GLOBALPLATFORM

THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

GlobalPlatform Device Technology

Secure Element Access Control

Version 1.0

Public Release

May 2012

Document Reference: GPD_SPE_013

This page intentionally left blank.

Contents

Figures

Tables

1 Introduction

GlobalPlatform has defined a standard that enables several parties to independently and securely manage their stakes in a single Secure Element. This security model has allowed applications such as banking and transport to be deployed in a variety of situations. As these services reach the context of personal devices such as mobile phones, service owners start to leverage the device's capabilities to enrich their customers' experiences.

These applications will rely both on the device itself and on Secure Elements. An API (such as [OpenMobileAPI]), referred to in this document as the Secure Element access API, is used by the device applications to exchange data with their counterpart applications running in the Secure Element.

Restricting the use of such an API is necessary since modern mobile operating systems do not efficiently prevent unauthorized parties from abusing the API and potentially causing damage to the Secure Element itself.

This security mechanism, called Secure Element access control, defined in this specification, is used in addition to existing protection mechanisms (such as permissions or security OS policy limiting access to sensitive APIs). The access control is designed to prevent unauthorized access to resources in the Secure Elements and typically to prevent denial of services attacks (PIN blocking, selection of non multi-selectable applets, etc.).

This access control mechanism is transparent to client applications running in the device and is enforced within the device operating system itself.

This document specifies how the access policy is stored in the Secure Element, and how it can be accessed and applied by the device.

1.1 Audience

This specification is intended primarily for Secure Elements manufacturers, handset manufacturers, and Secure Element issuers.

1.2 IPR Disclaimer

GlobalPlatform draws attention to the fact that claims that compliance with this specification may involve the use of a patent or other intellectual property right (collectively, "IPR") concerning this specification may be published at https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform takes no position concerning the evidence, validity, and scope of these IPR claims.

1.3 References

This section lists both normative and informative references.

1.3.1 Normative References

Table 1-1: Normative References

Standard / Specification	Description	Ref
GlobalPlatform Card Specification	GlobalPlatform Card Specification v 2.2.1, January 2011	[GPCardSpec]
GlobalPlatform Confidential Card Content Management	GlobalPlatform Card, Confidential Card Content Management, Card Specification v2.2 – Amendment A, v1.0.1, January 2011	[GPAmdA]
GlobalPlatform Contactless Services	GlobalPlatform Card, Contactless Services, Card Specification v2.2 – Amendment C, v1.0.1, February 2011	[GPAmdC]
ETSI TS 102 221	Smart cards; UICC – Terminal interface; Physical and logical characteristics, Release 6, 2004	[102 221]
ETSI TS 102 622	Smart Cards; UICC – Contactless Front end (CLF) Interface; Host Controller Interface (HCI), Release 7, 2009	[102 622]
ISO/IEC 7816-4	Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange	[7816-4]
ISO/IEC 7816-5	Identification cards – Integrated circuit cards – Part 5: Registration of application providers	[7816-5]
ISO/IEC 7816-6	Identification cards – Integrated circuit cards with contacts – Part 6: Interindustry data elements for interchange	[7816-6]
ISO/IEC 7816-15	Identification cards – Integrated circuit cards with contacts – Part 15: Cryptographic information application	[7816-15]
ISO/IEC 8825-1| ITU-T Recommendation X.690	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), 2002	[X.690]
PKCS#15	PKCS #15 v1.1: Cryptographic Token Information Syntax Standard; RSA Laboratories; June 6, 2000	[PKCS15]

1.3.2 Informative References

Table 1-2: Informative References

Standard / Specification	Description	Ref
ETSI TS 102 225	Smart cards; Secured packet structure for UICC based applications, Release 6, 2004	[102 225]
ETSI TS 102 226	Smart cards; Remote APDU structure for UICC based applications, Release 6, 2004	[102 226]
ETSI TS 102 241	Smart cards; UICC Application Programming Interface (UICC API) for Java Card™, Release 6, 2004	[102 241]
GSMA APIs	GSMA NFC Handset APIs & Requirements Version 2.0	[GSMA]
ISO/IEC 14443-3	Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 3: Initialization and anticollision	[14443-3]
ISO/IEC 14443-4	Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol	[14443-4]
PKCS#1	PKCS #1 v2.0: RSA Cryptography Standard, RSA Laboratories, October 1998 (RFC 2437).	[PKCS1]
Java Card	Go to the following website for Java Card™ documentation:[1] http://www.oracle.com/technetwork/java/javacard/overview/index.html	[JavaCard]
Open Mobile API	SIMalliance Open Mobile API available under: http://www.simalliance.org	[OpenMobileAPI]

[1] Java Card is a trademark of Oracle and/or its affiliates.

