
**Identification cards — Integrated circuit
card programming interfaces —**

**Part 3:
Application interface**

*Cartes d'identification — Interfaces programmables de cartes à puce —
Partie 3: Interface d'application*



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2008

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

Page

Foreword	v
Introduction.....	vi
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 Abbreviated terms	3
5 Organization for interoperability.....	4
5.1 General	4
5.2 Computation model.....	4
5.3 Entity relationships on the application interface	5
5.4 Security model	13
6 Card-application-service access	16
6.1 General	16
6.2 Initialize	16
6.3 Terminate	17
6.4 CardApplicationPath	18
7 Connection service	19
7.1 General	19
7.2 CardApplicationConnect	20
7.3 CardApplicationDisconnect	21
7.4 CardApplicationStartSession.....	22
7.5 CardApplicationEndSession	23
8 Card-application service.....	24
8.1 General	24
8.2 CardApplicationList	25
8.3 CardApplicationCreate.....	26
8.4 CardApplicationDelete	27
8.5 CardApplicationServiceList	28
8.6 CardApplicationServiceCreate.....	29
8.7 CardApplicationServiceLoad	30
8.8 CardApplicationServiceDelete	31
8.9 CardApplicationServiceDescribe.....	32
8.10 ExecuteAction.....	33
9 Named data service	34
9.1 General	34
9.2 DataSetList	35
9.3 DataSetCreate	36
9.4 DataSetSelect.....	37
9.5 DataSetDelete	38
9.6 DSIList	39
9.7 DSICreate	40
9.8 DSIDelete.....	41
9.9 DSIWrite.....	42
9.10 DSIRead	43
10 Cryptographic service.....	44
10.1 General	44
10.2 Encipher	45

10.3	Decipher.....	46
10.4	GetRandom.....	47
10.5	Hash	48
10.6	Sign	49
10.7	VerifySignature	50
10.8	VerifyCertificate	51
11	Differential-identity service.....	52
11.1	General.....	52
11.2	DIDList	53
11.3	DIDCreate.....	54
11.4	DIDGet.....	55
11.5	DIDUpdate.....	56
11.6	DIDDelete	57
11.7	DIDAuthenticate	58
12	Authorization service	59
12.1	General.....	59
12.2	ACLList	60
12.3	ACLModify	61
Annex A	(normative) Authentication protocols	62
A.1	General.....	62
A.2	Common Definitions.....	63
A.3	Simple Assertion.....	64
A.4	Asymmetric Internal Authenticate	66
A.5	Asymmetric External Authenticate	69
A.6	Symmetric Internal Authenticate.....	72
A.7	Symmetric External Authenticate	75
A.8	Compare	78
A.9	PIN Compare	81
A.10	Biometric Compare.....	84
A.11	Mutual Authentication with Key Establishment	87
A.12	Client-Application Mutual Authentication with Key Establishment	90
A.13	Client-Application Asymmetric External Authenticate	93
A.14	Modular Extended Access Control Protocol (M-EAC)	96
A.15	Key Transport with mutual authentication based on RSA	100
A.16	Age Attainment	104
A.17	Asymmetric Session Key Establishment	107
A.18	Secure PIN Compare	114
A.19	EC Key Agreement with Card-Application Authentication.....	118
A.20	EC Key Agreement with Mutual Authentication	122
A.21	Simple EC-DH Key Agreement	128
A.22	GP Asymmetric Authentication.....	132
A.23	GP Symmetric Authentication (Explicit Mode)	138
A.24	GP Symmetric Authentication (Implicit Mode)	142
Annex B	(normative) Cryptographic algorithms.....	145
B.1	Interoperability requirements	145
B.2	Symmetric Algorithms	146
B.3	Asymmetric Algorithms	149
B.4	Elliptic Curve Algorithms	150
B.5	Hash Functions	151
B.6	Message Authentication Codes	152
B.7	Key Establishment.....	153
Annex C	(normative) ASN.1 Representation	154
C.1	General.....	154
Bibliography		193

Foreword

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

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

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

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

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

ISO/IEC 24727 consists of the following parts, under the general title *Identification cards — Integrated circuit card programming interfaces*:

- *Part 1: Architecture*
- *Part 2: Generic card interface*
- *Part 3: Application interface*
- *Part 4: Application programming interface (API) administration*
- *Part 5: Testing*
- *Part 6: Registration authority procedures for the authentication protocols for interoperability*

Introduction

ISO/IEC 24727 is a set of programming interfaces for interactions between integrated circuit cards (ICCs) and external applications to include generic services for multi-sector use. The organization and the operation of the ICC conform to ISO/IEC 7816-4.

ISO/IEC 24727 is relevant to ICC applications desiring interoperability among diverse application domains. This part of ISO/IEC 24727 specifies a language-independent and implementation-independent application level interface that allows information and transaction interchange with a card. ISO/IEC 7498-1 is used as the layered architecture of the application interface. That is, the application interface assumes that there is a protocol stack through which it will exchange information and transactions among cards using commands conveyed through the message structures defined in ISO/IEC 7816. The semantics of commands accessed by the application interface refers to application protocol data units (APDUs) as characterized in ISO/IEC 24727-2, and in the following standards:

- ISO/IEC 7816-4, *Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*
- ISO/IEC 7816-8, *Identification cards — Integrated circuit cards — Part 8: Commands for security operations*
- ISO/IEC 7816-9, *Identification cards — Integrated circuit cards — Part 9: Commands for card management*

The goal of this part of ISO/IEC 24727 is to maximize the applicability and solution space of software tools that provide application interface support to card-aware applications. This effort includes supporting the evolution of card systems as the cards become more powerful, peer-level partners with existing and future applications while minimizing the impact to existing solutions conforming to this part of ISO/IEC 24727.

Identification cards — Integrated circuit card programming interfaces —

Part 3: Application interface

1 Scope

This part of ISO/IEC 24727 defines services as representations of action requests and action responses to be supported at the client-application service interface. The services are described in a programming-language-independent way.

This part of ISO/IEC 24727 is the application interface of the Open Systems Interconnection Reference Model defined in ISO/IEC 7498-1. It provides a high-level interface for a client-application making use of information storage and processing operations of a card-application as viewed on the generic card interface.

This part of ISO/IEC 24727 does not mandate a specific implementation methodology for this interface.

2 Normative references

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

ISO/IEC 7816-11, *Identification cards — Integrated circuit cards — Part 11: Personal verification through biometric methods*

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

ISO/IEC 24727-2, *Identification cards — Integrated circuit card programming interfaces — Part 2: Generic card interface*

IETF RFC 2141, *URN Syntax*, May 1997

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24727-1, ISO/IEC 24727-2 and the following apply.

