</sup>, *Identification cards — Integrated circuit card programming interfaces — Part 4: Application programming interface (API) administration*

ISO/IEC 24727-5, *Identification cards — Integrated circuit card programming interfaces — Part 5: Testing procedures*

ISO/IEC 24727-6, *Identification cards — Integrated circuit card programming interfaces — Part 6: Registration authority procedures for the authentication protocols for interoperability*

### **3 Terms and definitions**

For the purposes of this document, the following terms and definitions apply.

#### **3.1 descriptive elements**

information nested in data objects and intended for the discovery mechanism and encapsulated along with procedural elements in the ACD and CCD

#### **3.2 procedural elements**

translation code to process any request at the Generic Card Interface (GCI) and every relevant card response

Note 1 to entry: The translation has one entry point, theTranslationCode() function as per ISO/IEC 24727-2.

#### **3.3 middleware**

set of abstraction layers which serves as the intermediate between a client-application and an application resident in the ECC and behind which the actual pieces of software running these abstraction layers are implementation-specific and out of the scope of this document

#### **3.4 eService**

application based locally on the client PC or based somewhere in the internet (eg government eService, eBusiness eService,...) which offers in combination with the ECC smart card the execution of a task

### **4 Symbols and abbreviations**

ADF	Application Dedicated File
AID	Application Identifier
AJAX	Asynchronous JavaScript and XML
AMB	Access Mode Byte
AT	Authentication Template

---

1) This document is currently impacted by the draft amendment ISO/IEC 24727-2:2008/DAmD 1.

2) This document is currently impacted by the draft amendment ISO/IEC 24727-3:2008/DAmD 1.

3) This document is currently impacted by the draft amendment ISO/IEC 24727-4:2008/DAmD 1.

ATR	Answer to Reset
ATS	Answer to Select
BER	Basic Encoding Rules
BHT	Biometric Header Template
BIT	Biometric Information Template
CA	Certification Authority
CAPICOM	Cryptographic API COM Object ( <a href="http://msdn.microsoft.com">http://msdn.microsoft.com</a> )
CAR	Certification Authority Reference
CBC	Cipher Block Chaining
CCT	Cryptographic Checksum Template
CED	Certificate Effective Date
CHA	Certificate Holder Authorization
CHR	Certificate Holder Reference
CIA	Cryptographic Information Application
CPI	Certificate Profile Identifier
CRT	Control Reference Template
CryptoAPI	Cryptographic Application Programming Interface ( <a href="http://msdn.microsoft.com">http://msdn.microsoft.com</a> )
CSP	Cryptographic Service Provider
CT	Confidentiality Template
CV	Card Verifiable
CXD	Certificate Expiration Date
DES	Data Encryption Standard
DF	Dedicated File
DH	Diffie Hellman
DOCP	Data Object Control Parameters
DST	Digital Signature Template
ECDH	Elliptic Curve DH
ELC	Elliptic Curve Cryptosystem
ECDSA	Elliptic Curve Digital Signature Algorithm
EF	Elementary File
FCI	File Control Information
FCP	File Control Parameters
GCAL	Generic Card Access Layer
HT	Hash Template
ICC	Integrated Circuit Card
DID	Differential Identity according to ISO/IEC 24727-3
IFD	Interface Device
IFDH	Interface Device Handler
JSON	Java Script Object Notation ( <a href="http://www.json.org">http://www.json.org</a> )

KAT	Control reference template for key agreement
LSB	Least Significant Byte
MAC	Message Authentication Code
MF	Master File
M.U.S.C.L.E	Movement for the Use of Smart Card in Linux Environment
MSE	Manage Security Environment
OID	Object Identifier
PAN	Primary Account Number
PIN	Personal Identification Number
PUK	unlocking password
PK – DH	Public key – Diffie Hellman (asymmetric key base algorithm)
PSO	Perform Security Operation
RFU	Reserved for Future Use
RSA	Rivest Shamir Adleman
SAL	Service Access Layer
SAL Lite	Service Access Layer Lite
SDO	Security Data Object
SCB	Security Condition Byte
SE	Security Environment
SEID	Security Environment Identifier byte
SM	Secure Messaging
SSD	Security Service Descriptor
TLV	Tag Length Value
UQB	Usage Qualifier Byte

## **5 ECC fitting in ISO/IEC 24727 model**

### **5.1 ISO/IEC 24727 main features**

This standardization initiative (ISO/JTC1/SC17 WG4/TF9) aims to design a new framework for interoperability based on a discovery mechanism of which the following paradigm:

The low level implementation features of the smart card, including proprietary specific features and characteristics based on ISO standards (i.e. ISO/IEC 7816-4) are hidden to the client-application through a high-level description provided to the terminal and processed by the middleware stack.

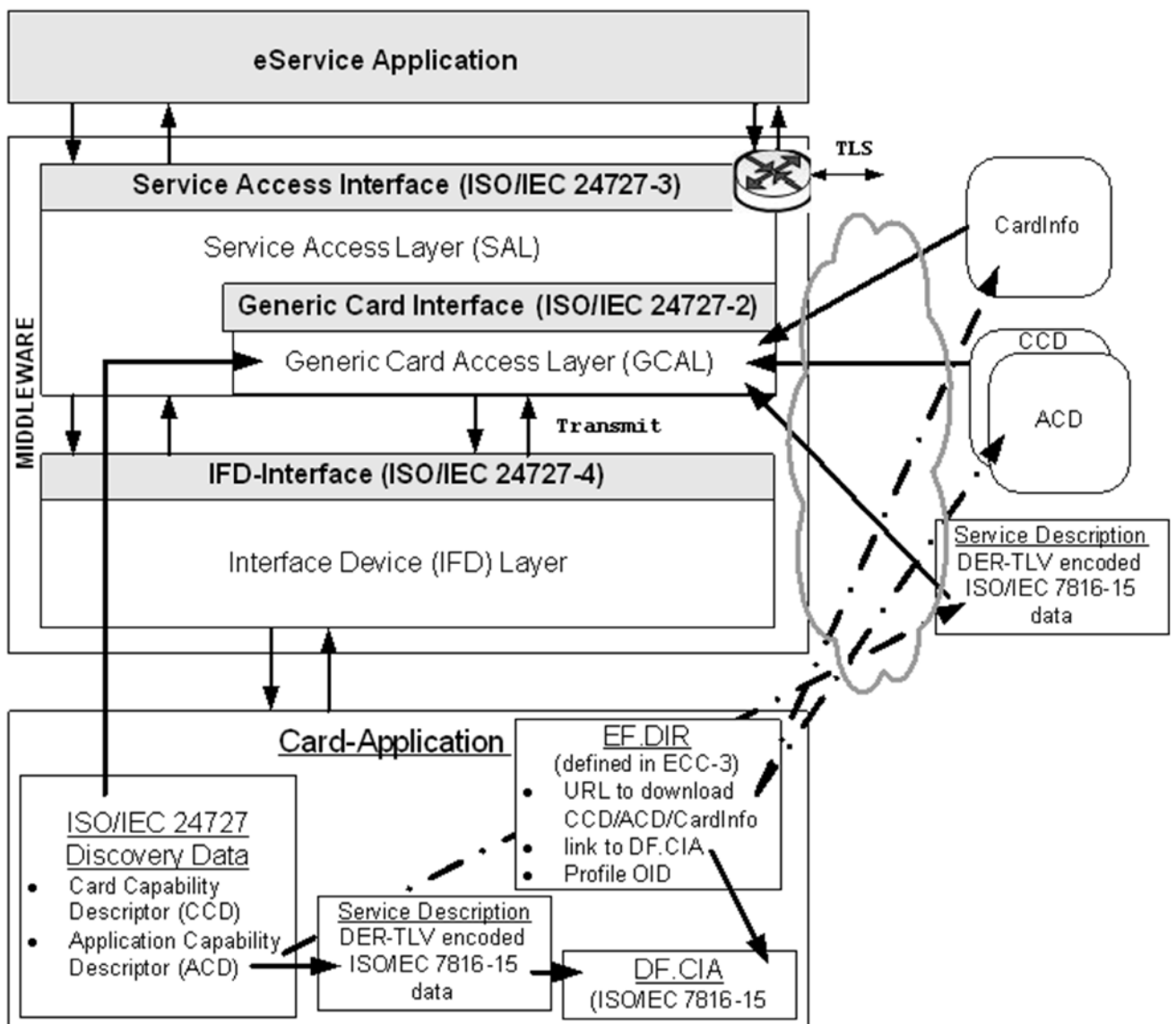
The middleware stack is defined by abstraction layers ensuring the interoperability. These layers are sustained by applicative components of which the implementation is up to the integrator and may vary according to the environment. For instance, the use of CAPICOM, CryptoAPI, CSP, proprietary API or DLL, ActiveX objects, Applets, PKCS#11 interface, JCE, M.U.S.C.L.E PC/SC emulation on Linux environment, etc, are all possible solutions upon which the ISO/IEC 24727 abstraction layers may run.

The middleware abstraction layers are of two kinds:

- The Service Access Layer (SAL) is in charge of interpreting the requests addressed by the client-application to the card via a high-level API (the SAL-API). The SAL translates the requests in terms of

sequences of APDU that are sent out to the underlying abstraction layer (the GCAL). This translation is performed according to the rules defined by a set of interoperability data reflecting the rules governing the card-applications. The SAL shall generate on the fly these interoperability data out of the ISO/IEC 7816-15 information available in the card. This ISO/IEC 7816-15 information is either provided within the CardApplicationServiceDescription data, or within the DF.CIA files, or both. Upon request from the eService, the SAL may surface the interoperability data to the eService. The SAL is specified in Part 3 of the ISO/IEC 24727 series.

- The Generic Card Access Layer (GCAL) is in charge of translating the APDU handed on by the SAL in terms of APDU understandable to the smart card. This translation is applied according to the rules defined in the ACD (Application Capability Descriptor) and/or in the CCD (Card Capability Descriptor) templates. These templates are read out of the card by the GCAL. The GCAL performs a bootstrap mechanism upon card detection in order to retrieve the ACD and CCD containers from the card. The bootstrap operation is the first step of the discovery mechanism. The GCAL functionality is specified in Part 2 of the ISO/IEC 24727 series.



**Figure 1 — CEN/TS 15480 compliant smart card in ISO/IEC 24727 framework**

The smart card hosts a set of interoperability data that are DER-TLV encoded according to an ASN.1 definition. This ASN.1 specification provided in ISO/IEC 24727-2 describes all the information items required

by a terminal to access the services offered by the card (resources, actions) and to allow as well the end-user to access services running on terminal side. This ASN.1 specification is built upon the computational model specified in ISO/IEC 24727-3. Therefore, the ASN.1 definition encompasses the descriptors of the services hosted on-card per application, the card application list, the access control rules applying to each on-card action or resource, and the description of each modular authentication item (called Differential-Identity or DID). This latter item (Figure 2 — Differential-Identity structure according to ISO/IEC 24727-3) is the modular descriptor of each authentication procedure. DID comprises five fields among which an optional element, the dIDQualifier, that conveys further information to clear ambiguities whenever required by cryptographic requests.

Element 1	Element 2	Element 3	Element 4	Element 5
DIDName	Authentication Protocol	Marker	Scope (Global or Local; implicit)	dIDQualifier
Mandatory	Mandatory	Mandatory		Optional

**Figure 2 — Differential-Identity structure according to ISO/IEC 24727-3**

The relationship between the elements constituting a DID is such as a Marker applied in a given Scope feeds the algorithm of a given Authentication Protocol that is applied to authenticate a given named DID.

The rules controlling the access to a card resource or authorising a card action are expressed has a boolean combination of Differential-Identities. If the boolean expression evaluates to “TRUE”, the requested card resources are made available or the card action is executed, otherwise the request is rejected.

The DifferentialIdentityQualifier may serve to convey to the SAL the OID related to a cryptographic operation or authentication protocol. The DIDQualifier type depends on the Authentication Protocol to which it applies. Details of DIDQualifiers are given in ISO/IEC 24727-3:2008, Annex A.

**5.2 General security issues – Applicable ISO/IEC 24727-4 Stack Configurations for the ECC environment**

The abstraction layers defined by ISO/IEC 24727 (all parts) are intended to hide seamlessly the implementation differences that may occur between smart cards from different vendors even when their respective implementation is based on the same standard or specification (i.e. CEN/TS 15480-2).

A high level description (according to ISO/IEC 24727 (all parts)) of the security rules governing the services hosted by the card allows an extension of the information provided by ISO/IEC 7816-15 implementation, ensuring thereby a more comprehensive interoperability protocol based on ISO/IEC 7816-15.

By personalising the necessary data sets for interoperability as per ISO/IEC 24727-2 and ISO/IEC 24727-3, a European Citizen Card can fit in the ISO/IEC 24727 framework with the following limitations:

- a) The GCAL being assumed to translate APDU handed on by the SAL, these command APDU cannot be secured (integrity, confidentiality) otherwise any change in the command header according to the ACD procedural elements will cause a rejection of the command by the smart card. Therefore, if the translation function of the GCAL is performed, end-to-end Secure Messaging cannot be applied between the client-application and the card-application in the ISO/IEC 24727 framework. With this respect, the European Citizen Card needs to discard the GCAL translation function except for use cases where no Secure Messaging is envisioned. This issue was solved by the first amendment to ISO/IEC 24727-4 as follows:
  - 1) Execution of Procedural Element (PE) functionality directly in the Service Access Layer implementation; the PE consume the Registry that is either XML-based CardInfo file or [ISO/IEC 7816-15]-based DER-TLV Data Object. The PE translate the SAL API calls into card-specific APDUs according the instructions brought by the Registry; consequently, the former process

in two stages i.e. first generation of generic APDU commands conforming to GCI, and second the translation by the GCAL of the commands so obtained into card-specific APDUs is no longer useful and is replaced by one-step process handled by PE with Registry input. The GCAL implementation becomes abstract and the PE is enabled for a variety of card managements through SAL API including Global Platform.

- b) The Loyal-stack configuration (Figure 3 — ISO/IEC 24727-4 excerpt: Loyal-Stack configuration), the Remote-ICC-stack configuration (Figure 4 — ISO/IEC 24727-4 excerpt: Remote-ICC-Stack configuration), and the Remote-Loyal-Stack configuration (Figure 5 — ISO/IEC 24727-4 excerpt: Remote-Loyal-Stack configuration) are considered in the present document. A generic depiction of ISO/IEC 24727 middleware stack lying over different platforms is represented on Figure 6 — ISO/IEC 24727-4 excerpt: Generic elements of ISO/IEC 24727 MW stack).
- c) The implementation of Part-3 Layer (SAL) and Part-2 Layer (GCAL), if any, on separate machines raises the communication channel security issue to bridge the two layers. The use of a TLS session in between to prevent eavesdropping, tampering or message forgery, raises further constraints:
  - 1) For the client-application to keep control on the intermediate TLS secured communication channels it requires a handle on the TLS secure session and it needs to share the secrets involved in the secure session establishment. Such control is not provided by ISO/IEC 24727-4, nevertheless a parameter denoting the available routes between the client-application and the card-application is backed up to the client-application for awareness then path selection. During the first phase of the SSL session, the client and server negotiate which cryptographic algorithms will be used. In typical use, only the server is authenticated while the client remains unauthenticated. For the purposes of mutual authentication, a public key infrastructure (PKI) deployment to clients is required.
  - 2) ISO/IEC 24727-3 and ISO/IEC 24727-4 do not provide to the client-application control over the secure messaging features and therefore require session keys sharing, which raises different security issues. It is preferable for the session keys and more generally for the host security module functions and keys to remain the property of the eService.
  - 3) SSL runs on layers beneath HTTP, SMTP and NNTP application protocols, and above TCP or UDP transport protocol. SSL can be exploited to add security to any protocol using reliable connections like TCP. Therefore, to avoid using separate ports for encrypted communications and to enable the application protocols to upgrade to TLS from a plaintext connection, the client should support SSL natively instead of relying on standalone SSL products.
- d) The implementation of client-application and Part 3 Layer (SAL) on separate machines entails securing the requests between the SAL Proxy and the SAL Agent on both sides, and requires the marshalled requests to be transported over TLS. The marshalling protocol may be based on DER-TLV encoding according ASN.1 definition and encapsulation into ENVELOPE APDU command, or on WSDL-based XML messages transported over SOAP binding. Both methods are described in ISO/IEC 24727-4.

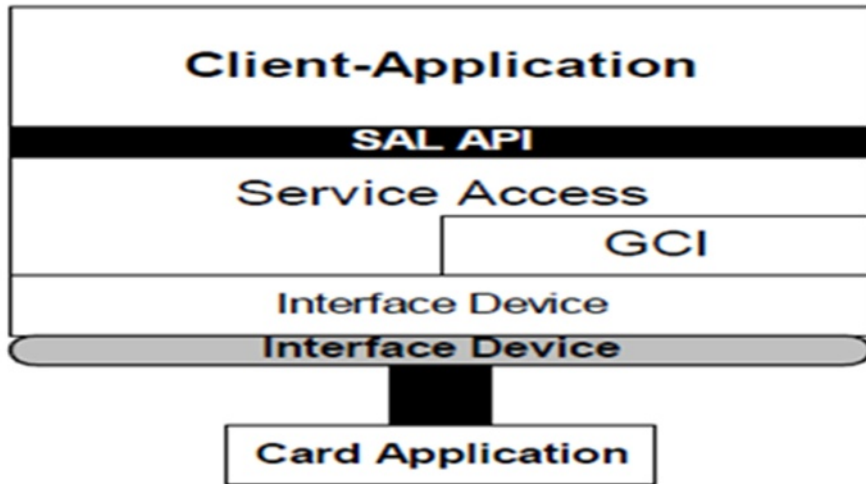


Figure 3 — ISO/IEC 24727-4 excerpt: Loyal-Stack configuration

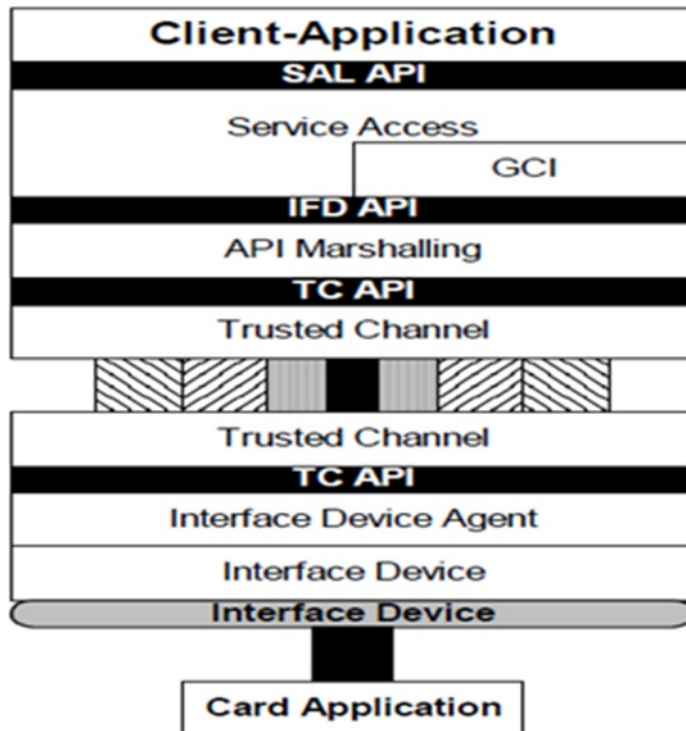


Figure 4 — ISO/IEC 24727-4 excerpt: Remote-ICC-Stack configuration



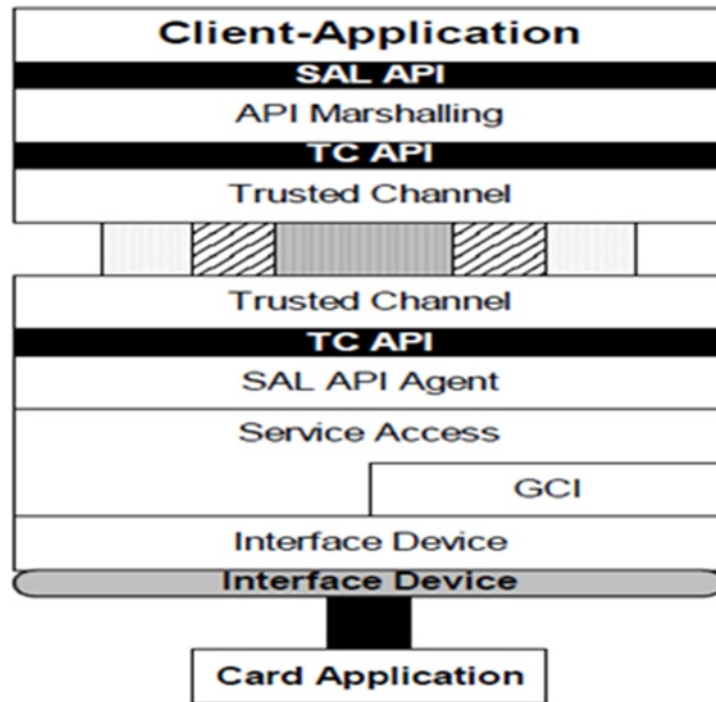


Figure 5 — ISO/IEC 24727-4 excerpt: Remote-Loyal-Stack configuration

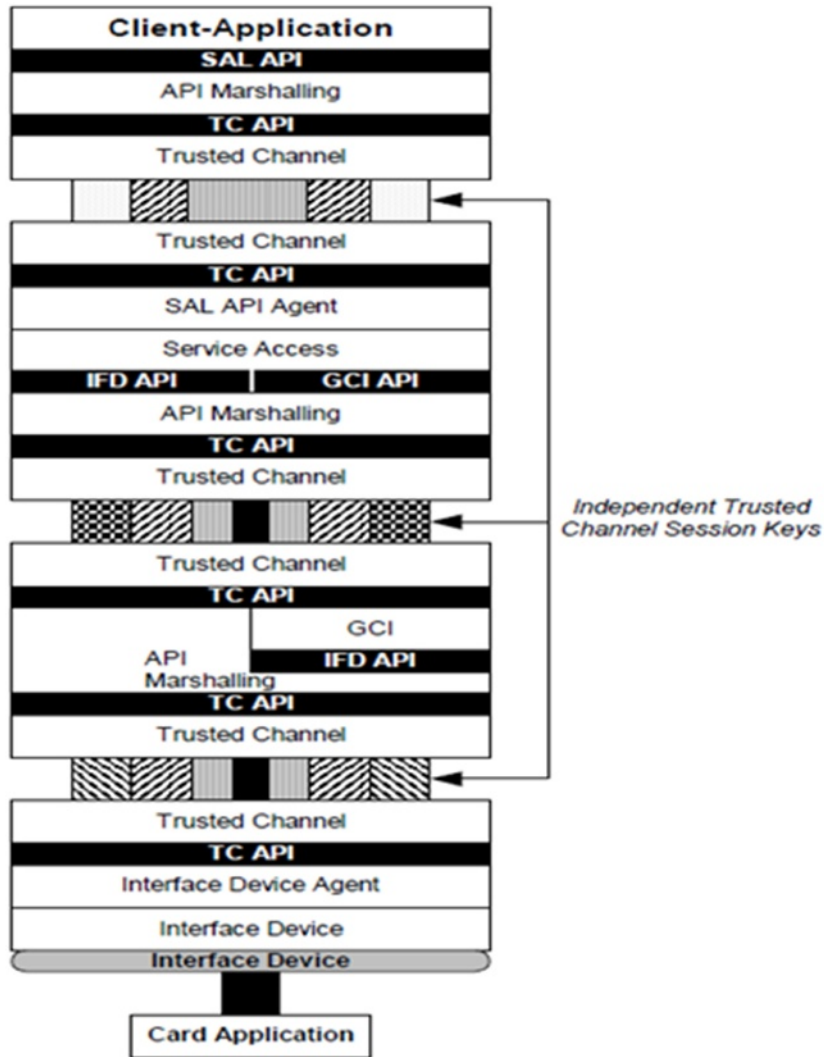
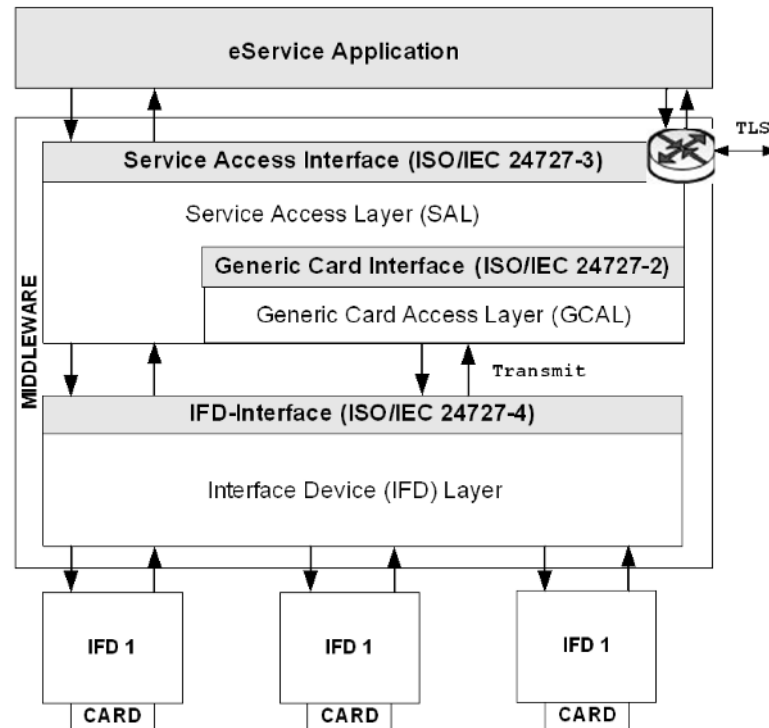


Figure 6 — ISO/IEC 24727-4 excerpt: Generic elements of ISO/IEC 24727 full-network stack

### 5.3 ECC-3 Middleware Architecture

#### 5.3.1 General

An overview of the architectural parts of ECC-3 is given in Figure 7 — ECC-3 Middleware Architecture. The ECC-3 Middleware consists of the Service Access Layer (SAL), either the Generic Card Access Layer (GCAL) or the Procedural Elements and the Interface Device Layer (IFD-API). A detailed description of the different parts of the ECC-3 Middleware can be found in the subsequent sections.