1.4 Terminology and Definitions

Table 1-3: Terminology and Definitions

Term	Definition
Access Control Enforcer	Software that is part of the Secure Element access API, it obtains access rules from the Secure Element and applies those rules to restrict device application access to the various Secure Element applications.
Device application	A third party application running on the open mobile OS.
Mobile device	Any device that includes a secure element, such as mobile phone.
Open mobile OS	An operating system for mobile devices that allows the loading of third party applications.
Secure Element	A device used to securely store application-critical data (such as secret keys). A Secure Element will host a number of Secure Element applications.
Secure Element application	A software application installed and running on the Secure Element.
Universal Integrated Circuit Card (UICC)	A Secure Element used in the mobile communications industry, as defined in ETSI TS 102 221 [102 221].

1.5 Abbreviations and Notations

Table 1-4: Abbreviations

Abbreviation	Meaning
ACCF	Access Control Conditions File
ACMF	Access Control Main File
ACRF	Access Control Rules File
AID	Application IDentifier, following ISO/IEC 7816-5 [7816-5]
APDU	Application Protocol Data Unit
API	Application Programming Interface
AR	Access Rule
AR-DO	Access Rule Data Object
ARA	Access Rule Application
ARA-C	Access Rule Application Client
ARA-M	Access Rule Application Master
ARF	Access Rule Files
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
DO	Data Object (BER encoded TLV)
DODF	Data Object Directory File

Abbreviation	Meaning
EFdir	Application Directory Elementary File
ETSI	European Telecommunications Standards Institute
EVT	Event
GSM	Global System for Mobile Communication, originally Groupe Spécial Mobile
GSMA	GSM Association
HCI	Host Controller Interface
ISD	Issuer Security Domain
ISO	International Organization for Standardization
MF	Master File
NFC	Near Field Communication
OidDO	Object identifier for Data Object
OTA	Over The Air
PKCS	Public Key Cryptographic Standard
RAM	Remote Application Management
RFM	Remote File Management
SD	Security Domain
SE	Secure Element
SHA-1	Secure Hash Algorithm One
SWP	Single Wire Protocol
TLV	Tag Length Value
TSM	Trusted Service Manager
UICC	Universal Integrated Circuit Card

1.6 Revision History

Table 1-5: Revision History

Date	Version	Description
May 2012	1.0	Initial publication.

2 Architecture

This specification defines a generic mechanism for Secure Element access control, usable for any kind of Secure Element (e.g. embedded SE, microSD card with security controller, UICC, etc.). It supports application management by multiple entities and allows each entity to set the access rules for its card applications.