3.1

access control list

set of access rules

3.2

access permission

granted capability to perform an action

- 3.3**
access rule
association of an action and a security condition in the context of a card-application
- 3.4**
action
operation that a client-application can request of a card-application at the ISO/IEC 24727-3 application interface
- 3.5**
alpha card-application
administrative card-application, specific to ISO/IEC 24727, providing card and application discoverability and administrative services
- 3.6**
card-application-path
ordered set of protocol termination points in the network that connects the client-application to the card-application
- 3.7**
card-application-service
set of actions
- 3.8**
channel
physical pathway allowing movement of bits of information between a client-application and a card-application
- 3.9**
client-application
software component running on a platform that uses data storage and computational services offered by a card-application
- 3.10**
connection
logically referenced channel
- 3.11**
global differential-identity
differential-identity recognized in all card-applications that are managed by the same alpha card-application
- 3.12**
local differential-identity
differential-identity recognized only within a specific card-application in which it is defined
- 3.13**
parameter
information required to define or effect an action
- 3.14**
return code
information, including status, returned as a consequence of an action
- 3.15**
security condition
boolean expression in terms of differential-identity authentication states
- 3.16**
session
connection used under a particular security context

3.17**target**

persistent entity that shall be manipulated through the actions of a card-application-service

4 Abbreviated terms

For the purposes of this document, the abbreviated terms given in ISO/IEC 24727-1 and the following apply.

ACL	access control list
AR	access rule
DID	differential-identity
DSI	data structure for interoperability

5 Organization for interoperability

5.1 General

A client-application and a card-application comprise two peer-level entities that interact through well-defined transaction and communication mechanisms. The ISO/IEC 24727-3 application interface shall be the sole mechanism through which a client-application interacts with a card-application.

This clause defines the model of computation and persistent entities on the ISO/IEC 24727-3 application interface by which the client-application and the card-application interact. A security model governs the security mechanisms through which trust in this interaction is achieved.

Clause 5.2 introduces the conceptual entities on the ISO/IEC 24727-3 application interface and describes their operational behavior and relationships. Clause 5.3 specifies the entities and relationships presented on the ISO/IEC 24727-3 application interface. Clause 5.4 specifies the security model provided on the ISO/IEC 24727-3 application interface.

5.2 Computation model

Using the terminology defined in ISO/IEC 24727-1:

5.2.1 A *client-application* may know a *card-application-path* to a *card-application*. Using this *card-application-path*, a client-application can initiate a *connection* to a card-application. The card-application shall be known as the *current card-application* for the connection.

5.2.2 The ISO/IEC 24727-3 *application interface* is the set of *card-application-services* provided to the client-application. Each card-application-service shall be comprised of *actions* that may be *requested* by a client-application and *confirmations* returned by the SAL.

5.2.3 If the client-application does not know a card-application-path to a card-application, it may make a request at the ISO/IEC 24727-3 application interface to discover a card-application-path to the requested card-application.

5.2.4 A card-application shall be uniquely identified by an AID.

5.2.5 The *alpha card-application* provides the basis for trusted management of card-applications by the client-application.

5.2.6 A client-application may open more than one connection. A client-application may open more than one connection with the same card-application, each connection being referenced by a different *connection handle*.

5.2.7 Using an open connection, a client-application may initiate a *session* with a card-application.

5.2.8 A client-application may open more than one session. Only one session may be open within a connection.

5.2.9 The *current state* is the current state of a connection defined by the current card-application, current data-set, authentication state of recognized differential-identities, and the presence of a session on the connection.

5.2.10 A card-application contains one or more card-application-services.

5.2.11 A specific card-application-service, the *Named Data Service*, provides access to zero or more *data-sets*.

5.2.12 A data-set contains zero or more *data structures for interoperability* (DSI). A data-set shall provide naming scope and access rules for the DSIs it contains.

5.2.13 Every data-set shall be accessible according to the *access rules* governing the actions available from the *Named Data Service*.

5.2.14 The currently selected data-set in a card-application shall be known as the *current data-set*.

5.2.15 A single *action request* at the ISO/IEC 24727-3 application interface may translate into multiple generic requests at the ISO/IEC 24727-2 generic card interface.

5.2.16 A card-application may support multiple actions that may be requested by the client-application through the ISO/IEC 24727-3 application interface.

5.2.17 An action request shall produce an *action confirmation*.

5.2.18 An *access rule* consists of the name of an action and a *security condition*. The security condition is said to be associated with the action by the access rule.

5.2.19 An *access control list* is a collection of zero or more access rules. An access control list shall be associated with each *target*.

5.2.20 An action involving a target may be successfully performed if and only if the security condition associated with the action by an access rule in the access control list of the target evaluates to TRUE.

5.2.21 An *authentication protocol* is the process by which a *differential-identity* demonstrates possession of a *marker*.

5.2.22 *Authentication* is the successful execution of an authentication protocol. In this case the differential-identity is said to be authenticated.

5.2.23 The *authentication state* of a differential-identity shall be a boolean variable that is TRUE if the differential-identity is authenticated and FALSE otherwise.

5.2.24 A client-application may learn about information, status or services provided by a card-application through a discovery process effected through the various List, Get, and Describe actions offered at the 24727-3 application interface.

5.2.25 Access rules are discoverable at the ISO/IEC 24727-3 application interface.

5.2.26 In general, the degree of interoperability of a card-application with client-applications depends on the discoverable information presented by the card-application at the ISO/IEC 24727-3 application interface.

5.3 Entity relationships on the application interface

5.3.1 General

This clause describes the entities accessible through a card-application, specifically the alpha card-application and card-applications it manages. Entities more directly involved in the security model are described in Clause 5.4.

Figure 1 illustrates the client-application to card-application paradigm that underlies the ISO/IEC 24727 standard. The ISO/IEC 24727-3 application interface facilitates the connection and session mechanisms shown in this figure.