**Figure 7 — ECC-3 Middleware Architecture**

### 5.3.2 Service Access Layer (SAL)

The ECC-3 Service Access Layer is based on the Service Access Layer of ISO/IEC 24727-3.

### 5.3.3 Generic Card Access Layer (GCAL)

The Generic Card Access Layer is responsible for mapping the SAL functionality to card application specific functions and protocols. To allow an easy adaptation within the GCAL, the GCAL uses ACD, CCD and CardInfo files. ISO/IEC 24727-4 abstract the GCAL implementation and rely on the Procedural Elements hosted in Service Access Layer for the mapping of SAL API calls onto card-specific functions.

### 5.3.4 Interface Device Layer and API (IFD API)

The Interface Device API is responsible for the APDU based communication with the IFD / card application.

### 5.3.5 ECC-3 Stack Distribution and Connection Handling

#### 5.3.5.1 General

If the eService is located on a server somewhere on the network and the ECC is used by a user at home, the Remote-ICC-Stack, the Remote-Loyal-Stack, or the Opaque-ICC-Stack (as per ISO/IEC 24727-4), further to the Loyal-Stack configuration of ISO/IEC 24727-4, will be used within the ECC-3 middleware architecture. Within this distributed system it is necessary that the eService will communicate directly with the card application. Additionally, the service embedded application on user machine may perform an authentication process with the card application or may interact by other means with the card application. APDUs generated at service side in a Remote-ICC-Stack like layout may be addressed directly to the card through the IFD-API, and marshalled requests generated at server side in a Remote-Loyal-Stack like layout may be addressed to the SAL implementation on user machine. This implies that the complete ECC-3 middleware stack shall be available on the user machine (client machine) and on the server side (see Figure 8 — ECC-3 Middleware Architecture in the Remote-ICC-Stack environment), and that a mix of different ISO/IEC 24727 configurations shall be deployed together to achieve electronic service delivery in the real life.

### **5.3.5.2 ICC-Resident-Stack specifics**

The present standard aims at providing guidelines for legacy card support to facilitate migration towards ECC-compliant cards. In this view, the ICC-Resident-Stack (ISO/IEC 24727-4:2008, 5.5) is a particular configuration such as the service layer processing the high-level calls from the eService application is embedded onto the card: The card-application emulating stack has an interface APDU exposing the ENVELOPE command. The eService application calls the SAL-API requests; the SAL-API requests are marshalled by a proxy according a transmission protocol based on ASN.1 (ASN.1-Binding) as follows: the SAL-API calls are encoded in DER-TLV. The DER-TLV payload is encapsulated within the data field of the ENVELOPE APDU command and sent out to the card. Alternatively, the marshalling protocol may use XML-Binding.

Upon reception of the ENVELOPE command, the applet parses it's data field through a DER-TLV encoder/decoder function or alternatively an XML-parser function. The result of the parsing is processed by the card-application according to ISO/IEC 24727 business logic.

One outstanding consequence of this implementation is that the card-application hosts directly the data structures employed by the ISO/IEC 24727 business logic. Usually for the rest of ISO/IEC 24727 configurations (except ICC-Resident-stack), a discovery mechanism takes place to translate the card-application services and related security rules in terms of logical data structures (DataSet, DSI, DID, ACL, CardApplicationList, CardApplicationServiceList, etc.): this *discovery mechanism* allows the middleware service layer (off-card) to discover the card function through a specific service-oriented language (based on ISO/IEC 7816-15). At initialisation time, the card delivers a container to the middleware service layer. This bootstrap procedure provides the middleware with the ACD and CCD containers. The middleware service layer gets and parses this contents and generates out of it the logical data structures according to ISO/IEC 24727 business logic. Only these resulting data structures for interoperability are exposed to the eService application upon request. The eService application (Client-Application) does not need to handle APDU (the ENVELOPE command being generated by the eService-Proxy layer).

The ICC-Resident-Stack is embedding the middleware service layer and therefore does not require ACD anymore. Only a CCD informing the middleware that the card conforms to the ICC-Resident-stack shall be available on-card. Consequently, this configuration spares the encoding and personalisation of ACD on the card.

This configuration has the advantage of facilitating the post-issuance creation/personalisation of data structures onto the card: the ISO/IEC 24727 data structures are directly hosted on-card as DER-TLV encoded Data Object. Personalising ISO/IEC 24727-compliant cards in post-issuance raises the issue of formatting properly the structures to be created: i.e a DataSet shall be interpreted as a DF, EF or DO; a DSI shall be interpreted as a EF or DO; the EF shall be of a type (record, transparent, TLV, etc) that shall be determined according the card features; those translations that are likely to make post-issuance personalisation complicate in common cases (see Annex C) are dispensed with for ICC-Resident-Stack.

The drawback of this configuration is mainly the lack of support for legacy: while existing cards in the field can become ISO/IEC 24727-enabled by simple supply of their respective ACD/CCD, the ICC-Resident-stack embeds a newly designed/issued card-application and may be convenient for newly issued cards, without need for legacy requirement.

### **5.3.5.3 SAL Lite component specifics**

In its first amendment, ISO/IEC 24727-4 design a new component derived from SAL functionality and dedicated ONLY to linking the entities defined at the Service Access Layer API (e.g. Differential Identity, DataSets, DSI) with actual on-card entities such as keys, files, Data Objects and Access Control Rules. SAL Lite is enabled with Registry processing capability, For instance, SAL Lite may retrieve the [ISO/IEC 7816-15]-based Registry out of the card, interpret it and derive from it the data structures representing the objects handled by the SAL API in conformance with ISO/IEC 24727-3. The main objective for SAL Lite component is to alleviate the workload of the client-application platform by supplying it upon request with pre-built data structures for interoperability.

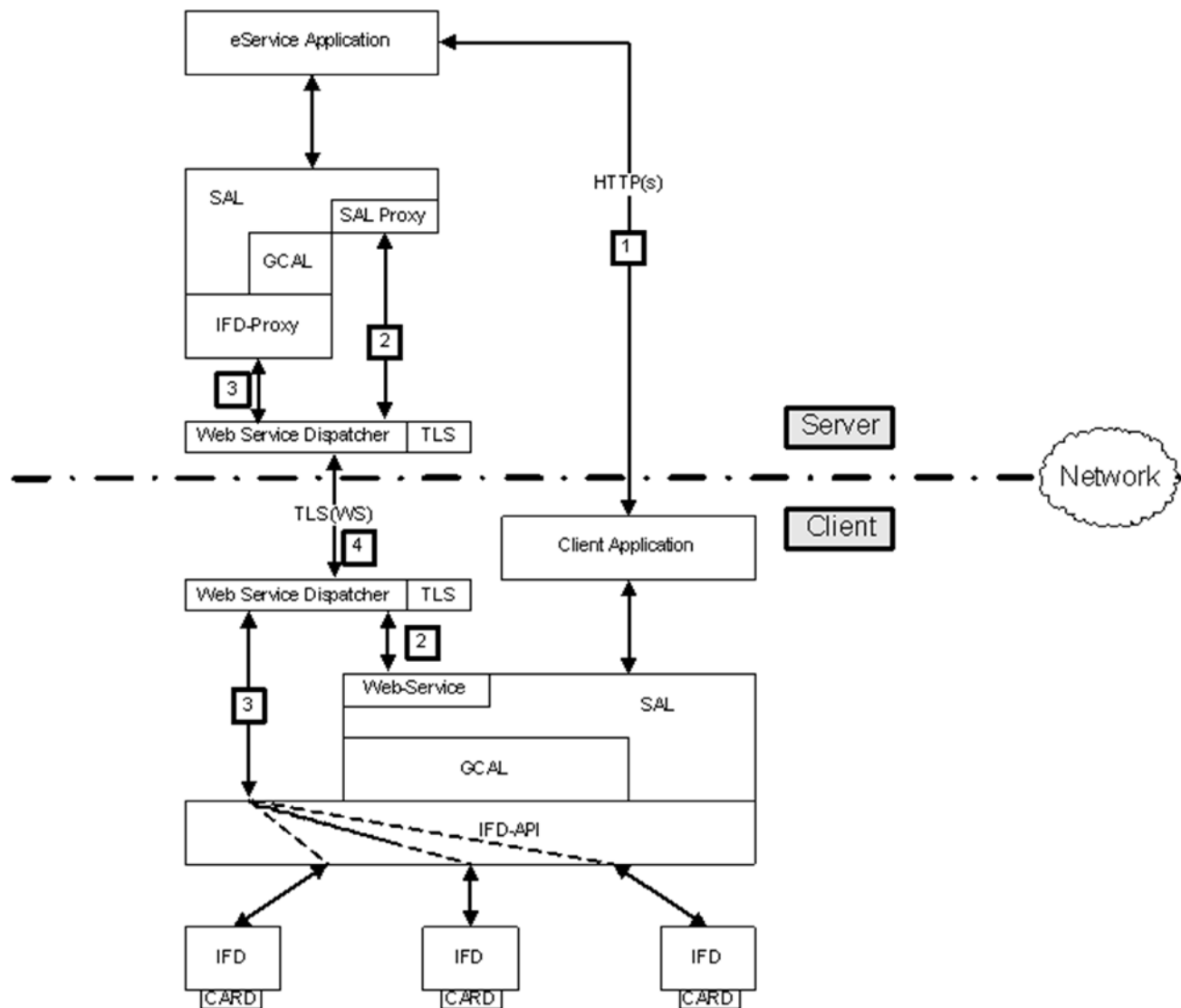
The SAL-Lite interface provides a subset of the full ISO/IEC 24727-3 API comprised of the following requests:

- CardApplicationConnect(),
- CardApplicationList(),
- CardApplicationServiceList(),
- CardApplicationServiceDescribe(),
- DataSetList(),
- DataSetSelect(),
- DSIList(),
- DSIRead(),
- DIDList(),
- DIDGet(),
- ACLList()

The distributed middleware configurations within which SAL Lite component fits appropriately are the following: Full-Network-Stack, Opaque-ICC-Stack and Remote-ICC-Stack.

**5.3.6 Multi-stack composed configuration**

**5.3.6.1 General**



**Figure 8 — ECC-3 Middleware Architecture in the Remote-ICC-Stack environment**

If the user wants to use an eService, the communication will be started by opening a secure channel from the client application to the eService 1 . To allow a communication based on high level commands (SAL), the connection 2 will be established. Connection 3 is responsible for the APDU based communication between the eService and the card application. This connection allows the end-to-end authentication and secure channel between the card application and the eService. Finally, connection 4 will be used to transport the web service requests and responses between the client and the server.

The three different connections are necessary, since the communication end points for each communication channel are different.

Since connection 1 is application specific, it is out of scope for this specification. The connection handling for connection 2, 3 and 4 are described in here after. Due to the usual firewall restrictions, these connection handling methods are designed such as the client will start the connection establishment.

### 5.3.6.2 Connection handling for the distributed stack components

#### 5.3.6.2.1 General

As the ECC specification uses web service based communication instead of the trusted channel API as defined in ISO/IEC 24727-4 it is necessary to specify an analogous XML-based binding for the SAL functions. This binding addresses the *connection establishment between distributed systems*.

Some of the protocols specified here involve two SAL-instances, which run on different systems (eService and Client) and communicate with each other via potentially insecure networks. Therefore, the relevant aspects of security shall be taken into consideration when setting up the transport channel.

#### 5.3.6.2.2 General security requirements

The security requirements of ISO/IEC 24727 (all parts) 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 MAY be used, whereby the associated private keys are to be adequately protected. Alternatively, anonymous TLS cipher suites, such as `TLS_DH_anon` from RFC 4346 or `TLS_ECDH_anon` from RFC 4492, MAY 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 is typically 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 is required for every connection to a card application for communication to take place between the identity layer or the application layer and the SAL.

#### 5.3.6.2.3 Connections for SOAP binding

When using the SOAP binding SOAP-v1.1, the connection is 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.

#### 5.3.6.2.4 Connections for PAOS binding

When using the PAOS binding PAOS-v1.1, however, a more complex process is 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.

The general connection sequence is shown in Figure 9.





end user's browser by the means of embedded JavaScript. The IFD-API call is unmarshalled and the APDU commands are extracted from the payload and handled by the embedded script that relies on the local library script extension. This library script wraps the IFD layer. The APDU responses from the ECC are managed the other way around and returned to the SOAP server.

For security purposes, before to grant access to its resources, the smartcard application may authenticate the Server. Secure APDUs may be conveyed as part of the SOAP payload across the network between the web service application and the end user browser as described above. ECC authentication mechanisms can take place in this framework without restriction.

The code that executes in the client (i.e javascript) makes call to the server side code to either get mor einformation from it or to submit infoprmtion to it. This call is typically done by XMLHttpRequest object which is built into all the browsers. This object facilitates the connectivity between the javascript and the server in a browser/server platform/application agnostic way. Some code executing in javascript tells this object to transmit data to a server end point. The XMLHttpRequest is capable of transmitting and receiving not only XML-based data but any text generated from javascript. In case of processing overhead for the code (javascript) that executes in the browser (i.e preliminary XML parsing and creation of and object tree for incoming data), XML payload may be replaced with the Javascript Native Object Notation (JSON) offering similar capabilities as that of XML. Therefore, JSON may alternatively be used to exchange data between javascript and the server side code with the following benefits:

- programming language-independent;
- human readable;
- presenting a structured view of data;
- less verbose than XML whereby saving the bandwidth;
- transformable syntax to a an object programming language without need for parsing.

In case of use, JSON shall be mapped onto the XML definitions provided in this standard.

The table below shows the main software components and related layout realizing the connectivity solution described in this clause.

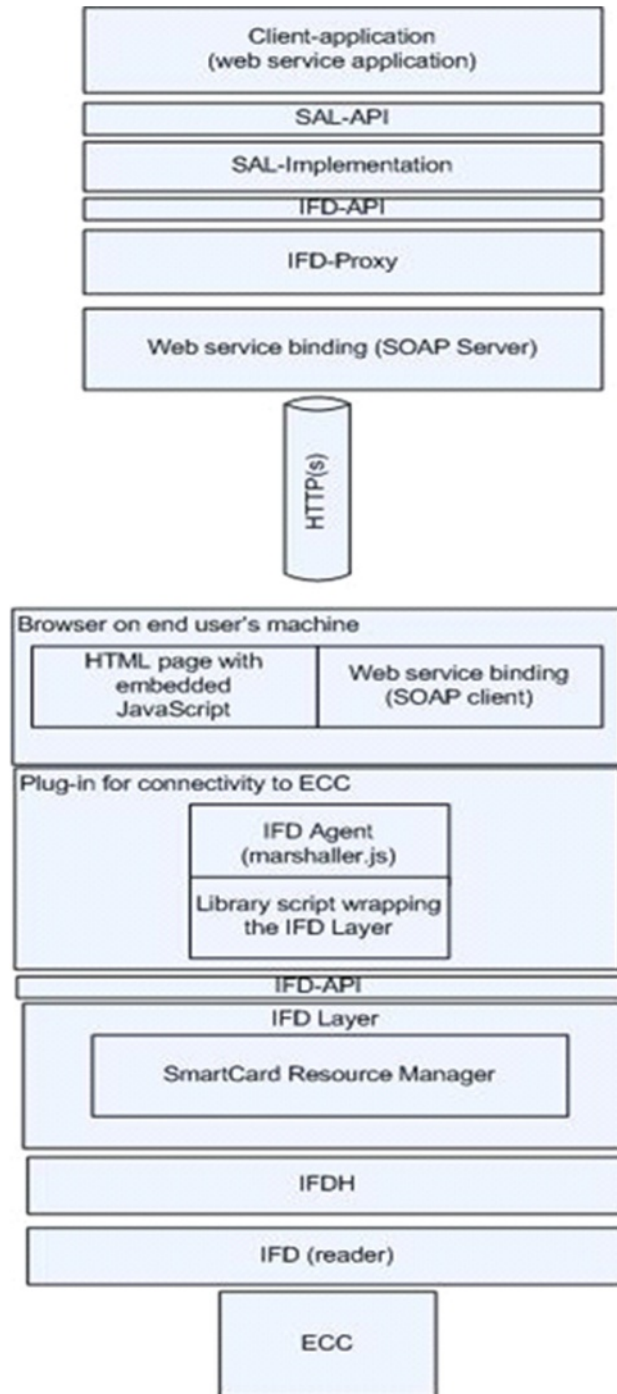
**Table 1 — Remote-ICC-Stack configuration as a web service**

Software components							
ISO/IEC 24727-4 Remote-ICC-Stack		End-user's machine		Web server		example of implementation	
1	Client-application			M		Web service running on application server Tomcat	
2	SAL-API			M		Acc. ISO/IEC 24727-3	
3	SAL-Implementation Layer			M		Acc. ECC-3	
4	Generic Card Interface	IFD-API		O	M	Based on Annex B	
5	IFD-Proxy			M		Marshalling acc. chapter 8	
6	WSDL Binding: Based on SOAP protocol over HTTP <sup>4)</sup>			M		Web app. server Apache SOAP.	
						JavaMail package (mail.jar)	SOAP package (soap.jar, soap.war)
						Java Activation Framework package (activation.jar)	Xerces package (XML parser) (xerces.jar)
						Application server Tomcat	
						Java VM	
7	SOAP request/response over http(s)		/		/		SSL session
8	BROWSER		M			(IE, Firefox or other)	
9	<b>Connectivity solution</b>	IFD-Agent: XML parser script acting as a < Marshaller.js >	M			Embedded JavaScript, java applet or/and JSP, ActiveX	
10		script accessing smart card (.JS script): PC/SC based	M			Embedded JavaScript	
11		Library script wrapping the IFD layer	M			Plug-in loaded securely from the web server	
12	IFD-Layer		M			A.2	
13	Smart Card Resource Manager		M				
14	Reader handler (driver)		M			PC/SC compatible IFDH	
15	Reader	Enhanced	M	O		IFD-API is	

4) Alternatively, XML may be translated into JSON.

Software components							
ISO/IEC 24727-4 Remote-ICC-Stack		End-user's machine	Web server	example of implementation			
		reader					recommended for enhanced readers
16	ECC		M				ECC with Application Discovery profile {OID = 1.3.162.15480.4.0.1. }
17	ACD/CCD or [ISO/IEC 7816-15]-based Registry for the discovery mechanism either hosted on-card or loaded form remote for legacy support		O				CardApplicationService-Description value acc. ECC-3: DER-TLV encoded information representing the card-application resources for a given eService.

**5.3.7.3 Web service ECC-3 Middleware architecture**



**Figure 10 — Middleware architecture based on smartcard connectivity solution**

The main advantages of this alternative are as follows:

- no installation required on end user's machine except the connectivity plug-in;
- light footprint scripts on end user's machine for the Browser extension;

- high level of mobility and interoperability: connection to the ISO/IEC 24727-enabled web service from any PC;
- interoperability of transaction based on IFD-Web Service Binding (Clause 8).

### 5.3.8 XML-based SAL interface

The present document specifies a new function available to the Client-Application: 'ExecuteSAL' defined as a single entry point to spare ASN.1 encoding/decoding to the client-application and to convey a XML-based payload instead. This XML-based payload is conform to the SAL-interface defined in ISO/IEC 24727-3 and may be used in Web Service based scenarios or with the ExecuteSAL function call.

The XML-Schema definitions for Service Access Layer functions are provided in Clause 10. The C-Language binding for the ExecuteSAL function is provided in Annex E. The Java-Language binding for the ExecuteSAL function is provided in Annex F.

To ensure conformance of Authentication Protocols to the ISO/IEC 24727 environment, a template for protocol metadata definition is provided in ISO/IEC 24727-3:2008, Annex A and its related Amendment ISO/IEC 24727-3:2008/DAmD 1. In ISO/IEC 24727-3:2008, Annex A, the authentication protocols binding is ASN.1 based, therefore an additional XML-Binding for these authentication protocols is required in order to make them fit with the ExecuteSAL function. Smart card profile fitting with ECC-3 stack

The figure below describes how ECC-compliant cards with profiles according ECC-4 can fit in ECC-3 middleware stack. Legacy interfaces as PKCS#11 or CryptoAPI are considered respectively for PKCS#11 enabled applications or Crypto API enabled applications. See ISO/IEC 24727-4:2008, Annex J (informative) for further details on CryptoAPI Access via Procedural Elements.