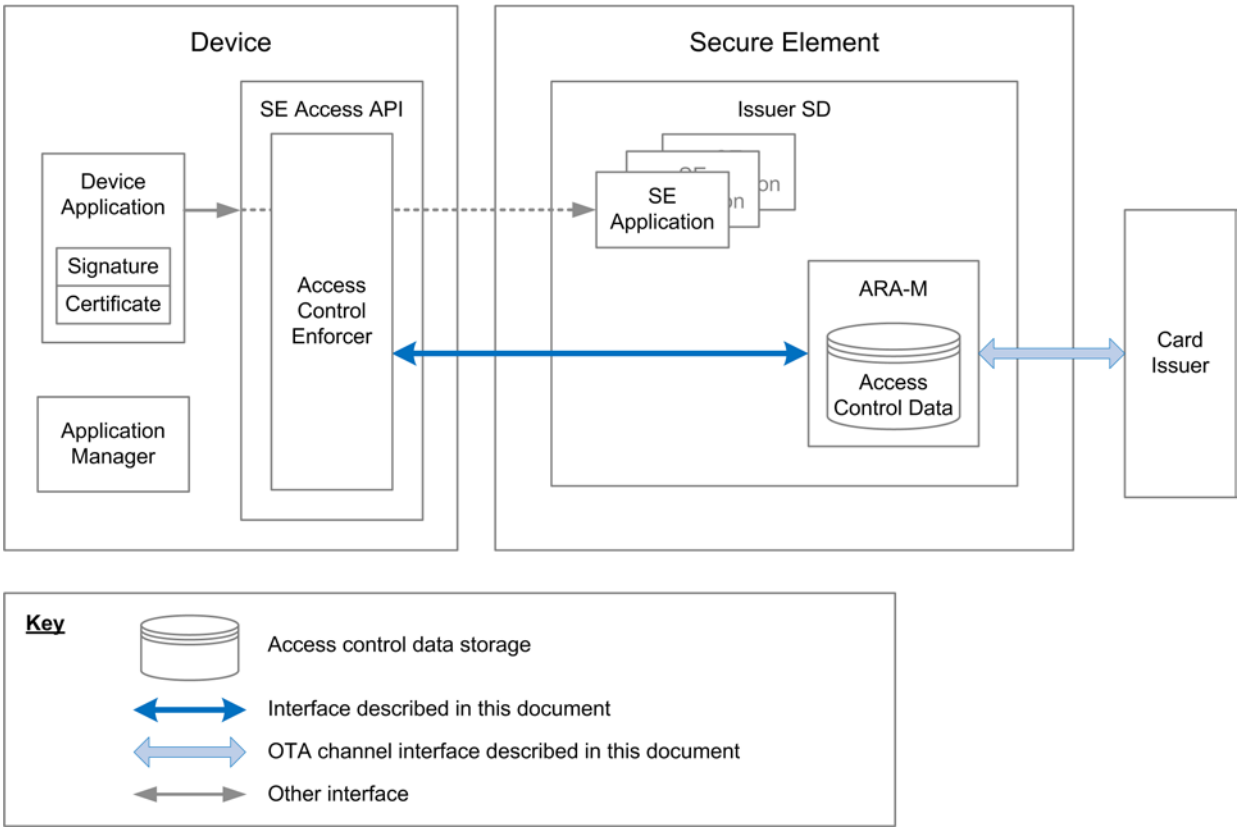
Secure Element access rule data is stored in the Secure Element (SE) and used by an Access Control enforcer on the device. The Access Control enforcer shall retrieve the access rules from the Secure Element and apply those rules to restrict device application access to the various Secure Element applications.

This chapter illustrates several variations on the system architecture:

2.1 Rules in Issuer Security Domain Only

In the most basic implementation of this specification, all Access Control rules are defined by the Card Issuer and stored in the Issuer Security Domain (ISD) as illustrated in Figure 2-1.

Figure 2-1: Access Control Architecture – Rules in ISD Only



The Card Issuer defines access control rules for the SE applications, and supplies those rules to the Access Rule Application Master (ARA-M). (The Card Issuer may delegate administration to a TSM.)

When a device application attempts to access an SE application, the Access Control enforcer shall use the device interface provided by the ARA-M to retrieve access rules from the SE (or shall consult the full set of rules that it obtained in advance), and shall permit the access only if the rules indicate that it is acceptable.

The ARA-M is an ordinary SE application which can be selected by a GlobalPlatform-defined AID, as follows:

Executable Load File AID:	'A00000015141434C'
Executable Module AID:	'A00000015141434C00'
Application AID:	'A00000015141434C00'