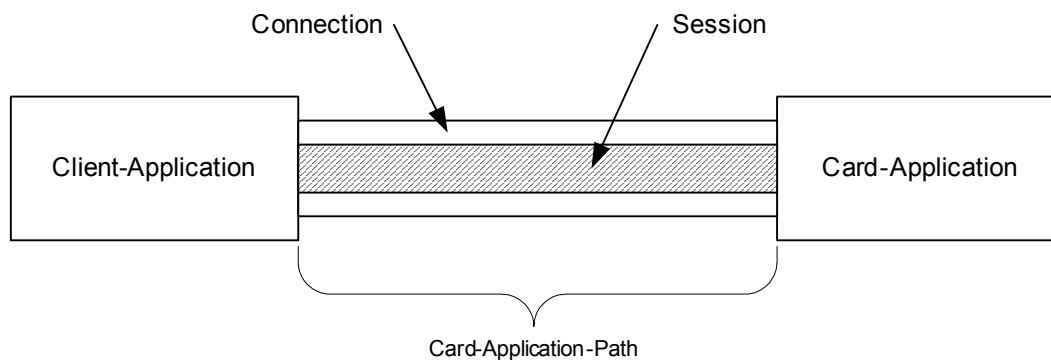


Figure 1 — Application Connection Paradigm

A card-application-path can be specified to allow a client-application to be connected to a specific card-application. Through such a connection, the client-application can access card-application-services using the ISO/IEC 24727-3 application interface. A session adds a security context to a connection in addition to its intrinsic security characteristics that can be used to protect the data flowing between the client-application and the card-application.

Figure 2 illustrates the entities defined in the model of computation and establishes the relationships among these entities. This entity-relationship diagram is elaborated below to describe individual card-application-services. The type of target applicable to each action is shown in the rounded boxes associated with each action.

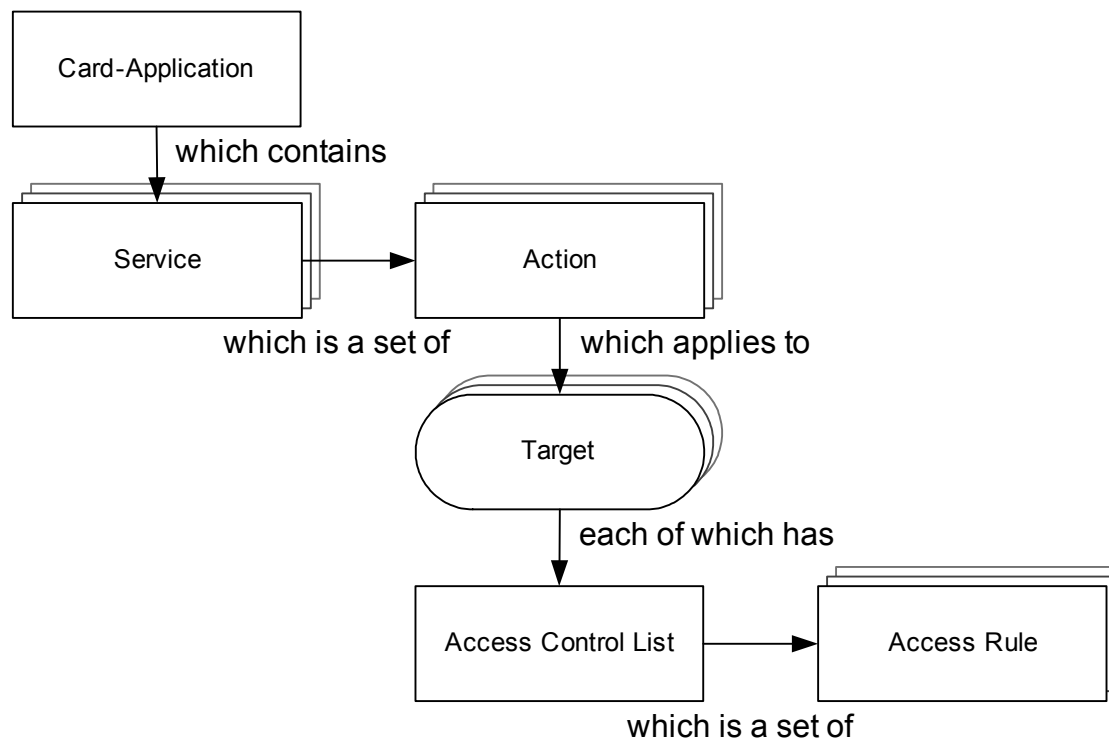


Figure 2 — Model of Computation Entities and Relationships

A card-application is a collection of targets and services. A service is a collection of actions. An action is an operation which is applied to a target. The actions and services of a card-application are accessed by a client-application using the ISO/IEC 24727-3 application interface.

5.3.2 Alpha Card-Application

The alpha-card-application shall be available at the application interface. It may either be present in the ICC or emulated by the SAL or GCAL implementation. In all cases, a connection by the client application to the alpha-card-application grants the availability of the card application list.

5.3.3 Accessing Card-Application-Services

A client-application uses the Initialize and CardApplicationPath entry points on the ISO/IEC 24727-3 application interface to gain access to card-application-services. A client-application uses the Terminate entry point to terminate access to card-application-services.

Before accessing card-application-services, a client-application must request the Initialize entry point on the ISO/IEC 24727-3 application interface. After the Initialize confirmation, the client-application may open connections and request actions with multiple card-applications, simultaneously or sequentially. The client-application should request the Terminate entry point when use of card-application-services is no longer needed.

5.3.4 Connection Service Interface

This card-application-service provides actions for the establishment of a connection between a client-application and a card-application. Once a connection is established, a session may be effected through this connection in order to enhance the security characteristics of the communication between the client-application and the card-application. Figure 3 illustrates the entity relationships of the Connection Service.

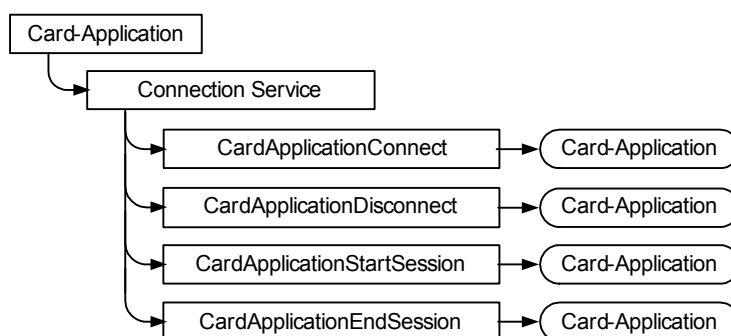


Figure 3 — Connection Service

The target of each of these actions, as specified in the diagram above, is the current card-application, or the card-application to which the client-application is attempting to connect.

5.3.5 Card-Application Service Interface

This card-application-service provides actions for the creation and manipulation of card-applications. Figure 4 illustrates the entity relationships of the Card-Application Service.