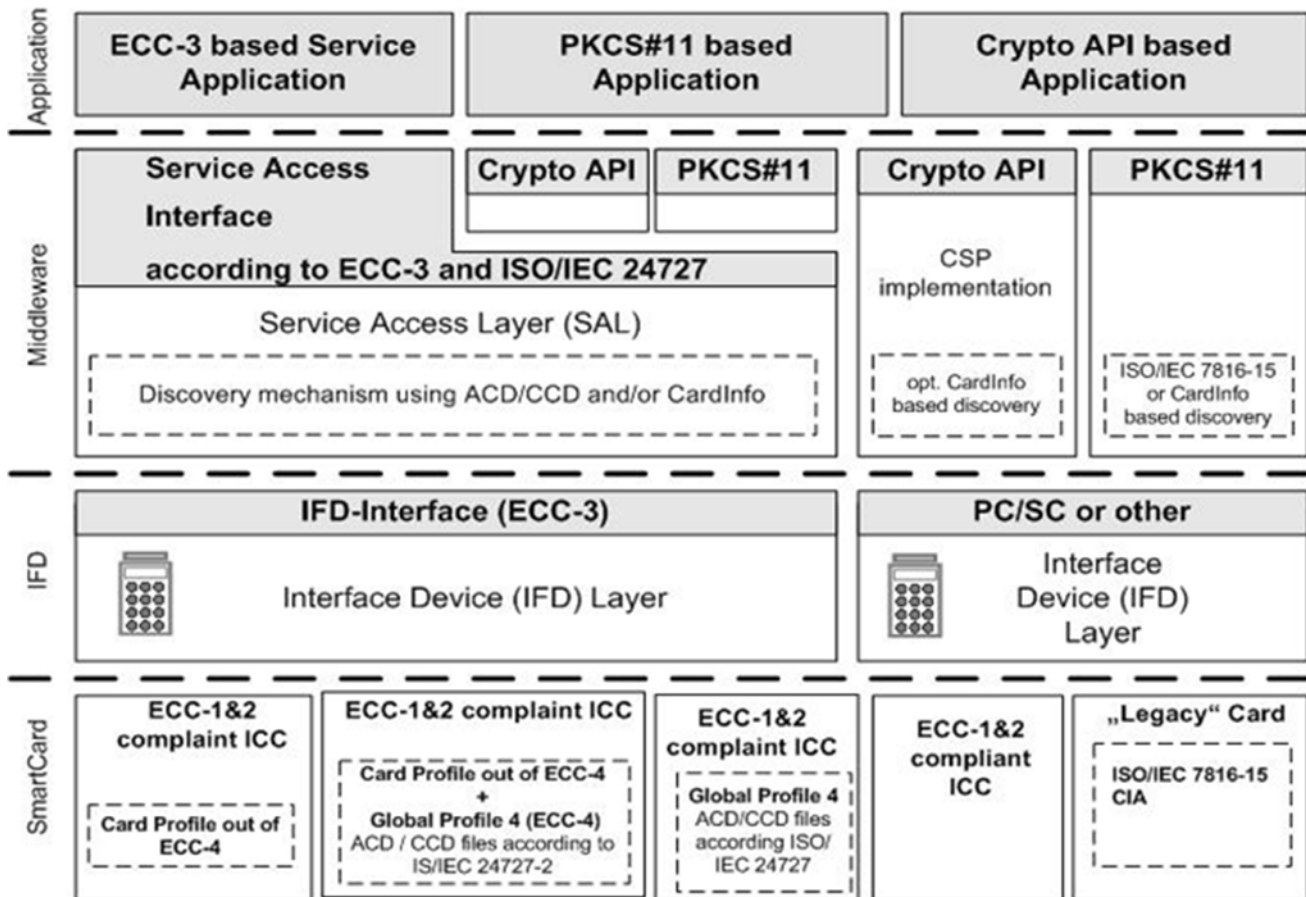


Figure 11 — ECC-compliant and legacy cards fitting with ECC-3 middleware stack

## 6 Card Discovery Mechanisms

### 6.1 General

The ECC-3 middleware supports multiple characteristics of ECCs. To adapt the SAL commands, the protocols and the data format to the card, the data structures described in 6.4 are used. If none of these data structures is present, the ECC middleware offers a mechanism to identify the card and to use configuration files for the command, data and protocol mapping. For identifying the card, different card specific data like AIDs, ACD, CCD and ISO/IEC 7816-15 files are used. A possible inspection activity chain is visible in Figure 12. To optimize the card recognition time, the order of the investigation can depend on the ATR or other optimization methods of the middleware.

## 6.2 Discovery decision tree

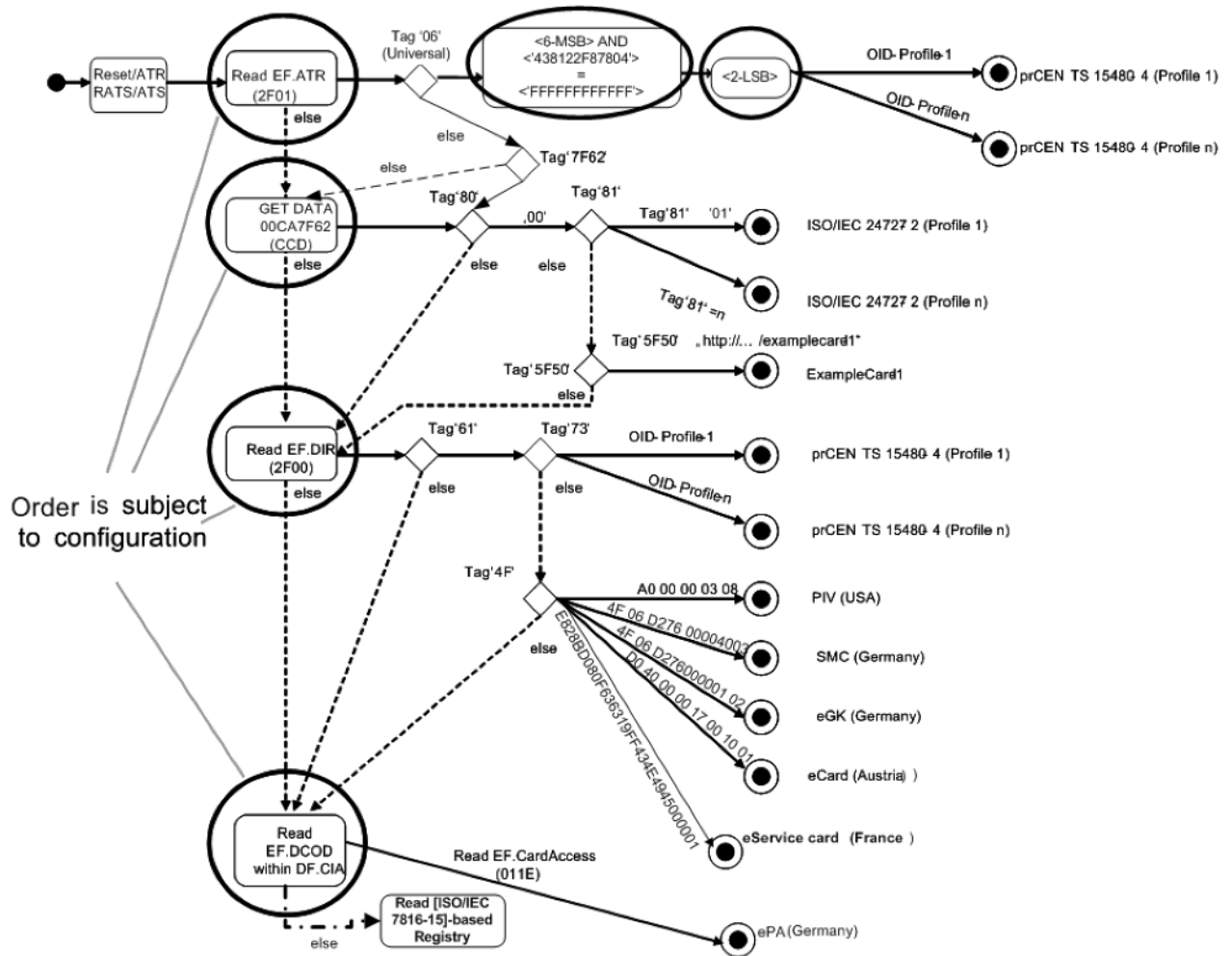


Figure 12 — Discovery decision tree

## 6.3 Migration path towards ECC and provision for legacy cards

### 6.3.1 General

The present document describes the requirements for ECC token to fit into the ISO/IEC 24727 framework and provides recommendations as to how other cards may be supported. These requirements bear upon the encoding and personalisation of some data that are to be made available to the terminal during the bootstrap procedure. According to ISO/IEC 24727-2, this procedure requires the presence of CCD and ACD containers in the card. As ISO/IEC 24727-non-compliant ID-cards and other proprietary cards are already deployed in the field, their personalisation in post-issuance with additional data (i.e CCD/ACD) might be very costly or impossible. The present document provides recommendations to allow for support of “legacy cards”.

For this purpose it is necessary that the middleware:

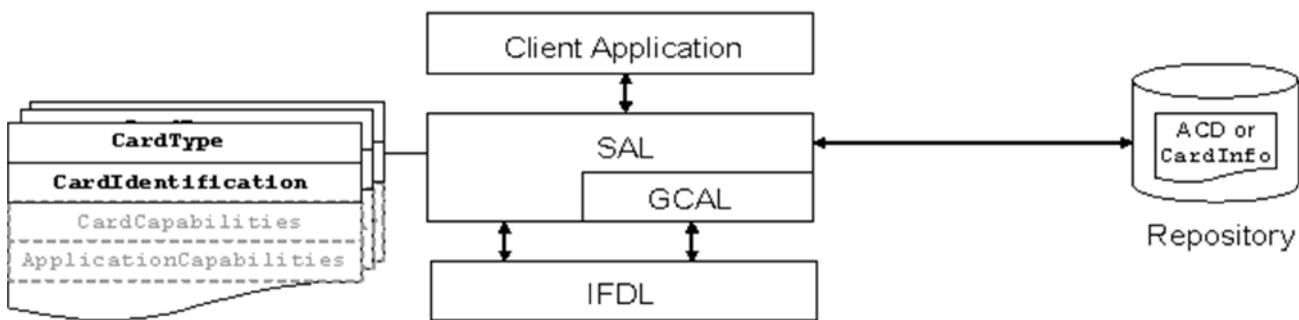
- first takes provisions to recognize the type of a presented card; and
- then obtains the CardInfo-file or the [ISO/IEC 7816-15]-based Registry, which is required for the mapping of generic SAL-requests to card-specific APDUs by other means.

In order to implement the first step there are two alternatives:

- If the card contains a minimum CCD, the SAL will retrieve the full *CardInfo*-file or the full [ISO/IEC 7816-15]-based Registry at the address (URL) indicated in data object '5F50' of the CCD.
- In order to support card which does not contain such a minimum CCD, the SAL shall be equipped with minimal XML-based *CardInfo*-files (cf. chapter 9) or alternatively with ASN.1 based Card Discovery module (ref. ISO/IEC 24727-2:2008, 6.5 on Discovery Mechanism for legacy cards), which just contain enough information to build the decision tree depicted in Figure 12. For each legacy card, which is to be supported, the *CardType*- and *CardIdentification*-element shall be described. Within the *CardType*-element there may be a *CardInfoRepository*-element, which provides the address (URL) of a repository.

After this procedure, the type of the card and/or the URL, which allows retrieving the *CardInfo*-file have been determined. There are the following possibilities to implement this second step:

- The SAL retrieves ASN.1-based ACD- and CCD-files from a remote server.
- The SAL uses the *CardCapabilities*- and *ApplicationCapabilities*-elements of the locally available XML-based *CardInfo*-file, which was used to build the decision tree in the first step.
- The SAL retrieves a full XML-based *CardInfo*-file from a remote server.



**Figure 13 — Local or remote retrieval of ACD-like information**

### 6.3.2 Interoperable access to the Repository

To ensure interoperable access from the SAL-implementation layer to the remote repository hosting ACD of *CardInfo* file containers, an XML-based transmission protocol is defined. The related Web Service Binding protocol is SOAP-based.

### 6.4 Set of data for interoperability

This subclause defines the standard data structures to be personalised on the ECC card for compliance with ISO/IEC 24724 (all parts). These data structures are intended to hide the low-level implementation features to the client-application. This section will refer to:

- application Capability and Card Capability descriptors;
- ISO/IEC 7816-15 implementation including EF.DIR;
- any other information container located either on-card or off-card to achieve the fit with ISO/IEC 24727 framework (i.e. *CardInfo*-file, other).



## 6.5 Application and Card Capability Descriptors

This describes the options required to encode ACD and CCD for ECC-compliant cards are assumed to host both CCD and as many ACD as interoperable application present on card. For each card-application which is not made available to an ISO/IEC 24727-compliant terminal, no ACD is required.

**Table 2 — Card Capability Descriptor options for ECC-compliant card**

Requirements for ECC-compliant cards				Comments
CCD with DO '7F62'				
Data Object	Tag	description	value	
PRO	'80'	ISO/IEC 24727-2 profile	'00'	Current version of ISO/IEC 24727-2 with which this CCD complies
SAID	'A0'	AID list	AIDs or empty	all card-applications made available to the eService Client-Application via the SAL-API shall be referenced here as concatenated DO'4F'
LANG	'A1'	Procedural element description template	not used	The mandatory set of APDU of ECC Part2 does not cover the whole set of commands of the Generic Card Interface (GCI) defined in ISO/IEC 24727-2. GCAL shall be abstracted by Procedural Elements (PE) running in SAL implementation and a Registry shall be available and consumed by PE.
LANG-URL	'5F50'	URL to the full Registry (either CardInfo-file or DER-TLV encoded ISO/IEC 7816-15 definition)	URL	<ul style="list-style-type: none"> <li>the address of the CardInfo-file or the [ISO/IEC 7816-15]-based resource is present within a minimum CCD. SAL can retrieve the Registry from a repository through the API described in 6.2 and related subsections.</li> </ul>

Requirements for ECC-compliant cards				Comments
CIA-PROFILES	'81'	CIA profiles present on GCI	value field = '00'	<p>Bit 1 set to 0 meaning that a profile is present but does not conform to profile in ISO/IEC 24727-2:2008, Annex A, i.e iso7816DO are used in DCOD.</p> <p>the card may implement a DF.CIA for the needs of the discovery mechanism (i.e CIOChoice values are encoded as PathOrObject. If encoded as Path, CIOChoice items are nested in DF.CIA files, otherwise, they are encoded as Objects within the CardApplicationServiceDescription value). The DF.CIA may serve for further purposes not related to compatibility with ISO/IEC 24727 framework.</p>
CIA-PROFILES	'82'	CIA profiles present on GCI.	Not used	Optional
DIGITAL-SIGNATURE-CODE	'5F3D'	Digital signature information for procedural elements	Data object of digital signature block	Depending on security policy
IF-PROFILE	'83'	Profile of ISO/IEC 24727-3 interface	Not used	ECC card does not support ISO/IEC 24727-3 interface (unless it is an HTTP-enabled card which is out of ECC scope)

**Table 3 — Application Capability Descriptor options for ECC-compliant card**

Requirements for ECC-compliant cards				Comments
ACD with DO '7F63' (may be empty)				
Data Object	Tag	description	value	
LANG	'A1'	Procedural element description template	Optional	if used, the template shall refer to the standard describing the procedure language of the Registry (processing either [ISO/IEC 7816-15]-based DER-TLV or XML)
LANG-URL	'5F50'	URL of the translation code	pointer to the Registry hosted either locally or in a remote repository	See 6.3 The Registry comprises card data structures and services descriptors, as well as translation code. Therefore DO'5F50' contains the same information as within DO'7F66' plus translation code.
SERVICE-DESCRIPTION	• '7F66'	Services supported by the card-application	Used only if DO'5F50' is absent. Either XML-based CardInfo or DER-TLV encoded CIAInfo and CIOChoice values according ISO/IEC 7816-15	discovery data are generated on the fly by the SAL out of the '7F66' data object content. The actual Registry is encoded within the value field of this Data Object.
SERVICE-DESCRIPTION-LOCATION	'7F67'	Location of the '7F66' data objects	Used only if DO'5F50' is absent. URL value	In case the '7F66' data object content is stored on a remote server, the '7F67' data object shall provide the server resource location. In case the ACD or CCD are not present onto the card and are to be loaded from remote or provided by other means, they shall be complete
DIGITAL-SIGNATURE-ON-CODE	'5F3D'	Digital signature information for procedural elements	Data object of digital signature block	. Depending on security policy

## 6.6 ISO/IEC 7816-15 implementation

### 6.6.1 General

This subclause describes the representation of ISO/IEC 24727-3 Card-Application discovery information as an ISO/IEC 7816-15 Cryptographic Information Application.

The ISO/IEC 7816-15 representation of an ISO/IEC 24727 card-application contains all the information that the ISO/IEC 24727-2, ISO/IEC 24727-3 implementations need to realize the ISO/IEC 24727-specified interoperable connection between the client-application and the card-application. Data structure for interoperability are discovery information made available to the SAL by the card through the bootstrap mechanism, unless it is provided by other means. The SAL shall undertake the processing of the discovery information in order to generate on the fly the data structures defined in ISO/IEC 24727-3 as Access Control List, Differential-Identity, Data-Set and Data Structures for Interoperability (DSI), and Card-Application Services and Actions. These data structures shall be surfaced to the eService upon request through the SAL-API.

To allow for coding of access control rules conforming to ISO/IEC 7816-15 implementation, a mapping of SAL-API actions onto ISO/IEC 7816-15 *accessMode* attributes is described in this subclause. This mapping is laid on the extension of *accessMode* according the second amendment of ISO/IEC 7816-15.

The information in the ISO/IEC 7816-15 representation of an ISO/IEC 24727 card-application is what is referred to informally in ISO/IEC 24727-1 as the *discovery information*.

### 6.6.2 Profile designation within EF.DIR

Whenever a ECC implements a Profile as per CEN/TS 15480-4, it shall conform to the recommendation of this subclause.

EF.DIR shall host the Profile Object Identifier assigned by ECC-4 rules. Data object '73' shall encapsulate Profile OID within the application template (DO '61'). Accordingly bit6 of first byte within *card service data* in historical bytes shall be set to '1' to denote that BER-TLV DO is present in EF.DIR.

The OID branch is:

OID = 1.3.162.15480.4.x.y with x.y denoting the Profile number

For instance, DER-TLV coding with **x** = 0 and **y** = 1 (Global Profile n°1, for instance Application Discovery Profile) valuates to 0x43 81 22 F8 78 04 **00 01**, therefore the Data Object denoting the Profile shall be 06 08 43 81 22 F8 78 04 00 01

The figure below shows an example of how a Profile shall be encoded in a EF.DIR.

Tag	L	Description	Value
61	var	Application template (for ADF CIA)	
		<b>Tag</b>	<b>L</b>
		<b>Description</b>	
4F	var	AID of CIA	0xEB28BD08F A0000000xxxxxxxxxx
50	03	Application Label "CIA"	0x43 49 41 («CIA»)
		<b>Tag</b>	<b>L</b>
		<b>Description</b>	<b>Card Profile OBJECT IDENTIFIER (Optional)</b>
73	var	CIODDO Discretionary data object	0x06 08 43 81 22 F8 78 04 00 02

Tag	L	Description	Value
61	var	Application template (for Profile ADF called 'IAS')	
		<b>Tag</b>	<b>L</b>
		<b>Description</b>	
4F	var	AID of ECC-4 application	0xA0000000xxxxxxxxxx
50	03	Application Label "IAS"	0x49 41 53 («IAS»)
51	02	FID	0x5F00
		<b>Tag</b>	<b>L</b>
		<b>Description</b>	<b>Card Profile OBJECT IDENTIFIER (opt.)</b>
73	Var	CIODDO Discretionary data object	0x06 08 43 81 22 F8 78 04 00 02

**Figure 14 — Example of EF.DIR content**

### 6.6.3 ISO/IEC 24727-3 data structures mapping

The data structures that may be surfaced to the client-application upon request are DataSet, DSI, ACL, Differential-Identities, list of Card-Application Services, list of Differential-Identities.

The SAL shall generate these data structures out of the information available in CardApplicationServiceDescription defined as an ASN.1 SEQUENCE of ISO/IEC 7816-15 CIAInfo value along with consecutive ISO/IEC 7816-15 CIOChoice(s) value(s).

Example of CardApplicationServiceDescription value implementation applying ISO/IEC 24727-2 rules is provided in ISO/IEC 24727-4:2008, Annex G.

The interpretation of CardApplicationServiceDescription value in terms of ACL, Differential-Identities, Services, DataSet or DSI, is based on the mapping described in ISO/IEC 24727-4:2008, Clause 8 Registry implementations.

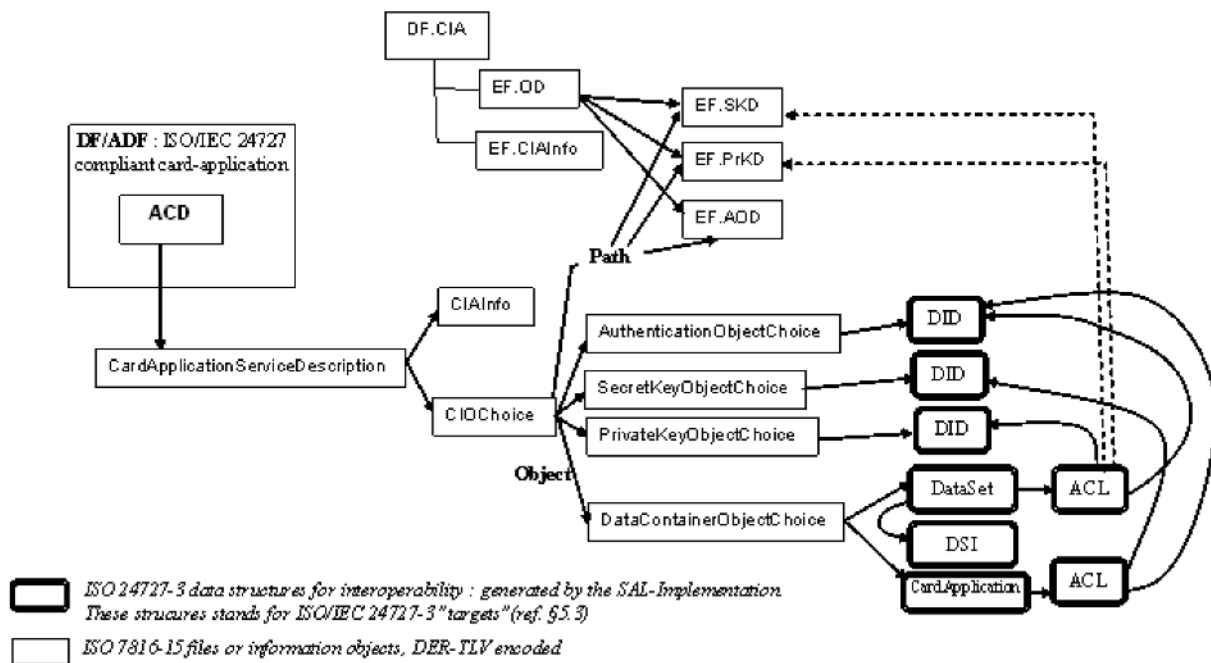
### 6.6.4 ISO/IEC 24727-3 data structures storage onto the card

According to ISO/IEC 24727-2, the CardApplicationServiceDescription data object of a card-application may be nested in the card-application ACD.

From the ACD, the mapping of DataSets and Services may make reference to Differential-Identities that are either encoded as appropriate Information Objects within the ACD or within the relevant EFs of a DF.CIA application present on the card. As example, the authId attributes from the access rules of a Named Service may point to a CommonKeyAttributes.iD of a Secret Key within the EF.SKD associated to a cryptographic information application.

Each CIOChoice encoded in the CardApplicationServiceDescription can be either encoded as Path or Object.

The figure below shows the relationships and indirections between these data structures and Information Objects.



**Figure 15 — Relationships between ISO/IEC 7816-15 information objects and ISO/IEC 24727-3 data structures**

Since the ISO/IEC 7816-15 data structures are involved in the discovery mechanism, it may be useful to the eService application to recover information about such structures. According to ISO/IEC 24727 (all parts), any on-card structure mapped as a DataSet or DSI may be discovered through the appropriate SAL-API calls. To allow for this discovery process, the cryptographic information application files and related containers may be encoded as much DataSets and DSIs within the CardApplicationServiceDescriptor value in ACD.

ISO/IEC 24727-4:2008/DAmD 1, Annex E ("API for ISO/IEC 7816-15 data structures handling") proposes an interface specification offering classes and functions allowing an eService application, for instance the SAL-implementation layer, to build a representation of ISO/IEC 7816-15 objects contained in a card personalised with DF.CIA associated to card-application(s). Accordingly, eServices laid on a SAL layer and wishing to check/administrate or/and to display the contents of ISO/IEC 7816-15 data may call upon SAL-API entry points that in turn may call the present interface in an interoperable way. The benefit out of the harmonization of the libraries giving access to ISO/IEC 7816-15 on-card data structures is many-fold:

- leveraging SAL layer interchangeability;
- facilitating SAL integration with pre-built classes to generate an object representation of [ISO/IEC 7816-15]-based card content;
- facilitating the *card services discovery* mechanism through rationalization of the recovery and parsing of on-card ISO/IEC 7816-15 data structures;
- supporting legacy cards with ISO/IEC 7816-15 implementation and enhancing interoperability based on ISO/IEC 7816-15;
- enabling the eService application with an [ISO/IEC 7816-15]-based card browser functionality;

additionally, the ECC3-API provides a *transparent channel* to perform administrative tasks in secure messaging mode, e.g. managing secrets from remote server or a personalisation desk.

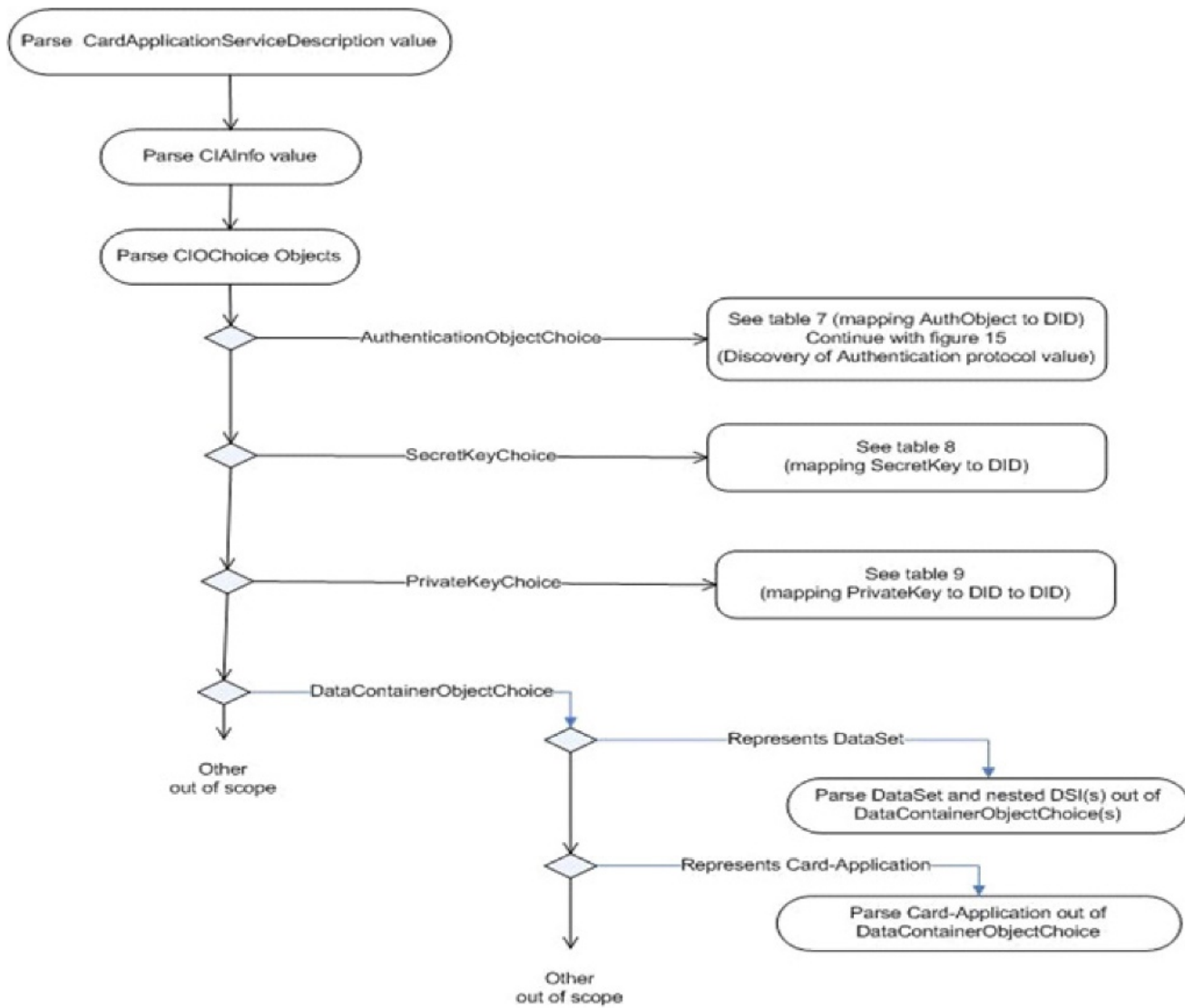
The optional ECC3-API interface described in ISO/IEC 24727-4:2008, Annex J (informative).

### **6.6.5 General discovery mechanism**

On the basis of the mappings relying on ISO/IEC 7816-15 implementation, the general discovery mechanism is exposed in the diagram on Figure 16 — ISO/IEC 7816-15 based general discovery mechanism. It indicates the ISO/IEC 7816-15 source from which data structures for interoperability shall be recovered. Accordingly, the Differential-Identities (DID) shall be retrieved from AuthenticationObjectChoice, PrivateKeyChoice, SecretKeyChoice information objects. DataSets and nested DSIs shall be retrieved from DataContainerObjectChoice information objects. CardApplications shall be retrieved from DataContainerObjectChoice information objects.

DataSet including DSIs, CardApplication and Differential-Identities are the three targets considered in ISO/IEC 24727-3:2008, 5.3:

- an SAL-API action applies upon a target;
- a target has an access control list (ACL) that maps SAL-API actions onto ISO/IEC 7816-15 ActionMode attributes;
- a target has an access control list (ACL) that maps security conditions controlling SAL-API actions onto ISO/IEC 7816-15 security condition byte;
- an access control list (ACL) is a set of access rules.



**Figure 16 — ISO/IEC 7816-15-based general discovery mechanism**

The AccessRule data structure for interoperability shall be generated by the SAL-implementation layer as described in Table 4 below.

For each Named CardApplicationService offered by the card-application, the SAL-implementation shall derive from the ACD as many AccessRule as SAL-API Action acting upon the Target.

```

AccessRule:: = SEQUENCE {
    cardApplicationService  CardApplicationServiceName,
    action                 ActionName,
    securityCondition       SecurityCondition
}
  
```



**Table 4 — Components of AccessRule data structure**

<b>Target</b>	<b>CardApplicationServiceName</b>	<b>ActionName</b>	<b>securityCondition</b>	<b>Description</b>
DATA-SET	“NamedData”, “Authorisation”	Acc. SAL-API value in ISO/IEC 24727 -4:2008, 8.1.2	According to ISO/IEC 24727-3:2008, C.1.3	each Named Service is ASCII encoded.
CARD- APPLICATION	“CardApplication”, “Connection”, “Authorisation”	Acc. SAL-API value in ISO/IEC 24727 -4:2008, 8.1.2	According to ISO/IEC 24727-3:2008, C.1.3	each Named Service is ASCII encoded.
DIFFERENTIAL- IDENTITY	“DifferentialIdentity”, “Authorisation”, “Cryptographic”	Acc. SAL-API value in ISO/IEC 24727 -4:2008, 8.1.2	According to ISO/IEC 24727-3:2008, C.1.3	each Named Service is ASCII encoded.

## 6.7 Other data descriptor

A alpha card-application may either be present onto the card or emulated by the card Root ADF or MF, or by the SAL/GCAL layer. The selection of the alpha card-application is mandatory for the client-application to retrieve the connectionHandle referring to the ECC token. Therefore the GET DATA command to retrieve the CCD is likely to be operated first onto the alpha card-application unless an initial access data element or other element in the historical bytes denoting data objects in EF.ATR are present and lead to the retrieval of the CCD.

In case another AID of a card-application present in the card is known to the client-application, such AID may be used to get the connectionHandle to the ECC token.

The CardInfo-structure may be used for the mapping of generic requests at the Service Access Layer to card-specific APDUs in case there is no CIA according to ISO/IEC 7816-15 available on the card. The cardInfo structure is described on Clause 9.

## 7 Authentication protocols

### 7.1 General

This clause provides description of some authentication protocols supported by ISO/IEC 24727 (all parts) and applicable according to ECC.

### 7.2 Authentication Mechanisms based on ISO/IEC 24727 SAL-API

This subclause describes the exploitation of the API Requests as per ISO/IEC 24727-3 to achieve authentication and end-to-end security between the eService and the card-application.

The *DIDAuthenticate* API request is called by the eService to perform authentication.

```
OUT    return_code  DIDAuthenticate (  
IN     connection_handle      connectionHandle,  
IN     did_scope             didScope,  
IN     did_name              didName,  
IN/OUT did_authentication_data authenticationProtocolData,  
IN     ConnectionHandle      samConnectionHandle  
);
```

- If secure messaging is required between eService and card-application, therefore eService and SAL shall belong to the same trusted environment.
- If the eService and the SAL do not run in the same environment (i.e: Remote-Loyal-Stack), the eService may be involved in the authentication protocol as an authenticated party in the peer-to-peer protocol, and may be led to process some data handled by the authentication protocol. Consecutive *DIDAuthenticate* requests may be necessary to unfold the complete protocol. The 'authenticationProtocolData' parameter (among the *DIDAuthenticate* request arguments) is provided as an alternative to achieve authentication and secure messaging by exchanging authentication data back and forth between the eService and the SAL or SAL Proxy: the SAL-API requests being synchronous, some data might have to be exchanged between the eService and the SAL.
- If the eService and the SAL run in the same environment (i.e Remote-ICC-Stack, Loyal-Stack), the authentication protocol is fully delegated to the SAL. One unique *DIDAuthenticate* request or alternatively *CardApplicationStartSession* shall be used to achieve the protocol, thereby delegating all the complexity to the SAL. *CardApplicationStartSession* API request may be used to establish a secure channel subsequent to an authentication. In a similar way as the *DIDAuthenticate* request, the 'authenticationProtocolData' argument of *CardApplicationStartSession* request allows the exchange of authentication data back and forth between the eService and the SAL.

The 'samConnectionHandle' argument of the *CardApplicationStartSession* request may be used in order to provide a further reference to another card-application or to a SAM (see IFD-API in Annex B for further details)

### **7.3 Asymmetric internal authentication**

Asymmetric internal authentication can be used for Client/Server authentication to grant access rights to the client. The main parts of the protocol are:

- Verification of the card certificate by the server
- Asymmetric Challenge/Response procedure to prove authenticity of the card

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.4 Asymmetric external authentication**

Asymmetric external authentication can be used for the role authentication of the server. The protocol offers the possibility to associate a specific role to an external entity and hence specific access rights. The main parts of the protocol are:

- Verification of the server certificate by the card;
- Asymmetric Challenge/Response procedure to prove the authenticity of the server.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.5 Symmetric internal authentication**

Symmetric internal authentication can be used for Client/Server authentication to grant access rights to the client. The main parts of the protocol are:

- Identify ICC (e.g.read EF.SN);
- Symmetric Challenge/Response procedure to prove authenticity of the card.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.6 Symmetric external authentication**

Symmetric external authentication can be used for the role authentication of the server. The protocol offers the possibility to associate a specific role to an external entity and hence specific access rights. The main parts of the protocol are:

- Identify ICC (e.g.read EF.SN);
- Symmetric Challenge/Response procedure to prove the authenticity of the server.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.7 Mutual authentication with key establishment**

This protocol can be used to perform a symmetric mutual authentication and to exchange session keys. The main parts of the protocol are:

- Symmetric Challenge/Response procedure to prove authenticity of the server and the card;
- Derivation of session keys.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.8 Device authentication with non traceability**

This protocol hinders a verifier to submit a the provable existence of the verification process. The protocol is popular being used with contactless implementations in the context of citizen cards and homeland security. It enhances the existing EAC protocol by aspects of privacy.The main parts of the protocol are:

- Verification of the server certificate by the card;
- Asymmetric Challenge/Response procedure to prove the authenticity of the server;
- Perform Diffie-Hellmann-Key-Agreement between the server and the card;
- Verification of the card certificate by the server.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.9 Key transport protocol based on RSA**

The key transport protocol provides the authentication steps for a two-factor authentication. The protocol is recommended for implementations which required compatibility with external world systems using this key transport protocol. The main parts of the protocol are:

- Verification of the server certificate by the card;

- Verification of the card certificate by the server;
- Asymmetric Challenge/Response procedure to prove authenticity of the card;
- Asymmetric Challenge/Response procedure to prove the authenticity of the server;
- Session key derivation.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

### **7.10 Terminal Authentication**

Asymmetric external authentication can be used for the role authentication of the server. The protocol offers the possibility to associate a specific role to an external entity and hence specific access rights. The main parts of the protocol are:

- a) Verification of the server certificate by the card
- b) Asymmetric Challenge/Response procedure to prove the authenticity of the server:
  - 1) the server requests a nonce from the card;
  - 2) the card verifies the signature performed by the server upon the nonce.

The Public Key attached to the server certificate is allowed to execute some actions on the card according to the role policy.

Details can be found in ISO/IEC 24727-3 and EN 14890-1.

## **8 IFD-API Web Service Binding**

The Web Service Binding for the IFD-API is defined in ISO/IEC 24727 (all parts) as follows:

- a) ISO/IEC 24727-4:2008, Annex B:
  - 1) ISOCCommon.XSD defines basic types such as the ResponseType, which forms the basis for all response messages.
  - 2) ISOIFD.XSD in which the parameters for each IFD-API request and response are specified as XML-elements.
  - 3) CENIFD.WSDL which includes CENIFD.XSD and specifies the Web Service binding for the IFD-API.
- b) ISO/IEC 24727-4:2008, Annex C:
  - 1) CENCallback.XSD which defines as XML-Elements the parameters for SignalEvent request and response.
  - 2) CENCallback.WSDL which includes CENCallback.XSD definitions and specifies the SOAP-based Web Service Binding for IFD-Callback-API.

## **9 Card-Info Structure — Introduction**

The present standard series specifies various aspects of the European Citizen Card and the present part provides additional technical specifications for a middleware architecture based on ISO/IEC 24727 (all parts). In order to use the generic application interface defined in this standard with cards of a given profile defined in

ECC-4 or with legacy cards, it is necessary to provide a certain set of information, which allows to map the generic calls at the Service Access Layer to card-specific APDUs.

The XML-schema based CardInfo defines a structure that may both be used for the specification of card profiles in ECC-4 *and* for the mapping of generic calls at the Service Access Interface (ref. ISO/IEC 24727) to card-specific APDUs of legacy cards.

Complete XML-based CardInfo structure is provided in ISO/IEC 24727-3:2008, D.3.

## 10 XML-based Service Access Layer Interface

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

- `CENSAL.XSD` – the basic types and the parameters for each SAL-API request and response are specified as XML-elements.
- `CENSAL.WSDL` – which includes `CENSAL.XSD` and specifies the SOAP-based Web Service binding for the SAL-API.

For details refer to ISO/IEC 24727-3:2008, Annex F.

## 11 Federative Framework-wise Authenticate API

### 11.1 General

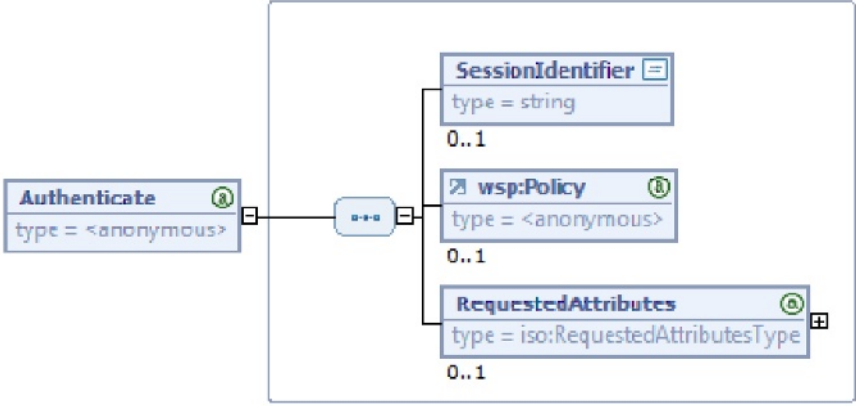
To allow for SAL-Interface to perform authentication with the same efficiency as existing federation and authentication systems, like SAML, OpenID, WS-Trust, the introduction of a new API is needed with the following rationale:

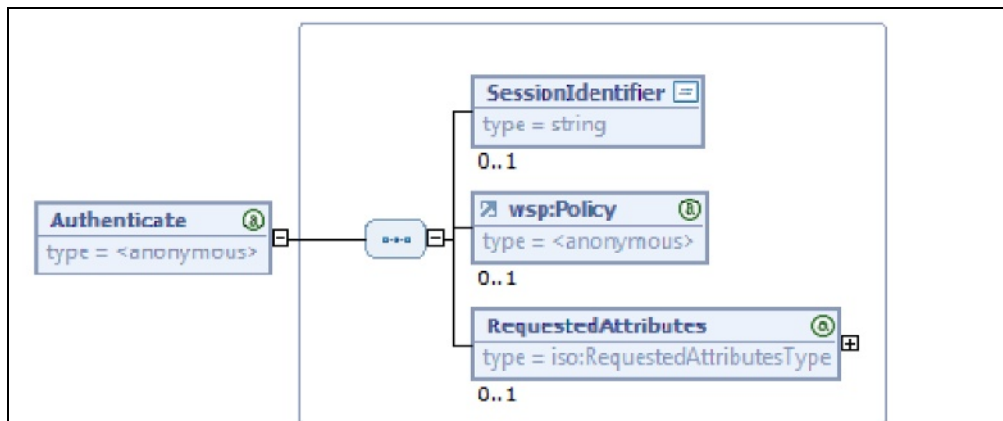
While all the authentication systems above mentioned require that an authentication is performed and certain attributes are read from the card, the use of the SAL-interface for this purpose is not optimal, as it would typically require many calls to perform an authentication and identification.

A new Authenticate() API call conveying at least three parameters (CardType, Application Policy, RequestedAttributes) is thus needed to cater to existing federative and authentication systems. This Authenticate() call occurs above SAL layer and is processed by a dispatcher that address it to the appropriate module: e.g. mERA (i.e FR eID), ISO/IEC 24727-compatible SAL(i.e DE eID), MOA (i.e AT eID), PEPS (SP eID) managing the transaction with eID card according national deployment system over Europe.

The Authenticate() call makes the service layer module transparent to the Service Provider.

### 11.2 Authenticate method

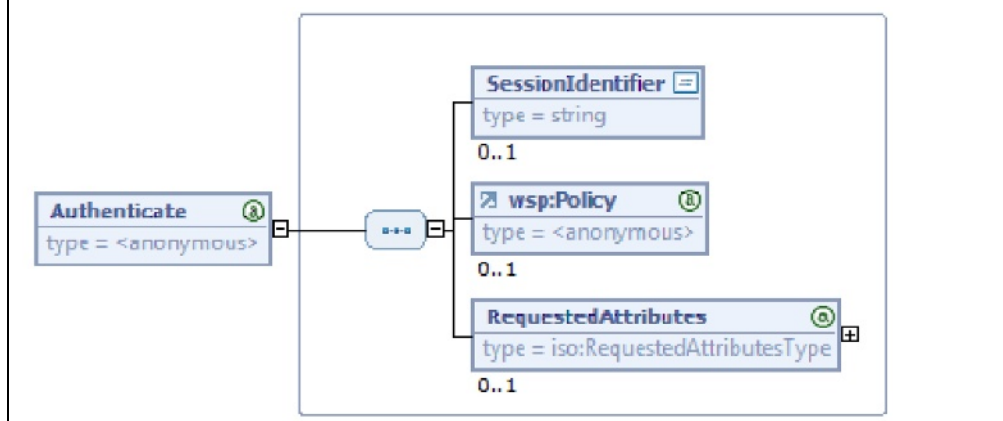
<b>Name</b>	Authenticate	
<b>Description</b>	The <code>Authenticate</code> function is used to request the authentication of an entity according to a specified policy and may request additional identity attributes.	
<b>Invocation parameters</b>	 <p>Invocation of the <code>Authenticate</code> function.</p>	
<b>Name</b>	<b>Description</b>	
SessionIdentifier	<p>Contains the identifier of a session in which the authentication is performed.</p> <p>If the parameter is omitted, the corresponding response message SHALL contain a <code>SessionIdentifier</code> element.</p>	
wsp:Policy	<p>Specifies the applicable authentication policy. Details with respect to the syntax and semantic of this generic element are defined in <b>[WS-Policy(v1.5)]</b>.</p> <p>If the parameter is omitted, the authentication service MAY apply a default policy.</p>	
RequestedAttributes	<p>Specifies which additional identity attributes are to be retrieved after the authentication procedure has been performed.</p> <p>If the parameter is omitted, the authentication service SHOULD NOT return identity attributes.</p> <p>More details with respect to this element are specified below.</p>	



The RequestedAttributes-element MAY be part of the Authenticate request. Instead of explicitly specifying the set of requested identity attributes, this element MAY contain an attribute RequestAllDefaults of type anyURI to indicate that a specific set of default attributes is requested. If the RequestedAttributes-element neither contains saml:Attribute-elements nor the RequestAllDefaults attribute, the authentication service MAY return a set of default attributes.