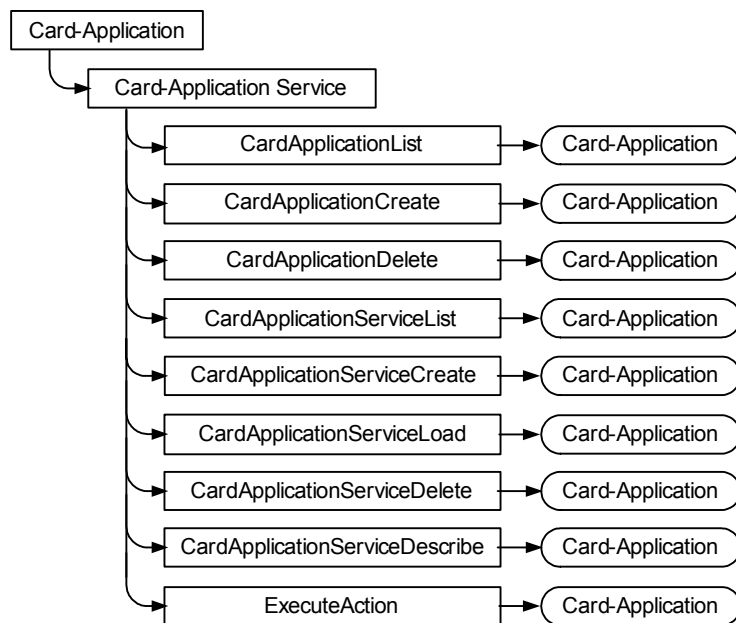


Figure 4 — Card-Application Service

The target of the CardApplicationCreate and CardApplicationDelete actions is the alpha card-application. The current card-application shall be the alpha card-application when requesting these actions.

The target of each of the remaining actions is the current card-application.

5.3.6 Named Data Service Interface

This card-application-service provides actions for the creation and manipulation of data-sets, a containment mechanism that allows for the establishment of common access rules for data contained in data structures for interoperability. Data-sets may contain any number of DSIs. Figure 5 illustrates the entity relationships of the Named Data Service.

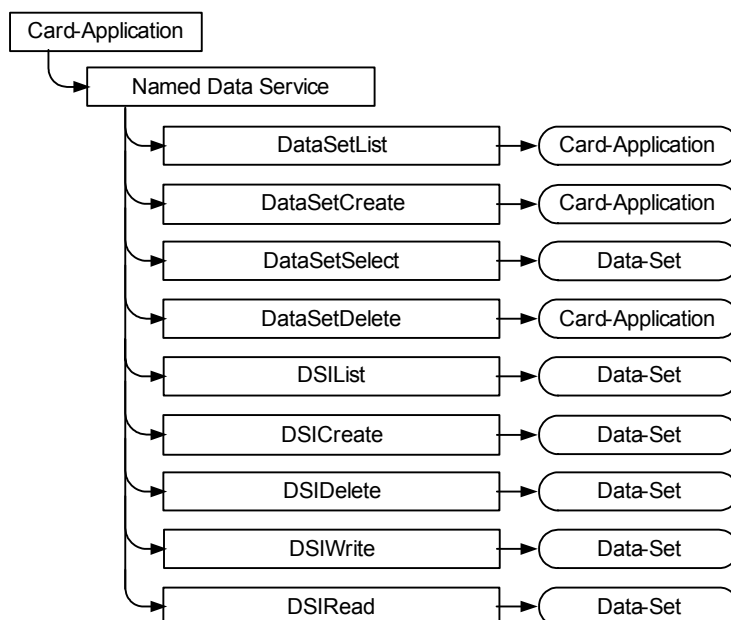


Figure 5 — Named Data Service

The target of each of these actions, as specified in the diagram above, is the current card-application, the current data-set, or the data-set which the client-application is attempting to select.

5.3.7 Cryptographic Service Interface

This card-application-service provides actions for a variety of cryptographic operations to be performed on or through the parameters contained within a differential-identity. Figure 6 illustrates the entity relationships of the Cryptographic Service.

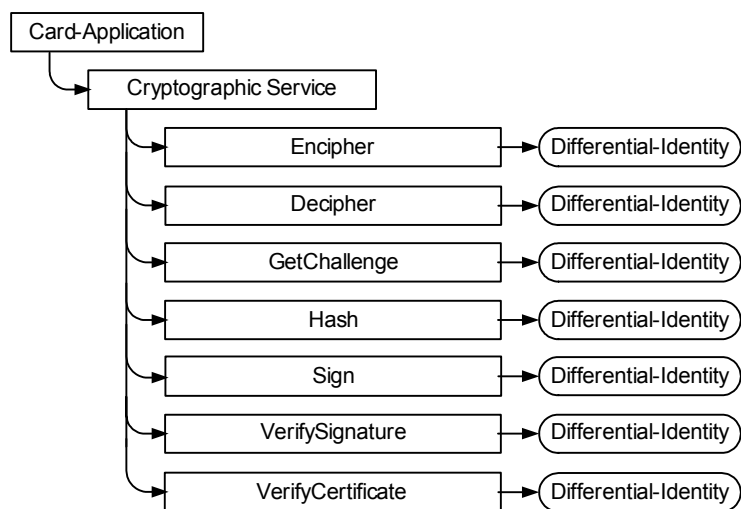


Figure 6 — Cryptographic Service

The target of each of these actions, as specified in the diagram above, is the differential-identity named in the request of the action.

5.3.8 Differential-Identity Service Interface

This card-application-service provides actions for the creation and manipulation of differential-identities. Figure 7 illustrates the entity relationships of the Differential-Identity Service.

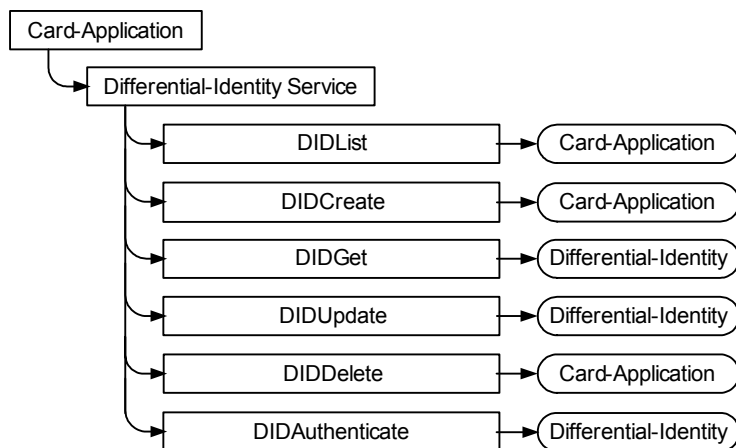


Figure 7 — Differential-Identity Service

The target of each of these actions, as specified in the diagram above, is either the current card-application or the differential-identity named in the request of the action.

5.3.9 Authorization Service Interface

This card-application-service provides actions for the manipulation of access control lists. Figure 8 illustrates the entity relationships of the Authorization Service.