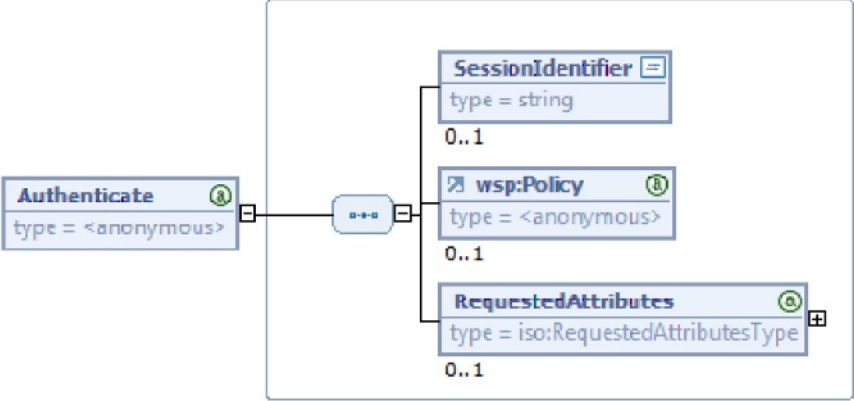
Name	Description
saml:Attribute	This element MAY appear an arbitrary number of times and is used to request a certain identity attribute. Details with respect to the syntax and semantic of this generic element are defined in Section 2.7.3.1 of [SAML(v2.0)].

**Return**



Return of the Authenticate function.

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

<p>SessionIdentifier</p>	<p>Contains the identifier of a session in which the authentication is performed.          This element SHALL be present if it was missing in the Authenticate-request.</p>
<p>saml:Attribute</p>	<p>This element MAY appear an arbitrary number of times and is used to convey the requested identity attributes.</p>
<div style="text-align: center;">  </div> <p>Status information and errors for Authenticate.</p>	
<p><b>Name</b></p>	<p><b>Error codes</b></p>
<p>ResultMajor</p>	<ul style="list-style-type: none"> <li>• <a href="#">/resultmajor#ok</a></li> <li>• <a href="#">/resultmajor#error</a></li> <li>• <a href="#">/resultmajor#warning</a></li> </ul>
<p>ResultMinor</p>	<ul style="list-style-type: none"> <li>• <a href="#">/resultminor/il/common#noPermission</a></li> <li>• <a href="#">/resultminor/il/common#internalError</a></li> <li>• <a href="#">/resultminor/il/common#parameterError</a></li> <li>• <a href="#">/resultminor/dp#unknownChannelHandle</a></li> <li>• <a href="#">/resultminor/dp#communicationError</a></li> <li>• <a href="#">/resultminor/dp#trustedChannelEstablishmentFailed</a></li> <li>• <a href="#">/resultminor/dp#unknownProtocol</a></li> <li>• <a href="#">/resultminor/dp#unknownCipherSuite</a></li> <li>• <a href="#">/resultminor/dp#unknownWebserviceBinding</a></li> <li>• <a href="#">/resultminor/il/is#authenticationFailure</a></li> <li>• <a href="#">/resultminor/il/is#unknownPolicyWarning</a></li> <li>• <a href="#">/resultminor/il/is#unknownAttributeWarning</a></li> <li>• <a href="#">/resultminor/il/is#unknownDefaultURIWarning</a></li> </ul>
<p>ResultMessage</p>	<p>MAY contain more detailed information on the occurred error if required.</p>



<b>Precondition</b>	In order to perform the authentication a valid <code>SessionIdentifier</code> is required.
<b>Postcondition</b>	
<b>Note</b>	If the required <code>SessionIdentifier</code> is not available yet, it can be requested by invoking <code>Authenticate</code> without the <code>SessionIdentifier</code> parameter.

## 11.3 Web Service Binding for Authenticate API

### 11.3.1 General

The Web Service Binding for the Authenticate API is defined in the following files:

- `Authenticate.XSD`: provide XML Elements for basic types and parameters for each request/response of the API;
- `Authenticate.WSDL`: provide SOAP-based web service binding for the API; include `Authenticate.XSD`.

### 11.3.2 Authenticate.XSD definition

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="1.0.0">

  <include schemaLocation="../../../2011/ISO24727-3.xsd" />
  <include schemaLocation="../../../2011/ISOCommon.xsd" />
  <import namespace="http://www.w3.org/ns/ws-policy"
  schemaLocation="http://www.w3.org/2007/02/ws-policy.xsd" />
  <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
  schemaLocation="../../../2011/saml-schema-assertion-2.0.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" />

  <!-- ===== ->
  <!--           Version / Date           ->
  <!-- ===== ->
  <!--           30.05.2011             ->
  <!-- ===== ->

  <!-- ===== ->
  <!--           Authenticate             ->
  <!-- ===== ->

  <complexType name="RequestedAttributesType">
```

```

    <sequence>
      <element ref="saml:Attribute" minOccurs="0" maxOccurs="unbounded"
/>
    </sequence>
    <attribute name="RequestAllDefaults" type="anyURI" />
  </complexType>

<element name="Authenticate">
  <complexType>
    <complexContent>
      <extension base="iso:RequestType">
        <sequence>
          <element name="SessionIdentifier"
            type="string" maxOccurs="1" minOccurs="0" />
          <element ref="wsp:Policy" maxOccurs="1"
            minOccurs="0">
        </element>
          <element name="RequestedAttributes"
            type="iso:RequestedAttributesType" maxOccurs="1"
            minOccurs="0">
        </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>

<element name="AuthenticateResponse">
  <complexType>
    <complexContent>
      <extension base="iso:ResponseType">
        <sequence>
          <element name="SessionIdentifier"
            type="string" maxOccurs="1" minOccurs="0">
        </element>
          <element ref="saml:Attribute"
            maxOccurs="unbounded" minOccurs="0">
        </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>

</schema>
```

### 11.3.3 Authenticate.WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
```

```
targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0.0">
```

```
<include schemaLocation="../../../2011/ISO24727-3.xsd" />
<include schemaLocation="../../../2011/ISOCommon.xsd" />
<import namespace="http://www.w3.org/ns/ws-policy"
schemaLocation="http://www.w3.org/2007/02/ws-policy.xsd" />
<import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
schemaLocation="../../../2011/saml-schema-assertion-2.0.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" />
```

```
<!-- ===== ->
<!--          Version / Date          ->
<!-- ===== ->
<!--          30.05.2011              ->
<!-- ===== ->
```

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

```
<complexType name="RequestedAttributesType">
  <sequence>
    <element ref="saml:Attribute" minOccurs="0" maxOccurs="unbounded"
/>
  </sequence>
  <attribute name="RequestAllDefaults" type="anyURI" />
</complexType>
```

```
<element name="Authenticate">
  <complexType>
    <complexContent>
      <extension base="iso:RequestType">
        <sequence>
          <element name="SessionIdentifier"
            type="string" maxOccurs="1" minOccurs="0" />
          <element ref="wsp:Policy" maxOccurs="1"
            minOccurs="0">
        </element>
          <element name="RequestedAttributes"
            type="iso:RequestedAttributesType" maxOccurs="1"
            minOccurs="0">
        </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
```

```
<element name="AuthenticateResponse">
  <complexType>
    <complexContent>
      <extension base="iso:ResponseType">
        <sequence>
          <element name="SessionIdentifier"
            type="string" maxOccurs="1" minOccurs="0">
          </element>
          <element ref="saml:Attribute"
            maxOccurs="unbounded" minOccurs="0">
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
```

```
</schema>
```

## Annex A (informative)

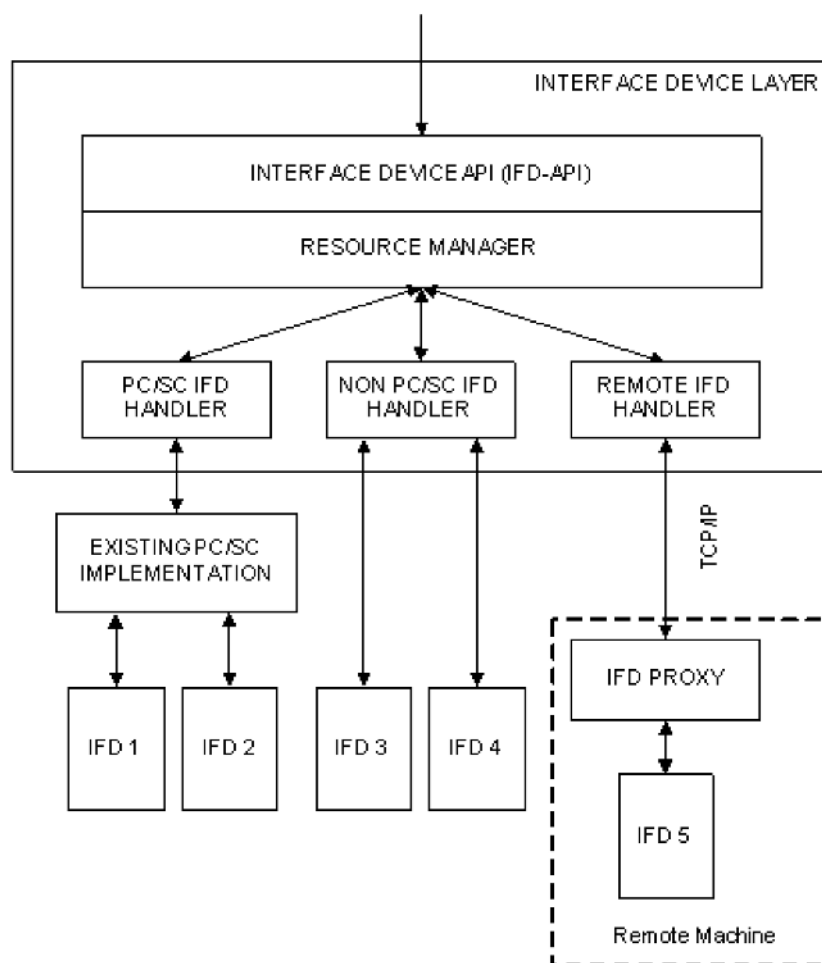
### Interface Device Layer Architecture and Management

#### A.1 Scope

The Interface Device Layer (IFD-Layer) is the lowest layer of the middleware stack. It is responsible for handling card readers and transporting APDUs to the card. This layer provides an abstraction of the applied card terminal technology and hence allows using different types of card terminals (PC/SC, MCT, SICCT, etc.) via a harmonized interface – the Interface Device API. Details of IFD-API functions are provided in ISO/IEC 24727-4:2008, 7.4. Additional features specific to ECC-3 are described in E.4.

#### A.2 IFD-Layer Architecture

The architecture of the IFD-Layer is depicted in the following figure:



**Figure A.1 — IFD-Layer Architecture**

The layers above the IFD-Layer access the card terminals and cards using the Interface Device API (IFD-API). The Resource Manager handles different IFD-Handlers, which in turn handle one or more Interface Devices.

## **A.3 Resource Manager**

### **A.3.1 General**

The Resource manager keeps a persistent database of registered IFD-Handlers present in the system. This database may be modified by appropriate management functions (e.g. RegisterIFDHandler and UnregisterIFDHandler). There is one special PC/SC IFD handler which is not included in this database, is not listed by ListIFDHandlers function and is always present in the system. This IFD-Handler manages all standard PC/SC readers present on the host operating system.

Resource manager processes on its own only the IFD-Handler related functions (see A.4.1) and forwards all other calls to the appropriate IFD-Handler for processing.

For a Wait request the resource manager creates as many threads as there are IFD-Handlers in the system and each thread calls Wait function. Function returns when one of the calls to Wait finishes or Cancel is called.

The ListIFDs function merges the lists of readers obtained from each IFD-Handler.

RegisterIFD function is used for introducing a new reader into the system. Resource manager shall check that the IFDName is unique before forwarding the call to the RegisterIFD function.

### **A.3.2 IFD-Handlers**

An IFD-Handler serves as a driver for a particular reader. One or more readers can be controlled by one IFD-Handler. Each IFD-Handler shall keep a persistent database of registered readers. This database is modified by RegisterIFDHandler and UnregisterIFDHandler functions. The IFD-Handler reads the database when it is loaded into memory by the Resource Manager and presents it through ListIFDHandlers function.

RegisterIFD is used to introduce a new reader into the system. One IFD-Handler can control more than one reader. It is the responsibility of the IFD-Handler to know how many readers are connected for a particular IFD-Handler and to indicate it as return value in the RegisterIFD function.

### **A.3.3 Card transactions**

An application may start a card transaction by calling BeginTransaction and ends it by calling EndTransaction. This transaction mechanism prevents other applications from accessing the card. If application A is inside a card transaction and application B calls any IFD-API function with the same card handle, processing of this call is postponed until application A ends card transaction.

### **A.3.4 Application threads**

If an application runs in more than one thread, only one call to the IFD Layer is processed at a time. Processing of calls from other threads is postponed until the call being processed returns and call from another thread is processed. The only exception to this rule is the Cancel function. This call is processed even though there is already an existing call to the IFD Layer in progress.

## **A.4 Administrative functions**

### **A.4.1 IFD-Handler related functions**

There are the following IFD-Handler specific functions:

- RegisterIFDHandler
- UnregisterIFDHandler
- ListIFDHandlers
- IFDHandlerCount

#### **A.4.2 Interface Device related functions**

- RegisterIFD
- UnregisterIFD

## Annex B (informative)

### IFD-API – C Language Binding

```
typedef unsigned int UnsignedIntegerType;
typedef unsigned int PositiveIntegerType;
typedef unsigned int StatusType;
typedef unsigned int ChannelHandleType;
typedef unsigned int ContextHandleType;
typedef unsigned int SlotHandleType;
typedef unsigned int BooleanType;
typedef wchar_t* StringType;
typedef char* OctetStringType;
typedef unsigned int CardHandleType;

typedef enum
{
    IFD_RESET = 0,
    IFD_UNPOWER,
    IFD_EJECT,
    IFD_CONFISCATE
} ActionType;

typedef enum
{
    IFD_BIOMETRIC_TYPE = 0,
    IFD_PIN_TYPE
} InputUnitStructureType;

typedef enum
{
    IFD_PIN_BCD = 0,
    IFD_PIN_ASCII,
    IFD_PIN_PIN2
} PinFormatType;

// ----- Miscellaneous definitions -----
#define IFD_INVALID_HANDLE          0x00000000

// ----- Return Codes -----
#define IFD_OK                      0
#define IFD_UNKNOWN_ERROR          1
#define IFD_INVALID_CHANNEL_HANDLE 2
#define IFD_INVALID_CONTEXT_HANDLE 3
#define IFD_UNKNOWN_IFD            4
#define IFD_TIMEOUT                 5
#define IFD_CANCEL_NOT_POSSIBLE    6
#define IFD_UNKNOWN_SLOT           7
#define IFD_SHARING_VIOLATION      8
#define IFD_NO_CARD                 9
#define IFD_INVALID_CARD_HANDLE    10
#define IFD_UNKNOWN_ACTION         11
#define IFD_NO_TRANSACTION_STARTED 12
```



```

#define IFD_UNKNOWN_INPUT_UNIT          13
#define IFD_CANCELLATION_BY_USER       14
#define IFD_UNKNOWN_PIN_FORMAT         15
#define IFD_UNKNOWN_BIOMETRIC_SUBTYPE  16
#define IFD_REPEATED_DATA_MISMATCH     17
#define IFD_DISPLAY_INDEX              18
#define IFD_SMALL_BUFFER               19

// DOUBLE CALL MECHANISM
/* Some functions return data of variable length. An application can use this
mechanism to find out how many bytes to allocate to be able to retrieve the data.
There are two parameters:
<Buffer> - pointer to allocated buffer
<Size> - pointer to buffer size
An application first calls function with <Buffer> parameter set to NULL and
receives in <Size> parameter necessary buffer size. The second time the
application calls the function with <Buffer> not set to NULL and <Size> parameter
shall contain number of allocated bytes. In addition to return codes described in
Annex B, each function supporting this mechanism can return IFD_SMALL_BUFFER if
buffer allocated for data is too small. */

// <pContext> - this parameter cannot be NULL
// <ChannelHandle> - if set to NULL, this parameter is not present
StatusType EstablishContext( ChannelHandleType *ChannelHandle,
ContextHandleType *Context);

StatusType ReleaseContext(ContextHandleType Context);

/*
<IfdName> - If NULL, Size parameter is filled with size in characters necessary
to read list of IFDs. If not NULL, Size parameter shall contain buffer size in
characters allocated for list of IFDs. IfdName is filled with multi-string
containing IFD names. Multi-string is defined as concatenation of NULL terminated
strings. Length of multi-string is supplied in <Size> parameter.

<Size> - is used as input parameter for defining size in characters of IfdName
buffer. As output parameter it supplies the length of multi-string.

In addition to return codes described in Annex B, this function returns
IFD_SMALL_BUFFER if buffer allocated for IFD list is too small. */

StatusType ListIFDs(ContextHandleType Context, StringType IfdName,
unsigned int *Size);

typedef struct
{
    UnsignedIntegerType Index;
    BooleanType ContactBased;
}SlotCapabilityType;

typedef struct
{
    UnsignedIntegerType Index;
    UnsignedIntegerType Lines;

```

```
    UnsignedIntegerType Columns;
    // if set to NULL this parameter is not present
    UnsignedIntegerType *VirtualLines;
    // if set to NULL this parameter is not present
    UnsignedIntegerType *VirtualColumns;
}DisplayCapabilityType;

typedef struct
{
    UnsignedIntegerType Index;
    PositiveIntegerType Keys;
}KeyPadCapabilityType;

typedef struct
{
    UnsignedIntegerType Index;
    UnsignedIntegerType BiometricType;
}BioSensorCapabilityType;

typedef struct
{
    SlotCapabilityType *SlotCapability;
    // number of items in SlotCapability array
    UnsignedIntegerType SlotCapabilityCount;

    DisplayCapabilityType *DisplayCapability;
    // number of items in DisplayCapability array
    UnsignedIntegerType DisplayCapabilityCount;

    KeyPadCapabilityType *KeyPadCapability;
    // number of items in KeyPadCapability array
    UnsignedIntegerType KeyPadCapabilityCount;

    BioSensorCapabilityType *BioSensorCapability;
    // number of items in BioSensorCapability array
    UnsignedIntegerType BioSensorCapabilityCount;

    BooleanType          OpticalSignalUnit;
    BooleanType          AcousticSignalUnit;
}IFDCapabilitiesType;

// DOUBLE CALL mechanism can be used for retrieving IFDCapabilities.
StatusType GetIFDCapabilities(ContextHandleType ContextHandle,
StringType IFDName,
IFDCapabilitiesType *IFDCapabilities, UnsignedIntegerType *Size);

typedef struct
{
    UnsignedIntegerType Index;
    BooleanType          CardAvailable;
    // if set to NULL, this parameter is not present
    OctetStringType     ATRorATS;
    UnsignedIntegerType ATRorATSLength;
}SlotStatusType;

typedef struct
{
```

```

    UnsignedIntegerType Index;
    BooleanType Available;
}SimpleFUStatusType;

typedef struct
{
    StringType IFDName;

    // if set to NULL this parameter is not present
    BooleanType *Connected;

    SlotStatusType *SlotStatus;
    // number of items in SlotStatus array
    UnsignedIntegerType SlotStatusCount;

    // if set to NULL this parameter is not present
    BooleanType *ActiveAntena;

    SimpleFUStatusType *DisplayStatus;
    // number of items in DisplayStatus array
    UnsignedIntegerType DisplayStatusCount;

    SimpleFUStatusType *KeyPadStatus;
    // number of items in KeyPadStatus array
    UnsignedIntegerType KeyPadStatusCount;

    SimpleFUStatusType *BioSensorStatus;
    // number of items in BioSensorStatus array
    UnsignedIntegerType BioSensorStatusCount;
}IFDStatusType;

// DOUBLE CALL mechanism can be used for retrieving IFDStatus.
// <IFDName> - if set to NULL, this parameter is not present
StatusType GetStatus( ContextHandleType ContextHandle,
StringType IFDName,
                    IFDStatusType *IFDStatus,
UnsignedIntegerType *Size);

// <Timeout> - if set to NULL, this parameter is not present
StatusType Wait( ContextHandleType ContextHandle,
PositiveIntegerType *Timeout,
                IFDStatusType *IFDStatus,
                UnsignedIntegerType IFDStatusCount,
                IFDStatusType *IFDEvent,
                UnsignedIntegerType IFDEventCount);

StatusType Cancel(ContextHandleType ContextHandle, StringType IFDName);

// DOUBLE CALL mechanism can be used for retrieving Response.
StatusType ControlIFD( ContextHandleType ContextHandle,
StringType IFDName,
OctetStringType Command,
UnsignedIntegerType CommandLength,
OctetStringType Response,
UnsignedIntegerType *ResponseLength);

```

```
// <Exclusive> - if set to NULL, this parameter is not present
StatusType Connect( ContextHandleType ContextHandle,
                   StringType IFDName,
                   UnsignedIntegerType Slot,
                   BooleanType *Exclusive,
                   SlotHandleType *SlotHandle);

// <Action> - if set to NULL, this parameter is not present
StatusType Disconnect(ContextHandleType ContextHandle, ActionType *Action);

StatusType BeginTransaction(SlotHandleType SlotHandle);
StatusType EndTransaction(SlotHandleType SlotHandle);

/* DOUBLE CALL mechanism cannot be used for this function. OutputAPDULength shall
contain length in bytes of allocated buffer and is filled with actual response
length. In addition to return codes described in Annex B, this function returns
IFD_SMALL_BUFFER if buffer allocated for OutputAPDU is too small. */
StatusType Transmit(SlotHandleType SlotHandle,
                   OctetStringType InputAPDU,
                   UnsignedIntegerType InputAPDULength,
                   OctetStringType OutputAPDU,
                   UnsignedIntegerType *OutputAPDULength);

typedef struct
{
    InputUnitStructureType StructureType;
    // pointer to either PinInputTypeType or BiometricInputTypeType structure
    void *Structure;
}InputUnitType;

typedef struct
{
    UnsignedIntegerType Index;
    PinFormatType PinFormat;
    UnsignedIntegerType MinLength;
    PositiveIntegerType MaxLength;
}PinInputTypeType;

typedef struct
{
    UnsignedIntegerType Index;
    UnsignedIntegerType BiometricSubtype;
}BiometricInputTypeType;

typedef struct
{
    StringType AuthenticationRequestMessage;
    StringType SuccessMessage;
    StringType AuthenticationFailedMessage;
    StringType RequestConfirmationMessage;
    StringType CancelMessage;
}AltVUMessagesType;
```

/\* DOUBLE CALL mechanism cannot be used for this function. ResponseLength shall contain length in bytes of allocated buffer and is filled with actual response length. In addition to return codes described in Annex B, this function returns IFD\_SMALL\_BUFFER if buffer allocated for Response is too small. <DisplayIndex>, <AltVUMessages>, <TimeoutUntilFirstKey>, <TimeoutAfterFirstKey> - if set to NULL, those parameters are not present. \*/

```
StatusType VerifyUser( SlotHandleType SlotHandle,
                      InputUnitType InputUnit,
                      UnsignedIntegerType *DisplayIndex,
                      AltVUMessagesType *AltVUMessages,
                      PositiveIntegerType *TimeoutUntilFirstKey,
                      PositiveIntegerType *TimeoutAfterFirstKey,
                      OctetStringType Template,
                      UnsignedIntegerType TemplateLength,
                      OctetStringType Response,
                      UnsignedIntegerType *ResponseLength);
```

typedef struct

```
{
    StringType AuthenticationRequestMessage;
    StringType SuccessMessage;
    StringType AuthenticationFailedMessage;
    StringType EnterNewAuthenticationDataMessage;
    StringType RepeatInputMessage;
    StringType ComparisonOfRepeatedDataFailed;
    StringType RequestConfirmationMessage;
    StringType CancelMessage;
}AltMVDMessagesType;
```

/\* DOUBLE CALL mechanism cannot be used for this function. ResponseLength shall contain length in bytes of allocated buffer and is filled with actual response length. In addition to return codes described in Annex B, this function returns IFD\_SMALL\_BUFFER if buffer allocated for Response is too small. <DisplayIndex>, <AltVUMessages>, <OldReferenceData>, <TimeoutUntilFirstKey>, <TimeoutAfterFirstKey>, <RepeatInput> - if set to NULL, those parameters are not present. \*/

```
StatusType ModifyVerificationData( SlotHandleType SlotHandle,
                                  InputUnitType InputUnit,
                                  UnsignedIntegerType *DisplayIndex,
                                  AltMVDMessagesType *AltVUMessages,
                                  OctetStringType *OldReferenceData,
                                  UnsignedIntegerType OldReferenceDataLength,
                                  PositiveIntegerType *TimeoutUntilFirstKey,
                                  PositiveIntegerType *TimeoutAfterFirstKey,
                                  BooleanType *RepeatInput,
                                  OctetStringType Template,
                                  UnsignedIntegerType TemplateLength,
                                  OctetStringType Response,
                                  UnsignedIntegerType *ResponseLength);
```

```
StatusType Output( ContextHandleType ContextHandle,
                  StringType IFDName,
                  UnsignedIntegerType DisplayIndex,
                  StringType Message,
                  BooleanType AcousticSignal,
                  BooleanType OpticalSignal);
```

## Annex C (informative)

### SAL-API Post-issuance personalisation requests

#### C.1 General

This annex specifies the SAL features allowing for a proper handling of the post-issuance personalisation requests. A set of SAL-API requests entails creation of data structures (files, dedicated files, logical containers) or services or applications onto the card. Such operations need a specific adjustment to meet the card requirements and to ensure interoperability. This adjustment is brought through two alternatives:

- specification of a canonical protocol that shall be supported by SAL implementations compliant with this standard, along with additional input or/and output parameters in the request prototype that provide extra means to process the SAL-API request
- use of an XML-based CardInfo file made available to the SAL implementation and instanciated upon the model provided by the CardInfo structure defined in Clause 8.

The canonical choice is more restrictive than the use of XML-based CardInfo since it determines a short set of APDU commands that is applied systematically to translate the SAL-API requests while the CardInfo file is customised for each card and can be adjusted to the card edge.

#### C.2 Post-issuance personalisation requests

- DataSetCreate
- DSICreate
- DIDCreate
- DIDUpdate
- CardApplicationCreate
- CardApplicationServiceCreate
- CardApplicationServiceLoad

Amongst the post-issuance personalisation requests listed above, only the following are addressed by the present standard: DataSetCreate, DSICreate, CardApplicationServiceCreate

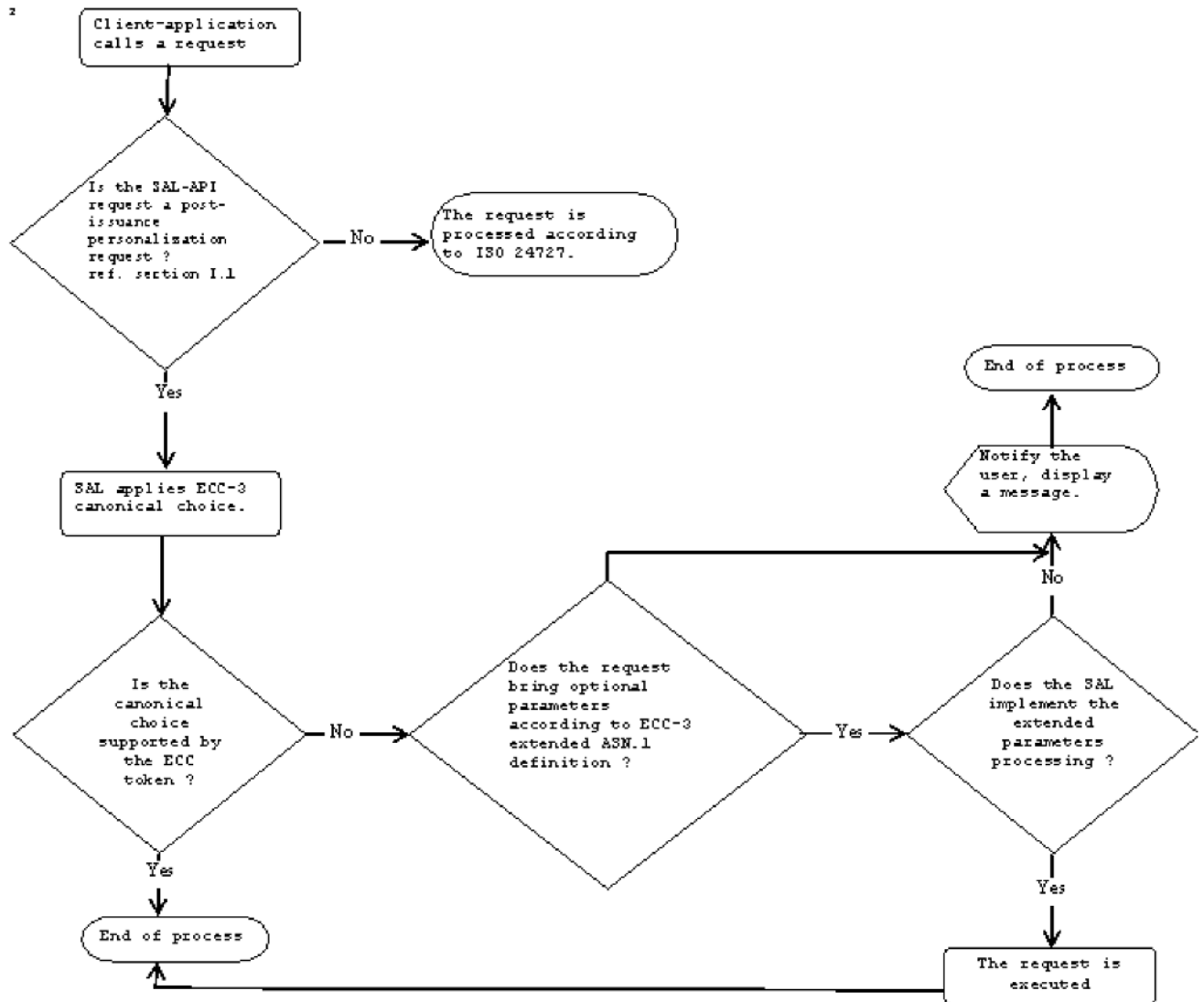
DIDCreate processing is specified uniquely when existing cryptographic objects and related mechanisms are reused.

DIDUpdate is described in the context of a specific authentication mechanism to which the DID is attached (e.g ISO/IEC 24727-3:2008, Annex A)

#### C.3 Canonical protocol

##### C.3.1 General

The canonical protocol is applied according to the activity diagram below.



**Figure C.1 — SAL global Decision Tree**

The canonical choice is specified in terms of translation from the SAL-API request parameters into resulting APDU(s) parameters intended for the card. The canonical choice is mandated for the ECC-3 compatible SAL.

If the canonical choice is not supported by the card, then SAL may process additional input parameters in the SAL-API request. These parameters are optional and completing the ASN.1 definition of ISO/IEC 24727-3 standard.

If the SAL is not enabled for additional ASN.1 parameters, then a notification message will be displayed to the end user.

The SAL decision tree substitutes for Procedural Elements that are not useful anymore.

### **C.3.2 DataSetCreate**

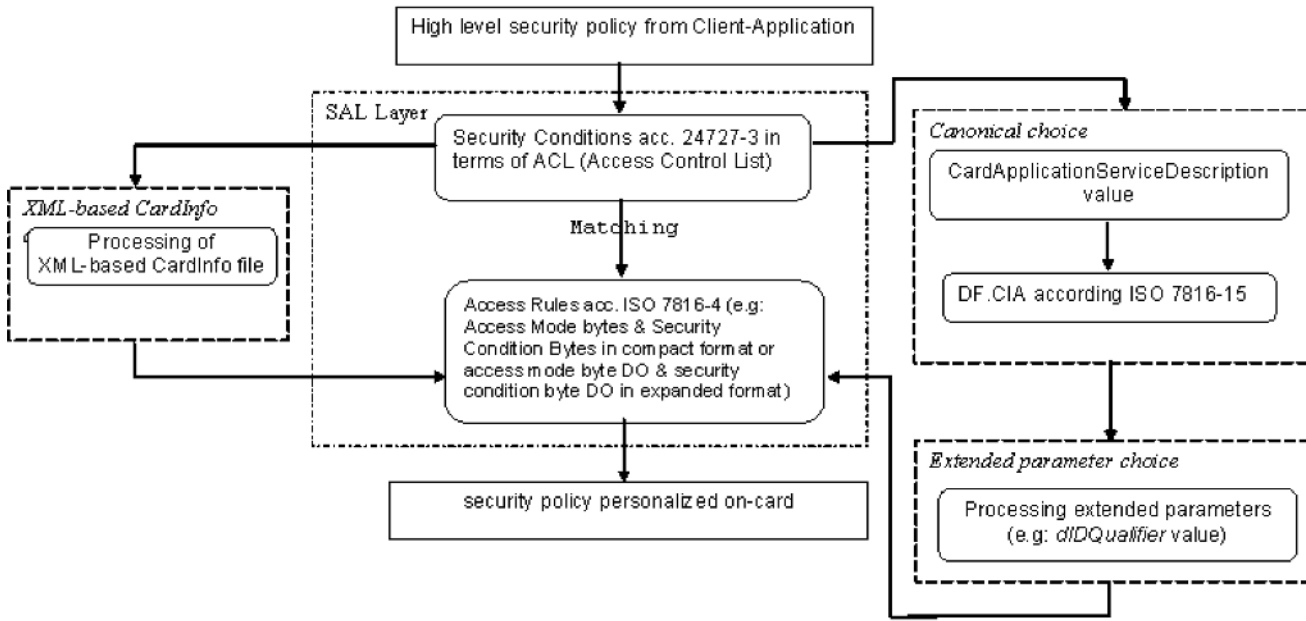
#### **C.3.2.1 General**

Input parameters: connectionHandle, dataSetName, dataSetACL

**C.3.2.2 Parameter mapping onto ISO/IEC 7816 (all parts)**

The APDU result of the SAL processing described in this clause and addressed at GCI shall be CREATE FILE (EF) with INS = 'E0' and P1≠'00'. The Data Field content of this CREATE FILE command is determined in this clause.

the diagram below shows the principle for matching high-level security policy with ISO/IEC 7816-4 access control rules:



**Figure C.2 —From ISO/IEC 24727-3 security policy to ISO/IEC 7816-4 access control rules**

Mapping per request parameter:

- a) connectionHandle: not applicable
- b) dataSetName: possible alternatives

File ID with tag '83' within FCP

DF Name with tag '84' within FCP

Proprietary information with tag '85' within FCP = canonical choice

- c) dataSetACL:

- 1) sub-parameter **cardApplicationServiceName** = "NamedData"



2) sub-parameter **action** = a set of SAL-API requests acting upon the DataSet currently being created is mapped as follows:

<b>SAL-API action</b>	<b>GCI APDU</b>	<b>Canonical choice</b>
DataSetSelect	GET DATA	Cached in SAL
	SELECT	X
DataSetList	GET DATA	Cached in SAL (see NOTE 1)
	SELECT	Cached in SAL
DataSetDelete	DELETE FILE	X
DSIList	GET DATA	Cached in SAL
	SELECT	Cached in SAL
DSICreate	PUT DATA	
	CREATE FILE	
	UPDATE BINARY	X
DSIWrite	UPDATE BINARY	
DSIRead	READ BINARY	X cached in SAL
	GET DATA	Cached in SAL
DSIDelete	DELETE FILE	X
	PUT DATA	X

NOTE 1 "Cached in SAL" means that the DataSetList result returned by the SAL may already be available among the meta-data generated on the fly by the SAL as part of the discovery mechanism. Therefore no APDU needs to be addressed to the card to recover DataSetList result. This statement is true for all the SAL-API requests withdrawing data from the card.

— sub-parameter **securityCondition** = mapped onto ISO 7817-4 security condition byte as follows

SecurityCondition as per ISO/IEC 24727-3, Annex C, C.1.3	Security condition byte according ISO/IEC 7816-4	Canonical choice	using extended parameters
ALWAYS	No condition	X	
NEVER	Never	X	
N/A	b4-b1 = '0'		
didAuthentication	Security Environment identifier	X (see paragraph just after NOTE 2)	X (see NOTE 2)
N/A	b4-b1 = '1' (RFU)		
OR	At least one condition	X	
AND	All conditions	X	
didAuthentication	Secure Messaging	X (see NOTE 4)	
didAuthentication	External Authentication	X (see NOTE 4)	
didAuthentication	User Authentication	X (see NOTE 4)	

NOTE 2 Alternatively, the Security environment reference can be clearly stated in the API action parameters. Such security environment is personalized on-card before the execution of DataSetCreate request.

If a DF.CIA implementation is available on-card it can be parsed by the SAL to discover the security environment reference. For this purpose, the SAL parse the didAuthentication parameters nested in the dataSetACL. Among these parameters, the diDName provides the identifier of the Differential-Identity determining the access rules that control the API action. Then the SAL shall read the Authentication Object that maps the Named DID within the EF.AOD file. The **CommonObjectAttributes.label** of this Authentication Object evaluates to "diDName". Then the SAL read the security environment to which the Authentication Object belongs, and of which the reference is encoded in the **CommonAuthenticationObjectAttributes.seIdentifier** attribute. The discovery of the Authentication protocol value described in ISO/IEC 24727-4:2008, 8.1.1.4 and Table 8 applies.

Starting from the diDName present in the request ACL (e.g of dataSetACL):

- the **CommonObjectAttributes** lookup the AuthenticationObject thereby identifying the DID,
- the **ClassAttributes** provide the SE identifier and the Marker reference,
- the **TypeAttributes** provide the Authentication Protocol OID according to ISO/IEC 24727-4:2008, 8.1.1.4 and Table 8-4.

The SAL can rebuilt the ISO/IEC 7816-4 security rules applying to the APDU command corresponding to the SAL-API request. Consistency between the information shall be checked by the SAL. The processing steps are illustrated in the following figure:

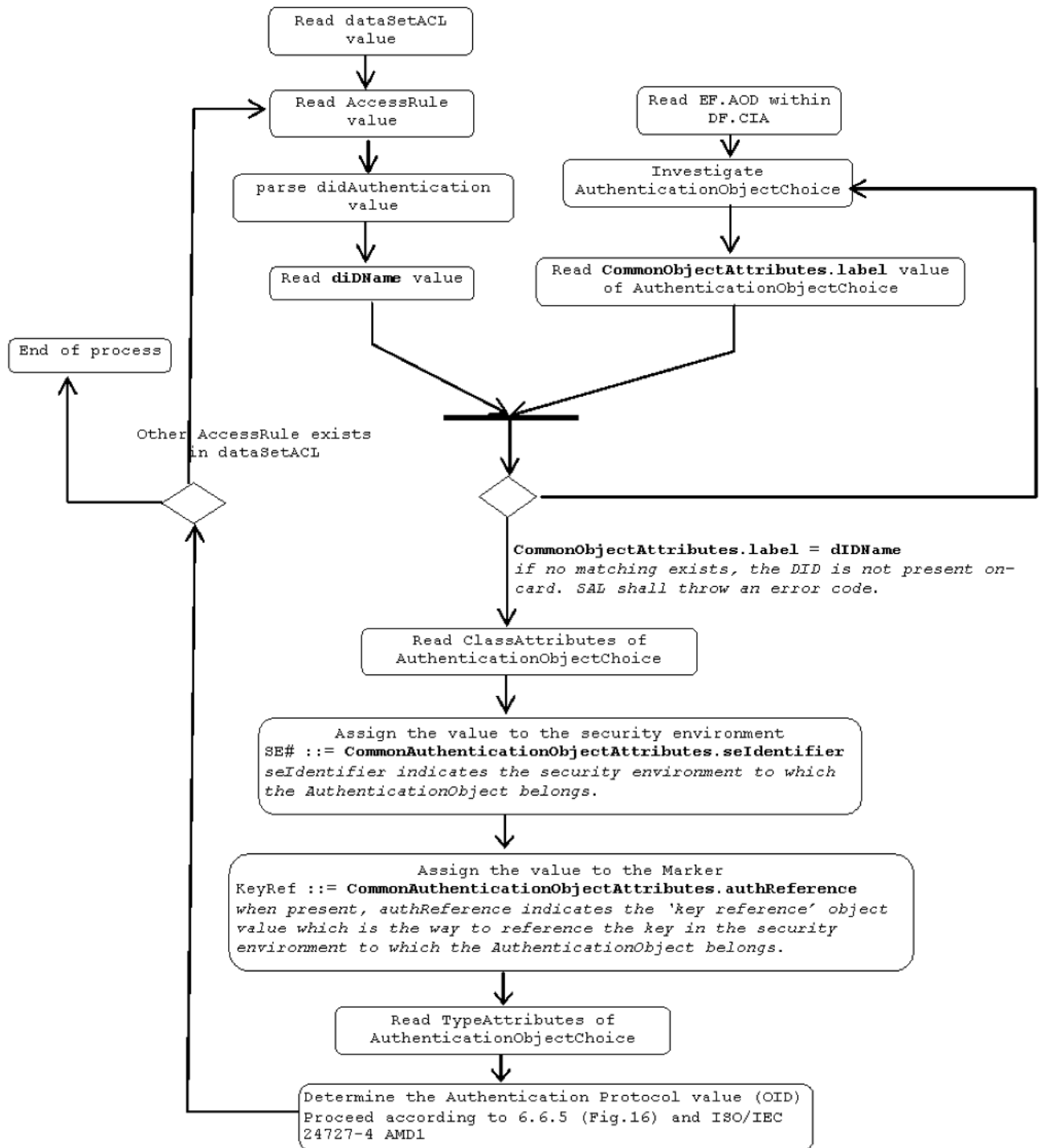


Figure C.3 — SAL implementation processing steps for the mapping of didAuthentication

NOTE 3 The discovery of the Authentication protocol value described in ISO/IEC 24727-4:2008, 8.1.1.4 and Table 8 applies.

### C.3.2.3 Extended API request parameters

In case a CIA implementation is not available on-card, the canonical protocol cannot take place and this clause describes an alternative based on extended parameters employed in the DataSetCreate request. DataSetCreate request conveys a nested input parameter *didName* that encodes according to ASN.1 definition as **dataSetACL.AccessRule.securityCondition.didAuthentication.didName**.

This *diDName* refers to a Differential-Identity of which the OPTIONAL parameter DIDQualifier is described as follows:

```
DIDQualifier ::= CHOICE {  
  applicationIdentifier ApplicationIdentifier,  
  objectIdentifier OBJECT IDENTIFIER,  
  applicationFunction BIT STRING  
}
```

The **DIDQualifier.applicationFunction** parameter may denote the security environment value as a BIT STRING encoded over one nibble, e.g:

0010 means SE#2 that encodes in BER-TLV as 03 02 04 20

This alternative entails that the referenced security environment is already personalised onto the card. But the client-application does not need to know this reference beforehand prior to execute a DataSetCreate call. The client-application can proceed as described in F.2.1.3 (informative use case) to discover the appropriate security environment value. Then, the client-application may call a DataSetCreate action that conveys a diDName input parameter referencing the appropriate security environment identifier. The SAL can therefore map accordingly the dataSetACL onto FCP (File Control Parameters) for the CREATE FILE APDU command.

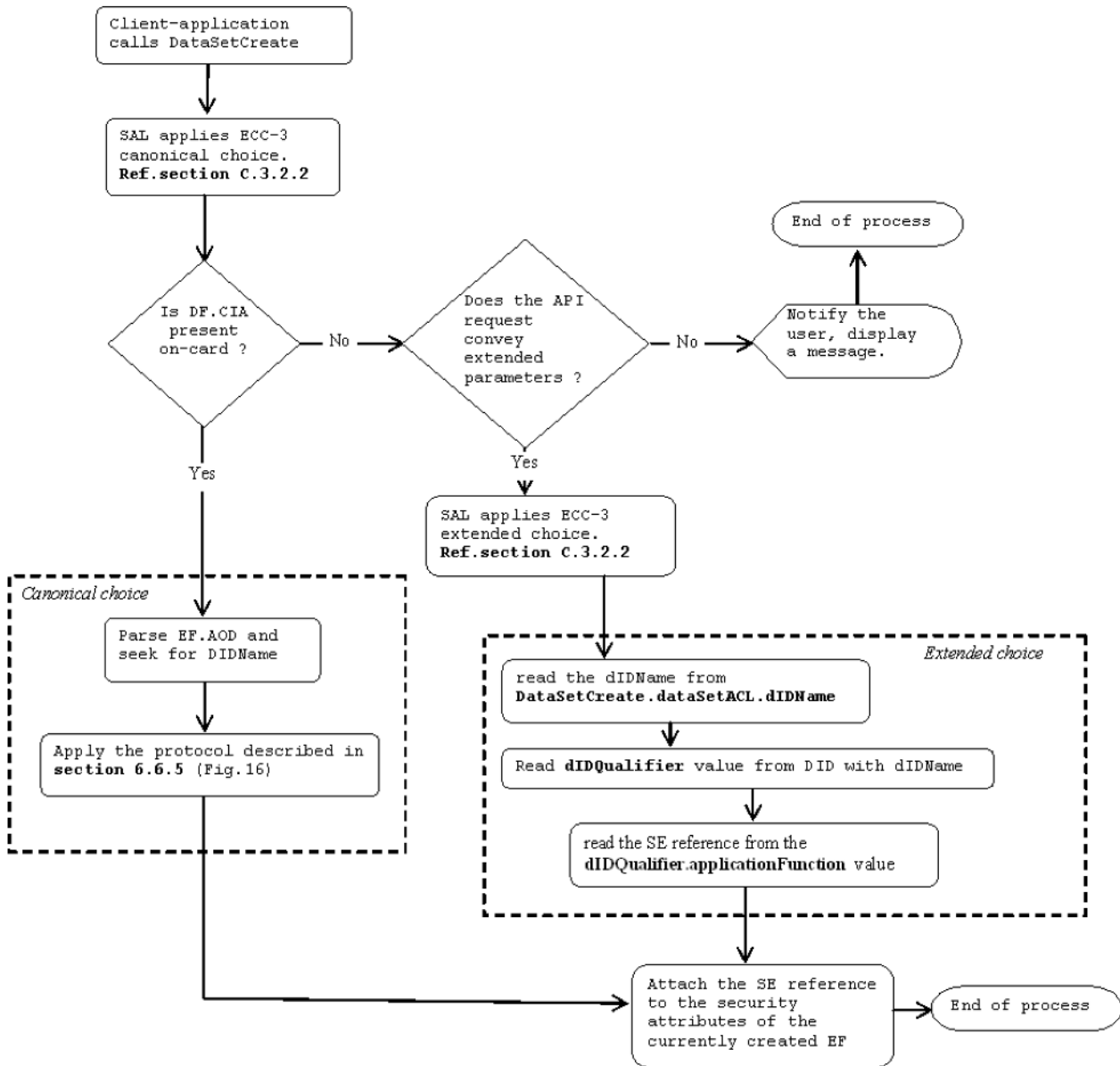
#### **C.3.2.4 Informative use case**

The following use case describes the main steps of the non-canonical choice based on the use of extended parameters for instance the diDQualifier value. The sequence of commands used in this use case are informative and other alternatives are possible depending on the context.