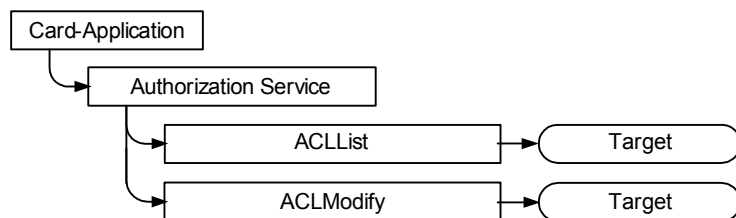


Figure 8 — Authorization Service

The target of these actions, as specified in the diagram above, can be any target. The target is specified in the request of the action.

5.4 Security model

5.4.1 General

A shared security state is established between a client-application and a card-application by successfully performing the required protocols between the client-application and the card-application. Establishing the required security state authorizes the performance of actions requested of a card-application by a client-application.

The mechanism for specifying the required security state for performing an action shall be the access rule. A security state is established by authenticating differential-identities.

5.4.2 Differential-Identity

In order to establish a security state encompassing a client-application and a card-application, the differential-identity authentication mechanism shall be used. The elements of each differential-identity are illustrated in Table 1. At the generic card interface, a differential-identity may be transmitted as defined by the ASN.1 in Annex C.

Table 1 — Elements of Differential-Identity

Element 1	Element 2	Element 3	Element 4	Element 5
DIDName	Authentication Protocol	Marker	Scope	Qualifier
Mandatory	Mandatory	Mandatory	Implicit	Optional

The DIDName element shall be specified by the client-application which creates the differential-identity at the time the differential-identity is created.

The Authentication Protocol element shall be specified as an object identifier (OID) and determines for which actions the differential-identity can be used. Only if the authentication protocol specified by this OID is defined for use in authentication as defined in Annex A shall its successful completion for the differential-identity set the authentication state of the differential-identity to TRUE within the card-application. A DIDName shall be associated with only one authentication protocol. The actions of the Cryptographic Service may make use of the information in the Marker element as defined in Annex A.

Note: The Authentication Protocol indicated by element 2 is used either for authentication or for other cryptographic services.

The Scope element is implicit. Differential-identities defined within the alpha card-application are global in scope and therefore are recognized within all card-applications managed by this alpha card-application. All other differential-identities are local in scope to the card-application in which they are defined. The scope of the corresponding authentication state shall be identical to the scope of the differential-identity.

The optional Qualifier element may provide details on the use of the differential-identity. For example, it may refer to an external specification by OID.

The Marker element is the information within the card-application, or a reference to it, that shall be used in executing the authentication protocol specified in the Authentication Protocol element.

Examples of markers include:

- a PIN
- a password

- a symmetric key
- an asymmetric key
- a digital certificate
- a biometric image or template
- a pair of symmetric keys; e.g., one for encryption and one for message authentication code (MAC) generation

As evidenced by the final entry in the previous list, the marker may be comprised of multiple parts; e.g., of multiple keys. This allows for complex authentication protocols that may need to use multiple keys in parallel or serially.

The Marker may often be implemented within a card-application as a key that is referred to by a key reference as specified in ISO/IEC 7816-4. One differential-identity and hence its marker may be related to multiple key references in some complex authentication protocols. The key reference may be coded as defined in ISO/IEC 7816-4 security environments. A specific key reference may appear in several differential-identities.

The association of the DIDName to a Marker and Authentication Protocol is the mechanism through which the differential-identity is associated with the marker. This mapping shall be maintained as part of the discovery information. This mapping allows the client-application and the card-application to recognize, exchange, and share information about a differential-identity.

5.4.3 Authentication Protocols

Authentication protocols shall be the mechanism by which the client-application sets an authentication state within the card-application for the differential-identity indicated by the DIDName. Authentication protocols are enumerated in Table 2, and described in Annex A.

Table 2 — Authentication Protocols

Simple Assertion	Modular Extended Access Control Protocol (M-EAC)
Asymmetric Internal Authenticate	Key Transport with mutual authentication based on RSA
Asymmetric External Authenticate	Age Attainment
Symmetric Internal Authenticate	Asymmetric Session Key Establishment
Symmetric External Authenticate	Secure PIN Compare
Compare	EC Key Agreement with Card-Application Authentication
PIN Compare	EC Key Agreement with Mutual Authentication
Biometric Compare	Simple EC-DH Key Agreement
Mutual Authentication with Key Establishment	GP Asymmetric Authentication
Client-Application Mutual Authentication with Key Establishment	GP Symmetric Authentication (Explicit Mode)
Client-Application Asymmetric External Authenticate	GP Symmetric Authentication (Implicit Mode)

To set the authentication state of a differential-identity to TRUE, the authentication protocol of that differential-identity shall be successfully completed using either the CardApplicationStartSession or DIDAuthenticate action.

Consecutive requests of the same action for the same connection handle and named differential-identity may be necessary to complete the authentication protocol as defined in Annex A. These represent individual steps in a multistep authentication protocol.

Differential-identity authentication states are valid for the connection handle specified during their authentication and shall be set to FALSE when the CardApplicationDisconnect action is requested for that connection handle or the connection handle otherwise becomes invalid. Authentication states established while connected to the alpha card-application shall be valid for all connections to card-applications managed by this alpha card-application. If a differential-identity was authenticated by requesting the CardApplicationStartSession action, a subsequent request of the CardApplicationEndSession action shall set its authentication state to FALSE.

The authentication state of a differential-identity shall be set to FALSE when the DIDAuthenticate or CardApplicationStartSession action is requested and set to TRUE only at the time of the action confirmation produced for the request that successfully completes the authentication protocol.

The authentication state of a differential-identity that is not recognized at the time its authentication state is evaluated shall evaluate to FALSE.

5.4.4 Session Keys

A protocol may involve the creation of cryptographic keys for use during a session.

The session keys shall become invalid when the CardApplicationDisconnect action is requested or the connection otherwise becomes invalid. Additionally, if the protocol was completed by requesting the CardApplicationStartSession action, the session keys shall become invalid when the CardApplicationEndSession action is requested.

5.4.5 Access Control Lists

An access control list (ACL) is a set of access rules (ARs) applicable to a target. An AR associates an action of a card-application-service with a security condition. A security condition is a boolean expression in terms of differential-identity authentication states. A target is a data-set, a differential-identity, or a card-application.

Authenticating a differential-identity shall set its authentication state to TRUE. Otherwise, its authentication state shall be FALSE. If the security condition evaluates to TRUE, the associated action is authorized. If the security condition evaluates to FALSE the associated action is not authorized.