**Table C.1 — Selection by the client-application of the appropriate security environment**

<b>Client-application</b>	<b>API Request/Resp</b>	<b>SAL</b>
CardApplicationConnect()	→ ←	
ACLList() Uncover the access rules applying to DataSetCreate action	→	
DIDlist() uncover the list of DIDs to find out the DIDs referencing security environments	→ ←	Return the list of DIDs
DIDGet() retrieve one or more DIDs	→ ←	Return the contents of Named DID
Parse the DIDQualifier and select accordingly a DID to use in the DataSetCreate() request		
DIDAuthenticate() optionally get rights to call DataSetCreate function	→ ←	Authenticate the DID
DataSetCreate()	→ ←	Check whether the <b>DIDQualifier</b> from within the <b>dataSetACL</b> brings a security environment reference and attach it to the security attributes of the newly created file.

The activity diagram below illustrates the detailed decision tree for the SAL applied to DataSetCreate request.



**Figure C.4 — SAL complete decision tree applied for 'DataSetCreate'**

**C.3.3 DSICreate**

**C.3.3.1 General**

Input parameters: connectionHandle, dsiName, dsiContent

**C.3.3.2 Parameter mapping onto ISO/IEC 7816-4**

The APDU result of the SAL processing described in this clause and addressed at GCI shall be UPDATE BINARY. The characteristics of this UPDATE BINARY command are determined in this subclause.

**Odd INS**

Enhanced UPDATE BINARY:

**Table C.2 — UPDATE BINARY command-response pair (odd instruction code)**

Command Parameter	Meaning
CLA	ISO
INS	'D7'
P1-P2	As indicated Table C.4 — P1-P2 coding when B1 of INS = 0b1 I.4
L <sub>c</sub> field	Length of data field
Data field	'54' '02' {Offset data object in file}    '73' L <sub>73</sub> {Discretionary data object}
L <sub>e</sub> field	Absent

Response Parameter	Meaning
Data field	Absent
SW1-SW2	See Table C.3 — SW1-SW2 for UPDATE BINARY command I.3

**Table C.3 — SW1-SW2 for UPDATE BINARY command**

	Meaning
SW1-SW2	'6700' - Wrong length; no further indication '6981' - Command incompatible with file structure '6982' - Security status not satisfied '6986' - Command not allowed (no current EF) '6A82' - File not found '6B00' - Wrong parameters P1-P2 (offset + length causes update outside the file)

**Table C.4 — P1-P2 coding when B1 of INS = 0b1**

P1								P2							
B8	B7	B6	B5	B4	B3	B2	B1	B8	B7	B6	B5	B4	B3	B2	B1
0	0	0	0	0	0	0	0	0	0	0	Short File Identifier 1 ≤ SFI ≤ 30				
Identifier of the file where to apply the command. P1-P2 = '0000' means current file															

mapping per request parameter:

- connectionHandle: not applicable
- dsiName: discretionary data object value of type OCTET STRING in UPDATE BINARY command Data Field. The dsiName shall be nested in DO '16' identifying the IA5String ASN.1 type that is derived from OCTET STRING. The APDU command Data Field shall be as follows according to the canonical choice: '54' '02' {Offset data object in file} || '73' L<sub>73</sub> {'16' L<sub>16</sub> dsiName || Discretionary data object}

- dsiContent: OCTET STRING that shall be encapsulated in a DO '04'. The APDU command Data Field shall be as follows according to the canonical choice: '54' '02' {Offset data object in file} || '73' L<sub>73</sub> {'16' L<sub>16</sub> dsiName || '06' L<sub>06</sub> dsiContent}

The UPDATE BINARY command shall be executed in the context of the file just created with CREATE FILE (EF)

### **C.3.4 DIDCreate**

#### **C.3.4.1 General**

This request can be performed according to two different use cases:

- The case where a new cryptographic object is securely loaded onto the card and assigned to a Differential-Identity that is currently being created along with an existing authentication protocol.
- The case where an existing cryptographic object, already present in the card, is assigned a new role respective to the Differential-Identity that is currently being created, along with an existing authentication protocol.

Uniquely the latter case is considered in the present specification. For this purpose, the DIDCreate request shall convey all the parameters necessary to build the links between the existing cryptographic object and the CardApplicationServiceDescription value encoding a DifferentialIdentity service.

DIDCreate Input parameters: connectionHandle, didName, authProtocolOID, didUpdateData, dataSetACL

#### **C.3.4.2 Parameter mapping onto ISO/IEC 7816 (all parts)**

The APDU result of the SAL processing described in this clause and addressed at GCI shall be CREATE FILE (EF) with INS = 'E0' and P1≠'00'. The Data Field content of this CREATE FILE command is determined in this clause.

Mapping per request parameter:

- a) connectionHandle: not applicable
- b) didName: no restriction
- c) authProtocolOID: shall refer to an authentication protocol supported by the card.
  - 1) The SAL shall seek the authentication protocols implemented by the card in the DF.CIA CIAInfo file. If the equivalence **authProtocolOID = CIAInfo.supportedAlgorithms.objId** is not verified, the DIDCreate request shall be rejected.
- d) didUpdateData: this parameter is defined as DIDUpdateData:: = SEQUENCE {
  - marker OCTET STRING,
  - qualifier DIDQualifier OPTIONAL}
  - 1) The sub-parameter **marker** shall indicate the cryptographic object reference that valuates to **CommonObjectAttributes.authId** of an existing CIO in the card application.
  - 2) The sub-parameter **qualifier** shall indicate the **ApplicationIdentifier** of the card application to which the **marker** belongs.
- e) dataSetACL:



- 1) sub-parameter **cardApplicationServiceName** = "DifferentialIdentity" (named service)
- 2) sub-parameter **action** = a set of SAL-API requests acting upon the CardApplication within which the Differential-Identity is currently being created can be mapped as follows:

SAL-API action	GCI APDU	Canonical choice
DIDList	GET DATA	Cached in SAL (see NOTE)
	READ BINARY	
DIDCreate	PUT DATA	X
DIDGet	READ BINARY	Cached in SAL
	GET DATA	
DIDUpdate	PUT DATA	
	UPDATE BINARY	X
DIDDelete	UPDATE BINARY	X
DIDAuthenticate	N/A	Protocol acc. ISO/IEC 24727, Annex A

NOTE "Cached in SAL" means that the request result returned by the SAL may already be available among the meta-data generated on the fly by the SAL as part of the discovery mechanism. Therefore there is no need for an APDU command to recover the request result.

- 3) sub-parameter **securityCondition** = mapped onto ISO/IEC 7816-4 security condition byte as follows:

SecurityCondition as per ISO/IEC 24727-3:2008, C.1.3	Security condition byte	Canonical choice	using extended parameters (see C.3.2.2)
ALWAYS	No condition	X	
NEVER	Never	X	
N/A	b4-b1 = '0'		
didAuthentication	Security Environment identifier	X	X
N/A	b4-b1 = '1' (RFU)		
OR	At least one condition	X	
AND	All conditions	X	
didAuthentication	Secure Messaging	X	
didAuthentication	External Authentication	X	
didAuthentication	User Authentication	X	

### C.3.5 DIDUpdate

Specified uniquely in the context of an authentication protocol as per the template in ISO/IEC 24727-3:2008, Annex A.

### C.3.6 CardApplicationServiceCreate

#### C.3.6.1 General

CardApplicationServiceCreate Input parameters: connectionHandle, cardApplicationServiceName

#### C.3.6.2 Parameters mapping onto ISO/IEC 7816 (all parts)

The APDU result of the SAL processing described in this clause and addressed at GCI shall be PUT DATA with INS = 'DB' and P1-P2 = '3FFF' (for current DF). The Data Field content of this PUT DATA command shall convey the ACD container once updated with the addition of the named card-application service that is newly created.

In case the ACD is not stored onto the card and hosted instead on a remote server or at SAL level, no APDU command is required as a result of the SAL processing of CardApplicationServiceCreate.

Mapping per request parameter:

— connectionHandle: not applicable

#### C.3.6.3 cardApplicationServiceName

Possible alternatives are amongst the Card-Application named services defined in ISO/IEC 24727-3:2008, C.1.4. Accordingly, a service can bear the name "Connection", "CardApplication", "NamedData", "Cryptographic", "DifferentialIdentity" or "Authorisation".

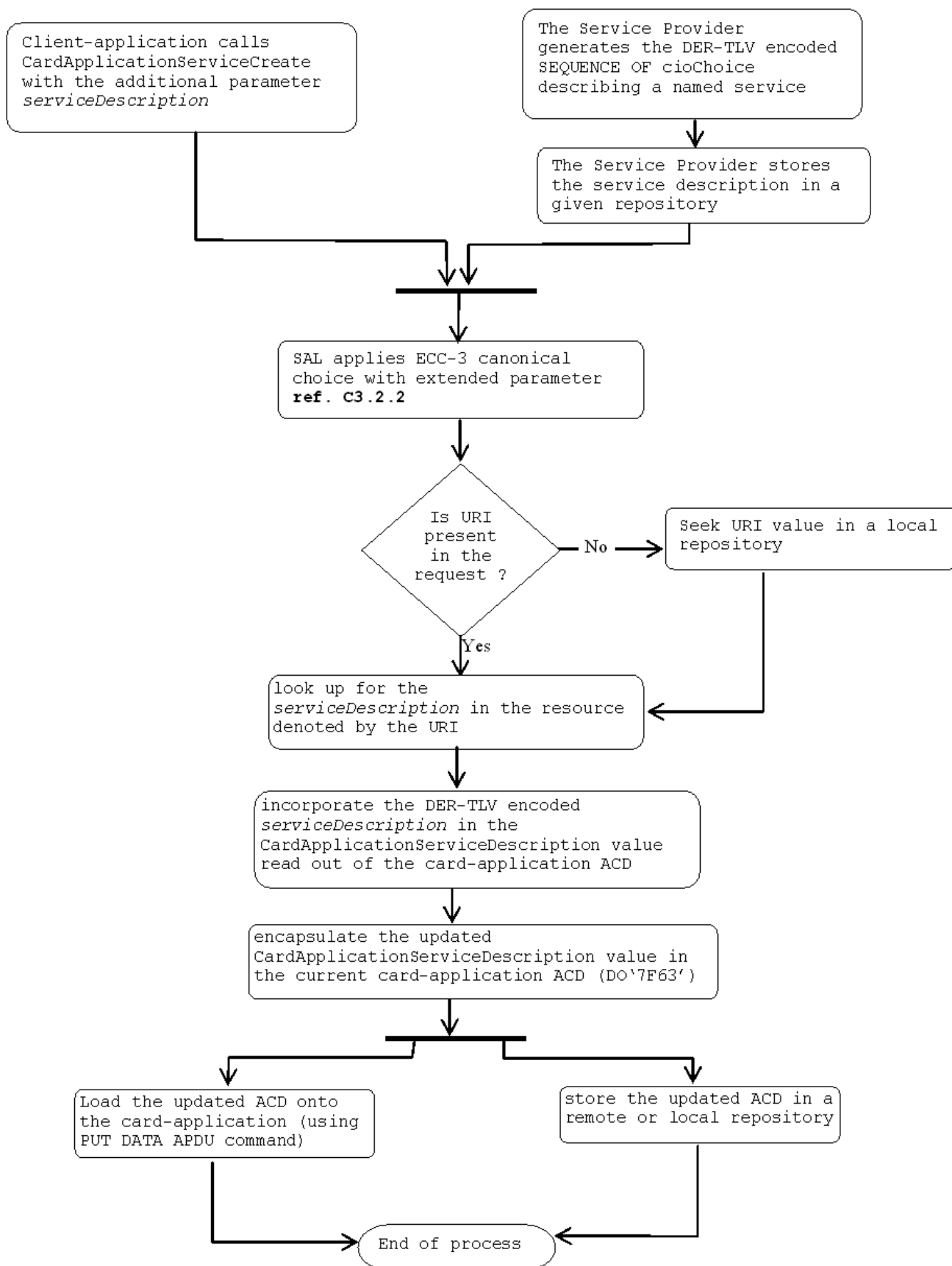
For the CardApplicationServiceCreate request to generate a new named service within the context of a card-application, it shall supply the SAL implementation layer with the complete description of such a service.

A service description may be a further parameter defined as a URI Type that points at the resource containing the DER-TLV encoded cioChoice describing the named service. This encoded description shall conform to ISO/IEC 24727-2:2008, Annex C.

The CardApplicationServiceCreate request with the additional parameter (*serviceDescription*) is as follows:

```
OUT ReturnCode CardApplicationServiceCreate (  
IN      ConnectionHandle      connectionHandle,  
IN      CardApplicationServiceName cardApplicationServiceName,  
IN      URI Type              serviceDescription  
);
```

The processing steps are described in the diagram on Figure C.5 — SAL implementation for the creation of a new named service I.5.



**Figure C.5 — SAL implementation for the creation of a new named service**

## **C.4 General recommendation and conclusion**

The complexity of data structures post-issuance management according to ISO/IEC 24727 rules that was explored in the present annex leads to the conclusion that a so-called canonical approach with its constraints, even though it leverages interoperability, seems less interesting than a per-card case approach. Based on card specifics, a more straightforward alternative may be the use of a Registry fed to the SAL implementation component and providing instructions for the Procedural Element to translate SAL API calls (i.e. including post-issuance card management calls) into card-specific commands according issuer's rules. This operation is usually performed by a "translation code". The Registry is either XML-based CardInfo file or [ISO/IEC 7816-15]-based DER-TLV Data Object. Such a Registry requires respectively an enhanced XML interpreter or DER-TLV encoder/decoder running at SAL level. As an example, the [ISO/IEC 7816-15]-based Registry comprises:

- card data structures and services description (see ISO/IEC 24727-4:2008, 8.1.1, ISO/IEC 24727-3 Data structures mapping)
- translation code (see ISO/IEC 24727-4:2008, 8.1.3 Card-specific APDU mapping onto ISO/IEC attributes)

The main advantages of using a translation code at SAL level is that it allows getting rid of intermediary generic card command set (i.e; GCI) and that it does not require from the middleware to discover the card access rules (i.e. ISO/IEC 7816-4 based) and to derive out of it the appropriate mapping: these information being provided directly by the Registry.

## Annex D (informative)

### Additional features versus ISO/IEC 24727 (all parts)

#### D.1 General

This annex provides the list of further enhancements adopted by the present standard comparing to ISO/IEC 24727 (all parts). These additional features do not impede compatibility with ISO/IEC 24727 (all parts); they solve instead issues that were not part of the scope of ISO/IEC 24727 (all parts).

#### D.2 Discovery Mechanism

- a) Decision Tree (support for legacy cards)
  - 1) Addition of EF.ATR in the decision tree Figure 12 in 6.2
  - 2) Addition of historical bytes and initial data string processing according to the card service data byte are involved implicitly in Figure 12
  - 3) Decision Tree is completed with CardInfo file alternative
- b) Discovery Information is also present within EF.DIR (through profile identifier: OID#).

#### D.3 General Procedures (SAL)

##### a) Differences with ISO/IEC 24727-3 SAL-API functions:

- 1) CardApplicationPath

In ECC-3 the `PathSecurity`-element is inside the `ChannelHandle`-element such that the connection between the SAL and the IFD-Layer, which is established with the `EstablishContext`-function defined in ISO/IEC 24727-4, may be secured accordingly. This feature is especially important in Remote-ICC-scenarios.

The `CardApplication`-element inside the `CardApplicationPathType` is optional such that it is possible to list all available cards regardless of their card applications.

- 2) CardApplicationConnect

In the `CardApplicationConnect` function defined in ECC-3 there is an additional optional parameter `Output`, which allows to output a message at an IFD, e.g. to ask the user to insert a specific card.

- 3) CardApplicationDisconnect

The optional `Action` parameter in ECC-3 is of type `ActionType` as defined in ISO/IEC 24727-4 and there is no separate `ReaderAction`-type as suggested in ISO/IEC 24727-3.

- 4) CardApplicationServiceCreate

With additional `serviceDescription` input parameter

- 5) CardApplicationDelete, CardApplicationServiceList, CardApplicationServiceCreate,  
CardApplicationServiceDelete, CardApplicationServiceLoad, CardApplicationServiceDelete,  
ExecuteAction and ACLModify

ECC-3 defines the CardApplicationServiceName element as string and hence is open for additional services.

- 6) ACLList and ACLModify

Unlike in ISO/IEC 24727-3 there is no explicit parameter TargetType in ECC-3, as the type of the target is implicitly clear from the chosen alternative within the TargetName-element, which is defined as choice.

- 7) Updating of the CardApplicationServiceDescription value whenever a change occurs at runtime on this value.

SAL layer returns the change status to a remote server where the update may take place before the CardApplicationServiceDescription is sent back to the SAL for loading onto the card.

SAL undertakes the update of CardApplicationServiceDescription value and loads it onto the card.

**b) Differences between the Access Control List structures defined in ISO/IEC 24727-3 and ECC-3:**

- 1) CardApplicationServiceName

While ISO/IEC 24727-3 only allows the standardised card application service names, ECC-3 allows to use arbitrary strings in the CardApplicationServiceName-element.

- 2) Action

In a similar fashion ISO/IEC 24727-3 only allows the standardised action names defined in the ActionName-structure, but the ActionNameType in ECC-3 also allows to use the string-based LoadedAction-alternative.

- 3) DIDAuthentication

The DIDScope-element in the DIDAuthentication-element is optional in ECC-3, as it is only necessary if there is a potential name conflict. Furthermore ECC-3 allows to include an optional DIDStateQualifier-element, which MAY contain a required certificate holder authorization template.

**c) Post-issuance personalisation:**

Two alternatives are explored for the ECC-3 middleware with as a conclusion, a preferred option using a Registry containing translation code:

- 1) the canonical/extended choices (off-line): *alternative 1*

Mapping of commands and related security rules

Basic principle: starting from the didName present in the request ACL (dataSetACL), the CommonObjectAttributes look up the DID, the ClassAttributes provide the SE identifier and the Marker reference, the TypeAttributes provide the Authentication Protocol OID. Consistency between the information shall be checked by the SAL.

- 2) the XML-based Card-Info parsing (off-line/online): *alternative 2*

Mapping of commands and related security rules

The generic XMLSchema definition shall serve for both ECC-4 Global profile and for any ECC-4 card profiles.

## D.4 Architecture

- Combination of Remote-Loyal and Remote-ICC stack, distributed model, authentication protocols may use multiple stack configurations
- No support of procedural elements running in GCAL
- A new function available to the Client-Application: '**ExecuteSAL()**' is specified as a single entry point to spare ASN.1 encoding/decoding to the client-application and to convey a XML-based payload instead.
- **ExecuteSAL()** C-language Binding is specified
- C-language Binding for the IFD-API is specified
- Completion of IFD-API: the differentiating features are presented in this standards and the commonalities are as per ISO/IEC 24727-4

## D.5 Differences between IFD-API in ISO/IEC 24727-4 and ECC-3

### D.5.1 More generale SlotCapabilityType

#### D.5.1.1 General

In the definition of the `SlotCapabilityType` in ISO/IEC 24727-4 it is only possible to indicate with the Element `ContactBased` of type `BooleanType`, whether the slot is contact-based or not. But it is not possible to indicate the precise types of supported transport protocols supported by the slot and to indicate that the slot is not meant to connect to external cards, but represents a security access module (SAM), which may be addressed in the `samConnectionHandle`-parameter of the SAL-functions `DIDAuthenticate` and `CardApplicationStartSession`.

In order to overcome this limitation the `SlotCapabilityType` defined in the present standard may contain a list of `Protocol`-elements, which may contain any URI, instead of the Boolean parameter `ContactBased`.

In case of a SAM the URI in the `Protocol`-element MAY be any DID-protocol. In case of a "regular" slot the URI corresponds to the transport protocol supported by the slot and the values defined for the `<Interface>`-element in 9.5 may be used.

#### D.5.1.2 IFD-API enhancement

##### D.5.1.2.1 General

`SlotCapabilityType` is an output parameter of IFD-API function `GetIFDCapabilities` and is enhanced as follows:

##### D.5.1.2.2 GetIFDCapabilities

- Purpose

The `GetIFDCapabilities` request returns information about the capabilities of a specific IFD and its associated functional units.

- Action

```

OUT   Status           GetIFDCapabilities (
IN    ContextHandleType ContextHandle,
IN    string           IFDName,
OUT   IFDCapabilitiesType IFDCapabilities
);

```

— Parameters

- ContextHandle addresses the established context with the IFD-Layer.
- IFDName is the unique name of the IFD, which is used to address the specific IFD.
- IFDCapabilities contains information about the capabilities of the given IFD and its functional units. It is of type IFDCapabilitiesType, which is structured as follows:

```

structure
{
    SlotCapabilityType           SlotCapability[ ],
    DisplayCapabilityType       DisplayCapability[ ],
    KeyPadCapapbilityType       KeyPadCapability[ ],
    BioSensorCapabilityType     BioSensorCapability[ ],
    BooleanType                 OpticalSignalUnit,
    BooleanType                 AcousticSignalUnit
} IFDCapabilitiesType

```

SlotCapability is present for every existing slot of the IFD and contains information about this slot. It is of type SlotCapabilityType, which is structured as follows:

```

structure
{
    nonNegativeInteger Index,
    URIType             Protocol[]
} SlotCapabilityType

```

Index – is the index of the slot ranging from 0 to number of slots minus 1.

ContactBased Protocol – may be present in several instances indicating the supported transport protocols; in case of a “regular slot” or the supported DID-protocols (see Section 8); in case the “slot” in fact is a security access module (SAM). indicates whether the slot is for contact-based or contactless cards.

DisplayCapability is present for every existing display of the IFD and contains information its capabilities. It is of type DisplayCapabilityType, which is structured as follows:

```

structure
{
    nonNegativeInteger Index,
    nonNegativeInteger Lines,
    nonNegativeInteger Columns,
    nonNegativeInteger VirtualLines
    OPTIONAL,
    nonNegativeInteger VirtualColumns
}

```



OPTIONAL

```
}DisplayCapabilitiesType
```

**Index** – is the index of the display ranging from 0 to number of displays minus 1.

**Lines** – contains the number of visible lines supported by the display of the card terminal.

**Columns** – contains the number of visible columns supported by the display of the card terminal.

**VirtualLines** – optionally contains the number of virtual lines supported by the display of the card terminal via scrolling.

**VirtualColumns** – optionally contains the number of virtual columns supported by the display of the card terminal via panning.

KeyPadCapability

is present for every existing key pad of the IFD and contains information about its capabilities. It is of type KeyPadCapabilityType, which is structured as follows:

```
structure
{
    nonNegativeInteger    Index,
    positiveInteger       Keys
}KeyPadCapabilitiesType
```

**Index** – is the index of the key pad ranging from 0 to number of key pads minus 1.

**Keys** – contains the number of keys on the key pad

BioSensorCapability

is present for every existing biometric sensor of the the IFD and contains information about the capabilities of this biometric sensor. It is of type BioSensorCapabilityType, which is structured as follows:

```
structure
{
    nonNegativeInteger    Index,
    nonNegativeInteger    BiometricType
} BioSensorCapabilityType
```

**Index** – is the index of the biometric sensor ranging from 0 to number of biometric sensors minus 1.

**BiometricType** – indicates the type of the biometric sensor as defined in 7.8 of ISO/IEC 19784-1.

OpticalSignalUnit

indicates whether an optical signal unit (e.g. LED) is available in the card terminal.