An action involving a target shall only be performed if an AR exists in the ACL applicable to that target associating the action with a security condition that evaluates to TRUE with respect to the current authentication states of the differential-identities recognized within the current card-application. An AR is only applicable to requests of its action and only when the request involves the target to which its ACL is associated.

An ACL is a part of the target to which it applies and is not a distinct target itself. Therefore when an action of the Authorization Service appears in an AR, the AR controls whether that action may be performed on the ACL in which it exists. The ARs controlling access to an ACL exist within the ACL itself.

Through the inclusion of a differential-identity in a security condition, the authentication of that differential-identity may become necessary to authorize the action associated with it in an access rule. If the differential-identity is associated with an authentication protocol that generates session keys, its inclusion in the security condition may make necessary the establishment of a security context.

6 Card-application-service access

6.1 General

This clause defines entry points on the ISO/IEC 24727-3 application interface through which a client-application can locate a specific card-application and establish communication channels to that card-application. A client-application may connect to more than one card-application.

6.2 Initialize

6.2.1 Purpose

This entry point shall initialize the ISO/IEC 24727-3 layer including connectivity to other components of ISO/IEC 24727 protocol stack. Card-application-services shall be available on the ISO/IEC 24727-3 application interface to the client-application only after a successful call to this entry point.

The initialization state (variables, possible values, resources, supporting card-application-services) of the ISO/IEC 24727-3 application interface implementation is out of scope for ISO/IEC 24727-3.

6.2.2 Entry point

```
OUT      ReturnCode      Initialize(
                                );
```

6.2.3 Parameters

None

6.2.4 Prerequisites

None

6.2.5 Return codes

API_OK

API_COMMUNICATION_FAILURE
API_INCORRECT_PARAMETER

6.2.6 Impact on current state

Upon successful completion, the ISO/IEC 24727-3 layer shall be initialized.

6.3 Terminate

6.3.1 Purpose

This entry point shall terminate access to card-application-services offered on the ISO/IEC 24727-3 application interface. Any connection(s) currently in effect between the client-application and any card-application(s) shall be disconnected. The Initialize entry point shall be called before any card-application-services are available on the ISO/IEC 24727-3 application interface to the client-application.

6.3.2 Entry point

```
OUT      ReturnCode  Terminate(
                                );
```

6.3.3 Parameters

None

6.3.4 Prerequisites

None

6.3.5 Return codes

```
API_OK
API_WARNING_CONNECTION_DISCONNECTED
API_INCORRECT_PARAMETER
```

```
API_NOT_INITIALIZED
API_COMMUNICATION_FAILURE
```

6.3.6 Impact on current state

The return code of API_WARNING_CONNECTION_DISCONNECTED is a successful completion of the request with the side effect of disconnecting at least one active connection.

The return code of API_COMMUNICATION_FAILURE is an unsuccessful completion of the action. In this case, the state of existing connections is undefined.

6.4 CardApplicationPath

6.4.1 Purpose

This entry point shall determine card-application-paths from the client-application to a named card-application.

6.4.2 Entry point

OUT	ReturnCode	CardApplicationPath (
IN	CardApplicationPath	cardAppPathRequest ,
OUT	CardApplicationPathSet	cardAppPathResultSet
		);

6.4.3 Parameters

cardAppPathRequest sequence of protocol termination points representing the final segment of a card-application-path from the client-application to the card-application, including its AID, a host or terminal identifier and optionally an IFD name or identifier. The specific syntax of this final segment is expressed by IETF RFC 2141.

cardAppPathResultSet set of card-application-paths between the client-application and the named card-application, each of which contains the value of **cardAppPathRequest** as its final segment.

6.4.4 Prerequisites

None

6.4.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_TOO_MANY_RESULTS

API_COMMUNICATION_FAILURE

6.4.6 Impact on current state

None

7 Connection service

7.1 General

This clause defines actions through which a client-application can connect to or disconnect from a card-application and start or end a session based on a connection.

7.2 CardApplicationConnect

7.2.1 Purpose

This action shall establish an unauthenticated connection between the client-application and a card-application.

7.2.2 Action

OUT	ReturnCode	CardApplicationConnect (
IN	CardApplicationPath	cardApplicationPath,
IN	BOOLEAN	exclusiveUse,
OUT	ConnectionHandle	connectionHandle
		);

7.2.3 Parameters

cardApplicationPath card-application-path from the client-application to the card-application

exclusiveUse if TRUE, this indicates to the ISO/IEC 24727-3 layer that the connection should be opened if no other connection currently exists with the the card-application and no other connection should be established to the card-application until this connection is closed; if FALSE, multiple simultaneous connections may be opened to this card-application

connectionHandle opaque reference for use at the ISO/IEC 24727-3 application interface in action requests

7.2.4 Prerequisites

None

7.2.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED

API_EXCLUSIVE_NOT_AVAILABLE

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

7.2.6 Impact on current state

Upon successful completion, the card-application named in the card-application-path is the current card-application and the card-application associated with the connection handle.

7.3 CardApplicationDisconnect

7.3.1 Purpose

This action shall terminate a connection between the client-application and a card-application.

7.3.2 Action

OUT	ReturnCode	CardApplicationDisconnect (
IN	ConnectionHandle	connectionHandle,
IN	ReaderAction	action
		);

7.3.3 Parameters

connectionHandle connection handle

action optional argument indicating a requested IFD action to reset or unpower the IFD or eject or confiscate the ICC

7.3.4 Prerequisites

A valid `connectionHandle` is expected.

7.3.5 Return codes

API_OK
 API_WARNING_SESSION_ENDED

 API_INCORRECT_PARAMETER

 API_NOT_INITIALIZED
 API_SECURITY_CONDITION_NOT_SATISFIED
 API_COMMUNICATION_FAILURE

7.3.6 Impact on current state

The return code of `API_WARNING_SESSION_ENDED` is a successful completion of the action with the side effect of ending at least one active session on the connection.

7.4 CardApplicationStartSession

7.4.1 Purpose

This action shall establish a session between a client-application and a card-application.

A connection handle to a second card-application, e.g. a security access module (SAM), may be provided to perform cryptographic operations on behalf of the client-application in establishing the session.

Consecutive requests of this action for the same connection handle and named differential-identity may be necessary to complete the protocol as defined in Annex A.

7.4.2 Action

OUT	ReturnCode	CardApplicationStartSession(
IN	ConnectionHandle	connectionHandle,
IN	DIDScope	didScope,
IN	DIDName	didName,
IN/OUT	DIDAuthenticationData	authenticationProtocolData,
IN	ConnectionHandle	samConnectionHandle
		);

7.4.3 Parameters

connectionHandle	connection handle
didScope	scope of the named differential-identity
didName	name of the differential-identity containing the authentication protocol to be executed to establish the session between the client-application and the card-application
authenticationProtocolData	data exchanged according to the protocol
samConnectionHandle	optional argument to a card-application that can be used by the ISO/IEC 24727-3 layer

7.4.4 Prerequisites

A valid `connectionHandle` is expected.