AcousticSignalUnit

indicates whether an acoustic signal unit (e.g. beep) is available in the card terminal.

— Return Codes

IFD_OK	The request was successful.
IFD_INVALID_CONTEXT_HANDLE	The provided ContextHandle is invalid.
IFD_UNKNOWN_IFD	The provided IFDName is unknown.

API\_UNKNOWN\_ERROR                      There was some unknown error.

**D.5.1.3 ISOIFD.XSD definition enhancement**

```
<complexType name="SlotCapabilityType">
  <sequence>
    <element name="Index" type="nonNegativeInteger" maxOccurs="1"
minOccurs="1"/>
    <element name="Protocol" type="anyURI" maxOccurs="unbounded"
minOccurs="0"/></element>
  </sequence>
</complexType>
```

**D.5.2 Transmit with support for batch processing**

**D.5.2.1 General**

Unlike in ISO/IEC 24727-4 the `Transmit` function is able to send a batch of APDUs to the IFD-Layer, which in turn sends the APDUs sequentially to the connected card. In order to support the batch processing a set of `AcceptableStatusCode`-elements (9000, etc.) may be attached to each `InputAPDU`. If the card returns some not expected status code it is – even in case of secure messaging - clear that there is a serious error and it does not make sense to feed further `InputAPDU`-elements in the batch to the card.

**D.5.2.2 IFD-API enhancement**

**D.5.2.2.1 General**

`Transmit` is an IFD-API function enhanced as follows.

**D.5.2.2.2 Transmit**

— Purpose

APDU(s) to a connected card. In order to support the batch processing, a set of `AcceptableStatusCode`-elements (9000, etc.) MAY be attached to each `InputAPDU`. If the card returns some not expected status code it is – even in case of secure messaging - clear that there is a serious error and it does not make sense to feed further `InputAPDU`-elements in the batch to the card.

— Action

```
OUT   Status           Transmit (
IN    SlotHandleType   SlotHandle,
IN    InputAPDUInfoType InputAPDUInfo[],
OUT   OCTET STRING    OutputAPDU[],
);
```

— Parameters

- `SlotHandle`            is a handle identifying the connection to the smart card.
- `InputAPDUInfo`       MAY be present multiple times and contains the command APDU, which will be sent to the smart card and optionally a set of acceptable status codes. The `InputAPDUInfoType` is structured as follows:

structure

```
{
    OCTET STRING      InputAPDU,
    OCTET STRING      AcceptableStatusCode[ ] OPTIONAL
} InputAPDUInfoType
```

InputAPDU contains the command APDU, which will be sent to the smart card.

AcceptableStatusCode MAY contain a set of acceptable status codes. If the status code which is returned from the smart card is not among the expected values the batch processing SHOULD be stopped as this indicates that there is a serious error condition.

If the AcceptableStatusCode-element is omitted, any returned status code is assumed to be acceptable

OutputAPDU is a byte string containing a response APDU from the card. There SHALL be exactly as many InputAPDU- as OutputAPDU-elements.

— Return Codes

- IFD\_OK The request was successful.
- IFD\_INVALID\_CARD\_HANDLE The provided SlotHandle is invalid.
- IFD\_UNKNOWN\_ERROR There was some unknown error.

**D.5.2.3 ISOIFD.XSD definition enhancement**

```
<!-- Transmit -->
<element name="Transmit">
  <complexType>
    <complexContent>
      <extension base="iso:RequestType">
        <sequence>
          <element name="SlotHandle"
            type="iso:SlotHandleType" />
          <element name="InputAPDUInfo"
            type="iso:InputAPDUInfoType" maxOccurs="unbounded"
            minOccurs="1">
            </element>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="InputAPDUInfoType">
    <sequence>
      <element name="InputAPDU"
        type="hexBinary">
        </element>
      <element name="AcceptableStatusCode"
        type="hexBinary" maxOccurs="unbounded" minOccurs="0">
        </element>
    </sequence>
```

```
</complexType>
<element name="TransmitResponse">
  <complexType>
    <complexContent>
      <extension base="iso:ResponseType">
        <sequence>
          <element name="OutputAPDU" type="hexBinary"
maxOccurs="unbounded" minOccurs="1"></element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
```

NOTE Related web service binding definitions (.WSDL) for Transmit function remain unchanged.

### **D.5.3 Additional error code for SignalEvent**

In addition to the error codes defined for `SignalEvent` in ISO/IEC 24727-4 there is an additional error code:

../resultminor/al/common#unknownError

## Annex E (informative)

### C-Language Binding for ExecuteSAL function

This function serves as C-language binding for SAL layer. It is the only entry point for C-programming language. It processes XML encoded SAL function call and returns XML encoded function call response.

#### Prototype

```
StatusType ExecuteSAL (
    IN wchar_t      *FunctionCall,
    OUT wchar_t     *Response,
    IN/OUT unsigned int *BufferSize
);
```

#### Parameters

- FunctionCall

XML encoded SAL function call. Null-terminated buffer containing the XML payload as defined in chapter 10 (XML-Binding for SAL-API functions)

- Response

XML encoded SAL function call response. Null-terminated buffer containing XML payload as defined in chapter 10 (XML-Binding for SAL-API functions).

- BufferSize

Size of Response buffer. This parameter serves as input for specifying the size (number of characters, not bytes) of application allocated buffer. If the function returns API\_SMALL\_BUFFER, this parameter is filled with the actual buffer size (number of characters, not bytes) necessary for storing response data.

#### Return values

Data Type	Description
API_OK	Function executed successfully.
API_SMALL_BUFFER	Application allocated buffer for response data are too small.
API_XML_ERROR	Error in parsing XML encoded function call.
API_GENERIC_ERROR	Unknown error.

## Annex F (informative)

### Java-Language Binding for ExecuteSAL function

SALJavaBinding class serves as Java-language binding for SAL layer. It has only one method ExecuteSAL which processes XML encoded SAL function call and returns XML encoded function call response.

#### Prototype

```
public class SALJavaBinding {  
    public static String ExecuteSAL (String functionCall) throws SALException {}  
}
```

#### Parameters

- functionCall

XML encoded SAL function call. String containing XML payload as defined in chapter 10 (XML-Binding for SAL-API functions).

#### Returns

XML encoded SAL function call response. String containing XML payload as defined in chapter 10 (XML-Binding for SAL-API functions).

#### Throws

SALException

```
public class SALException extends Exception {  
    public enum SALExceptionReason {ApiXmlError, ApiGenericError};  
    SALExceptionReason reason;  
    public SALException(SALExceptionReason reason) {  
        this.reason = reason;  
    }  
    public SALExceptionReason getReason() {  
        return reason;  
    }  
    public String getMessage() {  
        switch (reason) {  
            case ApiXmlError:  
                return "Error in parsing XML encoded function call.";  
            default:  
                return "Generic error.";  
        }  
    }  
}
```

## **Annex G** (informative)

### **Application Discovery Profile: card requirements to access/offer services in ISO/IEC 24727 framework**

#### **G.1 General**

This profile may be referred to from within Application Profiles described in ECC-2 (CEN/TS 15480-2). See ECC-4 for OID assignment rules.

#### **G.2 OID**

The OID assigned to this profile is: 1.3.162.15480.4.0.1.

The Profile OID value is nested in EF.DIR as described in 6.6.2.

#### **G.3 General**

The objective of this profile is to determine the set of features and data structures that are to be supported by a card to fit in a ISO/IEC 24727 framework. The general requirements comprehend the card capability to support the bootstrap procedure allowing the subsequent discovery mechanism to take place.

#### **G.4 interfaces / transport protocols**

The mandatory interfaces and transport protocol are described in Table G.1 — Supported Interfaces:

**Table G.1 — Supported Interfaces**

<b>Application</b>	<b>Interface/Protocol</b>	<b>Platform support</b>	<b>Perso/activation before issuance</b>	<b>Remarks</b>
Alpha card-application	Contact or Contactless according to ISO/IEC 14443 (all parts) with randomized UID	○	○	Except for ICC-Resident-stack, the alpha card-application may either be present on-card or emulated by the SAL-implementation. For ICC-Resident-stack, alpha card-application shall be embedded on-card.

## G.5 Data elements and data structures

**Table G.2 —Mandatory and conditional data elements and data structures**

Feature	Ref.	Platform support	Perso/activation before issuance	Comments
Transparent Elementary Files (EF)	CEN/TS 15480–2:2012, 5.2	M	O	<p>Occurrence of EF e.g. If CCD is present and contains a CIA-PROFILES (§ ISO/IEC 24727-2:2008, Table 14), the related cryptographic information application EFs shall either be present in the card or referred to with a URL from DO'5F50'</p> <p>If CCD is present and delivered through EF.ATR, the EF.ATR shall be present in the card.</p> <p>Cryptographic information application's EFs are not necessary when:</p> <ul style="list-style-type: none"> <li>- the card services are entirely encoded within the CardApplication-ServiceDescription value (ACD DO '7F66') without path reference to on-card ISO/IEC 7816-15 CIOChoices values.</li> <li>- the ACD contains the data object '7F67' nesting the URL that locates the resource outside the card containing the concatenation of ISO/IEC 7816-15 CIOChoice values.</li> </ul>
Application Dedicated Files (ADF)	CEN/TS 15480–2:2012, 5.2	M	O	Alpha card-application
ATR/ATS	CEN/TS 15480–2:2012 5.3	M	O	<p>May serve to denote data objects '7F63' (ACD) or '7F62' (CCD) within EF.ATR/INFO with card service data byte set properly in historical bytes.</p> <p>Interindustry data</p>



Feature	Ref.	Platform support	Perso/activation before issuance	Comments
				element 'initial access data' may be used to recover the data object '7F62' (CCD) as part of the initial data string within the response data field.
EF.ATR/INFO	CEN/TS 15480-2:2012 5.3	M	O	May serve to nest data object '7F62' (CCD)
Cryptographic Information Application	ISO/IEC 24727-4:2008, Clause 8 Registry Implementation	M	O	May serve to host cryptographic information objects referred to from the DER-encoded CardApplicationService-Description value or from a XML-based CardInfo file
ACD	ISO/IEC 24727-2:2008, 6.2 and 6.3, Tables 15 and 16	O	O	According to 6.3 of the present standard, either the ACD or a XML-based CardInfo-file may be retrieved from a remote repository.
CCD	ISO/IEC 24727-2:2008, 6.1, Table 14	O	O	According to 6.3 of the present standard, a minimum-CCD with a DO '5F50' pointing to a XML-based CardInfo-file resource may be present on-card.
Procedural elements (PE)	ISO/IEC 24727-2:2008, 6.3	O	O	No procedural elements compliant with ISO/IEC 20060 are present for ECC. <sup>5)</sup> The Registry (XML-based or ASN.1-based) contains a translation code and is processed by PE at SAL level.

---

5) There is further APDU commands defined in the GCI than in ECC set of interoperable commands; for each of those additional commands, a XML-based CardInfo-file may be employed : PUT DATA, DELETE FILE, CREATE FILE, ACTIVATE FILE, DEACTIVATE FILE, GENERATE ASSYMETRIC KEY PAIR, PSO ENCIPHER, PSO VERIFY DIGITAL SIGNATURE. All the rest of GCI APDU commands are compatible with ECC. See 5.2 for details about GCAL and GCI abstraction.

**G.6 Command set**

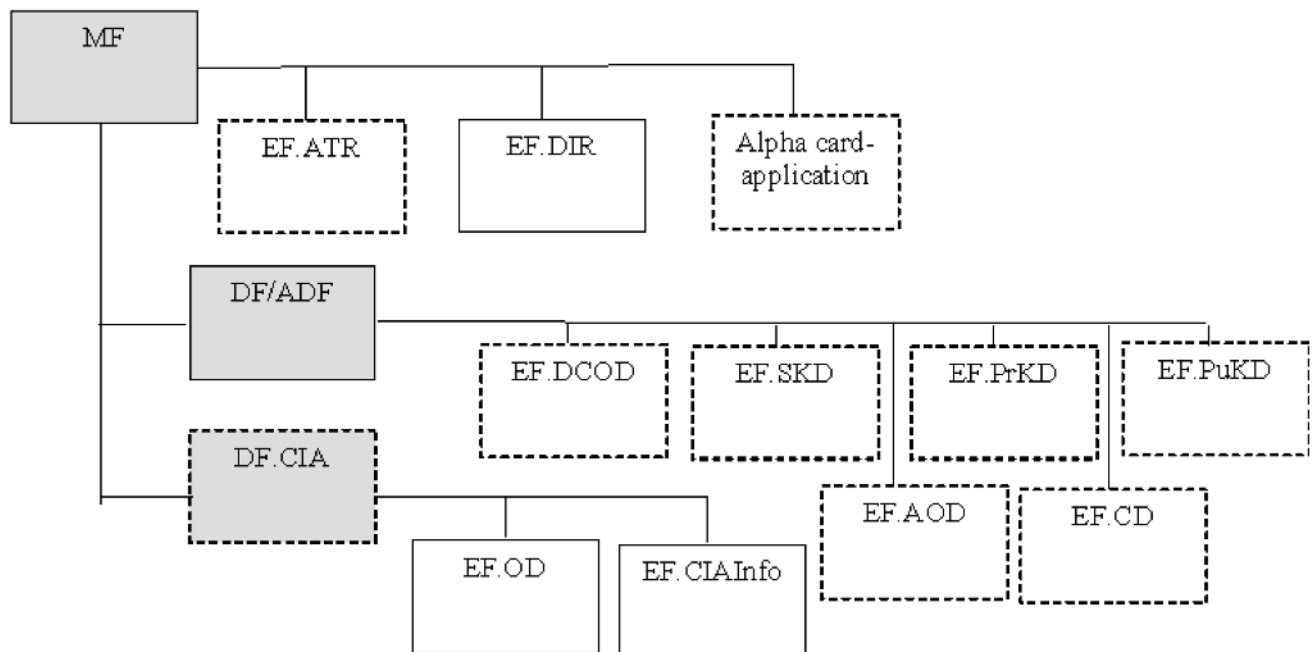
**Table G.3 — Mandatory and conditional commands**

Function	Application	Platform support	Perso/activation before issuance	Comments
SELECT	Alpha card-application, CIA files or any	M	M	<p>Alpha card-application shall be selectable by AID 'E8 28 81 C1 17 02' (ref. ISO/IEC 24727-2:2008, 5.5.1)</p> <p>Sequence of card-applications within CCD context-specific DO 'A0' shall be selectable.</p> <p>ACD may be retrieved upon card-application selection.</p>
GET DATA	Alpha card-application or ANY	O	O	<p>GET Data are not necessary if data object '7F62' (CCD) may be retrieved otherwise than with a GET DATA command.</p> <p>If upon selection of a card-application, the ACD can be read otherwise than with a GET DATA command, the GET DATA support is not necessary.</p> <p>if used, GET DATA shall support:</p> <ul style="list-style-type: none"> <li>a) P1-P2 = '7F62' and Le = '00' or P1-P2 = '3FFF' and command data field containing '5C027F62' for CCD recovery.</li> <li>b) P1-P2 = '7F63' and Le = '00' or P1-P2 = '3FFF' and command data field containing '5C027F63' for ACD recovery.</li> </ul>

Function	Application	Platform support	Perso/activation before issuance	Comments
READ BINARY	CIA files	M	M	Useful to read the CIA contents i.e. SAL Lite component may have to read the CIA files to generate out of it the data structures for interoperability.

## G.7 Data structure of Card Applications

### G.7.1 General



**Figure G.1 — Data Structure of the card**

Figure G.1 above depicts the minimum set of files necessary for a ECC card to fit in a ISO/IEC 24727-compliant environment.

### G.7.2 DF/ADF content

The card-application may contain a ACD unless the ACD is loaded from remote for legacy card support.

### G.7.3 EF DCOD content

If the CardApplicationServiceDescription value makes reference to on-card CIA objects then the EF.DCOD file shall contain DER-TLV encoded DataContainerObjectChoice(s) that are referred to. A DataContainerObjectChoice container may represent CardApplication target<sup>6)</sup> or DataSet target.

6) For further definition of *Target*, see ISO/IEC 24727-4:2008, G.1.1.1 (Computational Model based on Targets), 8.1.1.3 (Card Application target) and 8.1.1.1 (DataSet target).

#### **G.7.4 EF AOD content**

If the CardApplicationServiceDescription value makes reference to on-card CIA authentication objects then the EF.AOD file shall contain the AuthenticationObjectChoice(s) mapping Differential-Identities for authentication and encoded according to ISO/IEC 24727-4:2008, 8.1.1.4 and Table 8-4.

#### **G.7.5 EF SKD content**

If the CardApplicationServiceDescription value makes reference to on-card CIA secret key objects then the EF.SKD shall contain the KeyObjectChoice(s) mapping Differential-Identities used for the card-application cryptographic methods (Encipher, Decipher, Sign,...). These KeyObjectChoice(s) shall be encoded according ISO/IEC 24727-4:2008, 8.1.1.4 and Table 8-5.

#### **G.7.6 Ef PrKD content**

If the CardApplicationServiceDescription value makes reference to on-card CIA private key objects then the EF.PrKD shall contain the KeyObjectChoice(s) mapping Differential-Identities used for the card-application cryptographic methods (Encipher, Decipher, Sign, ...). These KeyObjectChoice(s) shall be encoded according ISO/IEC 24727-4:2008, 8.1.1.4 and Table 8-6.

## Bibliography

- [1] EN 1332-4, *Identification card systems — Man-machine interface — Part 4: Coding of user requirements for people with special needs*
- [2] EN 14890-1, *Application Interface for smart cards used as Secure Signature Creation Devices — Part 1: Basic services*
- [3] CEN/TS 15480-1, *Identification card systems — European Citizen Card — Part 1: Physical, electrical and transport protocol characteristics*
- [4] CEN/TS 15480-5, *Identification card systems — European Citizen Card — Part 5: General Introduction*
- [5] EN ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes (ISO 3166-1)*
- [6] ISO/IEC 7812-1, *Identification cards — Identification of issuers — Part 1: Numbering system*
- [7] ISO/IEC 7816 (all parts), *Identification cards — Integrated circuit cards*
- [8] ISO/IEC 9796-2, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms*
- [9] ISO/IEC 9798-3, *Information technology — Security techniques — Entity authentication — Part 3: Mechanisms using digital signature techniques*
- [10] ISO/IEC 10118-3<sup>7)</sup>, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*
- [11] ISO/IEC 14443 (all parts), *Identification cards — Contactless integrated circuit cards — Proximity cards*
- [12] ISO/IEC 15946-2, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 2: Digital signatures*
- [13] ISO/IEC 19794-2, *Information technology — Biometric data interchange formats — Part 2: Finger minutiae data*
- [14] ISO/IEC 20060, *Information technology — Open Terminal Architecture (OTA) — Virtual machine*
- [15] ANSI X9.63, *Public Key Cryptography For the Financial Services Industry — Key Agreement and Key Transport Using Elliptic Curve Cryptography*, January 8th 1999
- [16] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures
- [17] Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)

---

7) This document is currently impacted by the draft amendment ISO/IEC 10118-3:2004/Amd 1:2006.

- [18] International Civil Aviation Organization: *Machine readable travel document, Technical Report: Development of a logical data structure - LDS*, Revision 1.7, 18.05.2004
- [19] International Civil Aviation Organization: *Machine readable travel document, Technical Report: PKI for Machine Readable Travel Documents offering ICC read only access*, Revision 1.1, 01.10.2005
- [20] Smart Card Middleware – API specification (version 1.0), Onom@Topic+ project (MEDEA+), OKsystem s.r.o, Feb. 2007
- [21] RFC3280 R. HOUSLEY. W. Polk, W. Ford, D. Solo: *Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC 3280, via <http://www.ietf.org/rfc/rfc3280.txt>
- [22] RFC3281 S. FARRELL. R. Housley: *An Internet Attribute Certificate Profile for Authorization*, IETF RFC 3281, via <http://www.ietf.org/rfc/rfc3281.txt>
- [23] RFC 4492 S. Blake-Wilson, N. Bolyard, V. Guptam, C. Hawk, und B. Moeller: *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*, IETF RFC 4492, via <http://www.ietf.org/rfc/rfc4492.txt>
- [24] RFC 4346 T. Dierks, E. Rescorla: *The Transport Layer Security (TLS) Protocol, Version 1.1*, IETF RFC 4346, via <http://www.ietf.org/rfc/rfc4346.txt>
- [25] MS-CAPI MICROSOFT INC. *Cryptography Reference (Microsoft CryptoAPI), Platform SDK: Security*, via [http://msdn.microsoft.com/library/en-us/security/security/cryptography\\_reference.asp](http://msdn.microsoft.com/library/en-us/security/security/cryptography_reference.asp)
- [26] PAOS-v1.1 Liberty Alliance Project: *Liberty Reverse HTTP Binding for SOAP Specification*, Version v1.1, via <http://www.projectliberty.org/liberty/content/download/1219/7957/file/liberty-paos-v1.1.pdf>
- [27] SOAP-v1.1 W3C Note: *Simple Object Access Protocol (SOAP) 1.1*, 8 May 2000, via <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>
- [28] PKCS#11 v2.01 RSA Labs.: *PKCS #11: Cryptographic Token Interface Standard*, Version 2.20, 2004, <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf>
- [29] MS-CAPI MICROSOFT INC. *Cryptography Reference (Microsoft CryptoAPI), Platform SDK: Security*, via [http://msdn.microsoft.com/library/en-us/security/security/cryptography\\_reference.asp](http://msdn.microsoft.com/library/en-us/security/security/cryptography_reference.asp)
- [30] RFC 3275 D. Eastlage, J. Reagle, D. Solo: *(Extensible Markup Language) XMLSignature Syntax and Processing*, IETF RFC 3275, via <http://www.ietf.org/rfc/rfc3275.txt> SAML(v2.0)
- [31] CANTOR S., KEMP J., PHILPOTT R., MALER E. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15.03.2005.*, <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>, 2005
- [32] WS-POLICY. (v1.5) A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, Ü. Yalçınalp: *Web Services Policy 1.5 – Framework, W3C Recommendation 04 September 2007*, <http://www.w3.org/TR/ws-policy/>



# British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

## About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards-based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

## Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at [bsigroup.com/standards](http://bsigroup.com/standards) or contacting our Customer Services team or Knowledge Centre.

## Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at [bsigroup.com/shop](http://bsigroup.com/shop), where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

## Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to [bsigroup.com/subscriptions](http://bsigroup.com/subscriptions).

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

**PLUS** is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit [bsigroup.com/shop](http://bsigroup.com/shop).

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email [bsmusales@bsigroup.com](mailto:bsmusales@bsigroup.com).

## BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

## Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

## Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

## Useful Contacts:

### Customer Services

**Tel:** +44 845 086 9001

**Email (orders):** [orders@bsigroup.com](mailto:orders@bsigroup.com)

**Email (enquiries):** [cservices@bsigroup.com](mailto:cservices@bsigroup.com)

### Subscriptions

**Tel:** +44 845 086 9001

**Email:** [subscriptions@bsigroup.com](mailto:subscriptions@bsigroup.com)

### Knowledge Centre

**Tel:** +44 20 8996 7004

**Email:** [knowledgecentre@bsigroup.com](mailto:knowledgecentre@bsigroup.com)

### Copyright & Licensing

**Tel:** +44 20 8996 7070

**Email:** [copyright@bsigroup.com](mailto:copyright@bsigroup.com)



...making excellence a habit.™