7.4.5 Return codes

API_OK
 API_NEXT_REQUEST

 API_INCORRECT_PARAMETER
 API_NAMED_ENTITY_NOT_FOUND
 API_PROTOCOL_NOT_RECOGNIZED
 API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

 API_NOT_INITIALIZED
 API_SECURITY_CONDITION_NOT_SATISFIED
 API_INSUFFICIENT_RESOURCES
 API_COMMUNICATION_FAILURE

7.4.6 Impact on current state

See Annex A.

7.5 CardApplicationEndSession

7.5.1 Purpose

This action shall end a session between a client-application and a card-application.

7.5.2 Action

```
OUT      ReturnCode      CardApplicationEndSession(
IN      ConnectionHandle  connectionHandle
      );
```

7.5.3 Parameters

connectionHandle connection handle

7.5.4 Prerequisites

A valid `connectionHandle` is expected.

7.5.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NO_ACTIVE_SESSION

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

7.5.6 Impact on current state

Upon successful completion, the security characteristics of the connection revert to the intrinsic security characteristics of the connection.

8 Card-application service

8.1 General

This clause defines actions through which card-applications can be created and deleted and through which services can be created and deleted within card-applications. The `CardApplicationCreate()` action provides administrative functionality to establish the existence of the card-application. Card-application executable code is loaded via the `CardApplicationServiceLoad()` action.

8.2 CardApplicationList

8.2.1 Purpose

This action shall return the names of the card-applications listed in the alpha card-application. The access rule applicable to this action shall be in the access control list of the current card-application.

8.2.2 Action

OUT	ReturnCode	CardApplicationList (
IN	ConnectionHandle	connectionHandle,
OUT	CardApplicationNameList	cardApplicationNameList
		);

8.2.3 Parameters

connectionHandle connection handle

cardApplicationNameList list of card-application names

8.2.4 Prerequisites

A valid `connectionHandle` is expected.

8.2.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

8.2.6 Impact on current state

None

8.3 CardApplicationCreate

8.3.1 Purpose

This action shall create a new card-application.

8.3.2 Action

OUT	ReturnCode	CardApplicationCreate (
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationName	cardApplicationName,
IN	AccessControlList	cardApplicationACL
		);

8.3.3 Parameters

connectionHandle connection handle

cardApplicationName name of the card-application to be created

cardApplicationACL access control list for this new card-application

8.3.4 Prerequisites

A valid `connectionHandle` is expected. The current card-application shall be the alpha card-application.

8.3.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAME_EXISTS

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_PREREQUISITE_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

8.3.6 Impact on current state

None

8.4 CardApplicationDelete

8.4.1 Purpose

This action shall delete the named card-application including all of its services, data-sets, and differential-identities.

8.4.2 Action

OUT	ReturnCode	CardApplicationDelete(
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationName	cardApplicationName
		);

8.4.3 Parameters

connectionHandle connection handle

cardApplicationName name of the card-application to be deleted

8.4.4 Prerequisites

A valid `connectionHandle` is expected. The current card-application shall be the alpha card-application.

8.4.5 Return codes

API_OK
 API_WARNING_CONNECTION_DISCONNECTED

 API_INCORRECT_PARAMETER
 API_NAMED_ENTITY_NOT_FOUND

 API_NOT_INITIALIZED
 API_SECURITY_CONDITION_NOT_SATISFIED
 API_PREREQUISITE_NOT_SATISFIED
 API_COMMUNICATION_FAILURE

8.4.6 Impact on current state

Upon successful completion, any connections to the named card-application shall be disconnected.

8.5 CardApplicationServiceList

8.5.1 Purpose

This action shall list the card-application-services in the current card-application.

8.5.2 Action

```
OUT      ReturnCode      CardApplicationServiceList (
IN      ConnectionHandle  connectionHandle,
OUT      CardApplicationServiceNameList cardApplicationServiceNameList
);
```

8.5.3 Parameters

connectionHandle connection handle

cardApplicationServiceNameList list of names of card-application-services in the current card-application

8.5.4 Prerequisites

A valid `connectionHandle` is expected.

8.5.5 Return codes

- API_OK
- API_INCORRECT_PARAMETER
- API_NOT_INITIALIZED
- API_SECURITY_CONDITION_NOT_SATISFIED
- API_COMMUNICATION_FAILURE

8.5.6 Impact on current state

None

8.6 CardApplicationServiceCreate

8.6.1 Purpose

This action shall create a new card-application-service in the current card-application.

8.6.2 Action

OUT	ReturnCode	CardApplicationServiceCreate(
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationServiceName	cardApplicationServiceName
		);

8.6.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service to be created

8.6.4 Prerequisites

A valid `connectionHandle` is expected.

8.6.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAME_EXISTS

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

8.6.6 Impact on current state

None

8.7 CardApplicationServiceLoad

8.7.1 Purpose

This action shall load executable code that implements a card-application-service into the current card-application.

8.7.2 Action

OUT	ReturnCode	CardApplicationServiceLoad(
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationServiceName	cardApplicationServiceName,
IN	CardApplicationServiceLoadPackage	code
		);

8.7.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service implemented by the code

code service load package

8.7.4 Prerequisites

A valid `connectionHandle` is expected.

8.7.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

8.7.6 Impact on current state

None

8.8 CardApplicationServiceDelete

8.8.1 Purpose

This action shall delete the named card-application-service, including the code that implements it, from the current card-application.

8.8.2 Action

OUT	ReturnCode	CardApplicationServiceDelete(
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationServiceName	cardApplicationServiceName
		);

8.8.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service to be deleted

8.8.4 Prerequisites

A valid `connectionHandle` is expected.

8.8.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

8.8.6 Impact on current state

None

8.9 CardApplicationServiceDescribe

8.9.1 Purpose

This action shall return a URL or full description of the named card-application-service. The resulting description allows the client-application to discover functionality beyond the standardized set of card-application-services described in ISO/IEC 24727-3.

8.9.2 Action

OUT	ReturnCode	CardApplicationServiceDescribe (
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationServiceName	cardApplicationServiceName,
OUT	CardApplicationServiceDescription	serviceDescription
		);

8.9.3 Parameters

connectionHandle	connection handle
cardApplicationServiceName	name of the card-application-service to be described
serviceDescription	URI, URL, or full description (syntax) of the named card-application-service (may be used as a basis for correct syntax for SAL requests on this named card-application-service via the ExecuteAction action)

8.9.4 Prerequisites

A valid `connectionHandle` is expected.

8.9.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

8.9.6 Impact on current state

None

8.10 ExecuteAction

8.10.1 Purpose

This action shall request access to an action in a card-application-service that is not defined in ISO/IEC 24727-3.

The functionality of the actions accessed through this action is out of scope.

8.10.2 Action

OUT	ReturnCode	ExecuteAction(
IN	ConnectionHandle	connectionHandle,
IN	CardApplicationServiceName	cardApplicationServiceName,
IN	ActionName	actionName,
IN	ExecuteActionRequest	request,
OUT	ExecuteActionConfirmation	confirmation
		);

8.10.3 Parameters

connectionHandle	connection handle
cardApplicationServiceName	name of the card-application-service containing the action to be requested
actionName	name of the action to be requested
request	action request data
confirmation	action confirmation data

8.10.4 Prerequisites

A valid `connectionHandle` is expected.

8.10.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

8.10.6 Impact on current state

None

9 Named data service

9.1 General

This clause defines actions through which data can be managed within a card-application.

9.2 DataSetList

9.2.1 Purpose

This action shall list the names of data-sets defined in the current card-application.

9.2.2 Action

OUT	ReturnCode	DataSetList (
IN	ConnectionHandle	connectionHandle,
OUT	DataSetNameList	dataSetNameList
		);

9.2.3 Parameters

connectionHandle connection handle

dataSetNameList list of names of the data-sets defined within the current card-application

9.2.4 Prerequisites

A valid `connectionHandle` is expected.

9.2.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

9.2.6 Impact on current state

None

9.3 DataSetCreate

9.3.1 Purpose

This action shall create a new data-set in the current card-application.

9.3.2 Action

OUT	ReturnCode	DataSetCreate(
IN	ConnectionHandle	connectionHandle,
IN	DataSetName	dataSetName,
IN	AccessControlList	dataSetACL
		);

9.3.3 Parameters

connectionHandle	connection handle
dataSetName	name of the data-set
dataSetACL	access control list for the data-set

9.3.4 Prerequisites

A valid `connectionHandle` is expected.

9.3.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAME_EXISTS

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

9.3.6 Impact on current state

Upon successful completion, the new data-set becomes the current data-set in the current card-application.

9.4 DataSetSelect

9.4.1 Purpose

This action shall select the named data-set in the current card-application.

9.4.2 Action

OUT	ReturnCode	DataSetSelect (
IN	ConnectionHandle	connectionHandle,
IN	DataSetName	dataSetName
		);

9.4.3 Parameters

connectionHandle connection handle

dataSetName name of the data-set

9.4.4 Prerequisites

A valid `connectionHandle` is expected.

9.4.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

9.4.6 Impact on current state

Upon successful completion, the selected data-set becomes the current data-set in the current card-application.

9.5 DataSetDelete

9.5.1 Purpose

This action shall delete the named data-set within the current card-application including all of the DSIs within that data-set.

9.5.2 Action

OUT	ReturnCode	DataSetDelete(
IN	ConnectionHandle	connectionHandle,
IN	DataSetName	dataSetName
		);

9.5.3 Parameters

connectionHandle connection handle

dataSetName name of the data-set

9.5.4 Prerequisites

A valid `connectionHandle` is expected.

9.5.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_COMMUNICATION_FAILURE

9.5.6 Impact on current state

If the current data-set at the time this action is requested is the data-set to be deleted, the current data-set is undefined following the action confirmation.

9.6 DSIList

9.6.1 Purpose

This action shall list the names of DSIs in the current data-set.

9.6.2 Action

OUT	ReturnCode	DSIList (
IN	ConnectionHandle	connectionHandle,
OUT	DSINameList	dsiNameList
		);

9.6.3 Parameters

connectionHandle connection handle

dsiNameList list of names of the DSIs in the current data-set

9.6.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.6.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_PREREQUISITE_NOT_SATISFIED

API_COMMUNICATION_FAILURE

9.6.6 Impact on current state

None

9.7 DSICreate

9.7.1 Purpose

This action shall create a new DSI in the current data-set.

9.7.2 Action

OUT	ReturnCode	DSICreate(
IN	ConnectionHandle	connectionHandle,
IN	DSIName	dsiName,
IN	DSIContent	dsiContent
		);

9.7.3 Parameters

connectionHandle	connection handle
dsiName	name of the DSI to be created in the current data-set
dsiContent	content to be stored in the DSI

9.7.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.7.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAME_EXISTS

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_PREREQUISITE_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

9.7.6 Impact on current state

None

9.8 DSIDelete

9.8.1 Purpose

This action shall delete the named DSI from the current data-set.

9.8.2 Action

```

OUT      ReturnCode      DSIDelete(
IN      ConnectionHandle  connectionHandle,
IN      DSIName           dsiName
                           );

```

9.8.3 Parameters

connectionHandle connection handle

dsiName name of the DSI to be deleted in the current data-set

9.8.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.8.5 Return codes

API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_COMMUNICATION_FAILURE

9.8.6 Impact on current state

None

9.9 DSIWrite

9.9.1 Purpose

This action shall replace the contents of the named DSI in the current data-set with the data provided.

9.9.2 Action

OUT	ReturnCode	DSIWrite(
IN	ConnectionHandle	connectionHandle,
IN	DSIName	dsiName,
IN	DSIContent	dsiContent
		);

9.9.3 Parameters

connectionHandle	connection handle
dsiName	name of the DSI to be written in the current data-set
dsiContent	content to be stored in the DSI

9.9.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.9.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_PREREQUISITE_NOT_SATISFIED

API_INSUFFICIENT_RESOURCES

API_COMMUNICATION_FAILURE

9.9.6 Impact on current state

None

9.10 DSIRead

9.10.1 Purpose

This action shall return the contents of the named DSI in the current data-set.

9.10.2 Action

OUT	ReturnCode	DSIRead(
IN	ConnectionHandle	connectionHandle,
IN	DSIName	dsiName,
OUT	DSIContent	dsiContent
		);

9.10.3 Parameters

connectionHandle connection handle

dsiName name of the DSI to be read in the current data-set

dsiContent content of the DSI

9.10.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.10.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED

API_SECURITY_CONDITION_NOT_SATISFIED

API_PREREQUISITE_NOT_SATISFIED

API_COMMUNICATION_FAILURE

9.10.6 Impact on current state

None

10 Cryptographic service

10.1 General

This clause defines actions through which cryptographic operations can be performed.

The algorithms used in the performance of these actions depend on the authentication protocols as defined in Annex A.