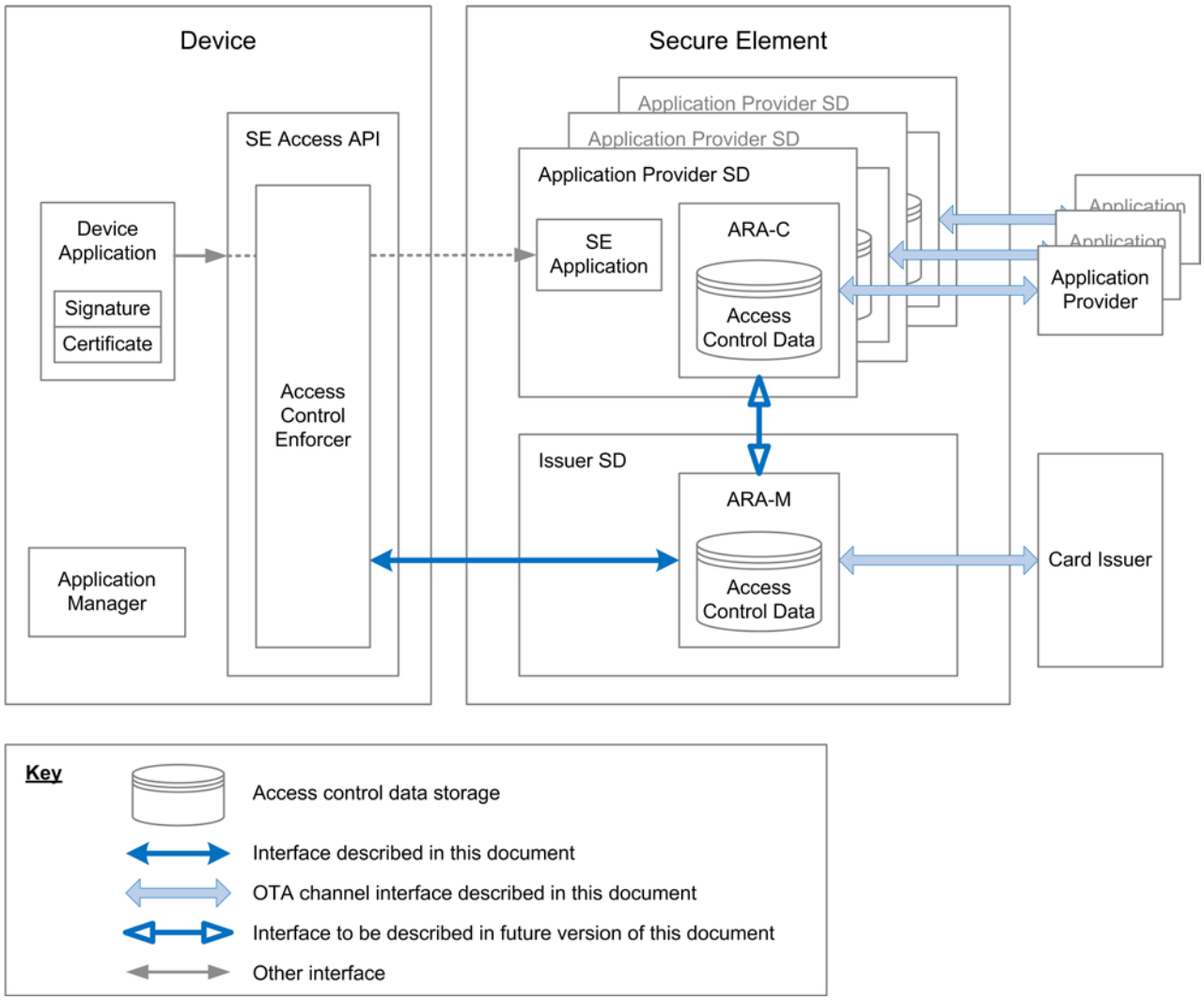
The ARA-M application is unique. Although access rule data can be stored in different locations within the SE (as discussed in the following sections), the ARA-M is in charge of retrieving all available access rules after a request from the Access Control enforcer on the device.

The interface between the Access Control enforcer and the ARA-M is described in Chapter 3. The interface between the Card Issuer (or TSM) and the ARA-M is described in section 4.3.

2.2 Rules in Issuer and Application Provider Security Domains

Application Providers may wish to define access control rules for the applications in their Security Domains and manage these rules by themselves. To support rules defined by both Card Issuers and Application Providers, this specification is implemented as illustrated in Figure 2-2.

Figure 2-2: Access Control Architecture – Rules in ISD and APSDs



Each Application Provider may define access rules for the applications in its Security Domain (SD), and supply those rules to an Access Rule Application Client (ARA-C). (An Application Developer may delegate administration to a TSM.)

When a device application attempts to access an SE application, the Access Control enforcer shall request the pertinent rules from the ARA-M. The ARA-M shall provide the appropriate rules, whether they are stored on the ARA-M or on an ARA-C. (As mentioned in section 2.1, the Access Control enforcer may obtain the full set of rules in advance.) The Access Control enforcer shall permit the access only if the rules indicate that it is acceptable.

The interface between the Access Control enforcer and the ARA-M is described in Chapter 3. The interface between the Application Provider (or TSM) and the ARA-C is described in section 4.3. The interface between the ARA-M and the ARA-C is out of scope of this specification.

2.2.1 Limitations on Rule Retrieval

If an ARA-C is deleted, it is expected that all the rules stored to that ARA-C will be deleted.

If an ARA-C is locked as defined by GlobalPlatform Card Specification [GPCardSpec], then the ARA-M shall ignore all rules stored to that ARA-C.

2.2.2 ARA-M and ARA-C Architecture and Security Considerations

An ARA-C shall register to the ARA-M so that all the rules stored to that ARA-C shall be taken into account by the ARA-M. This registration process can be performed by the ARA-C itself or can be performed by sending a STORE DATA (Command-Register-ClientAIDs-DO) command to the ARA-M.

This STORE DATA command could be used if the ARA-M has been replaced in the field, and therefore needs to be informed about already existing ARA-Cs on the SE.

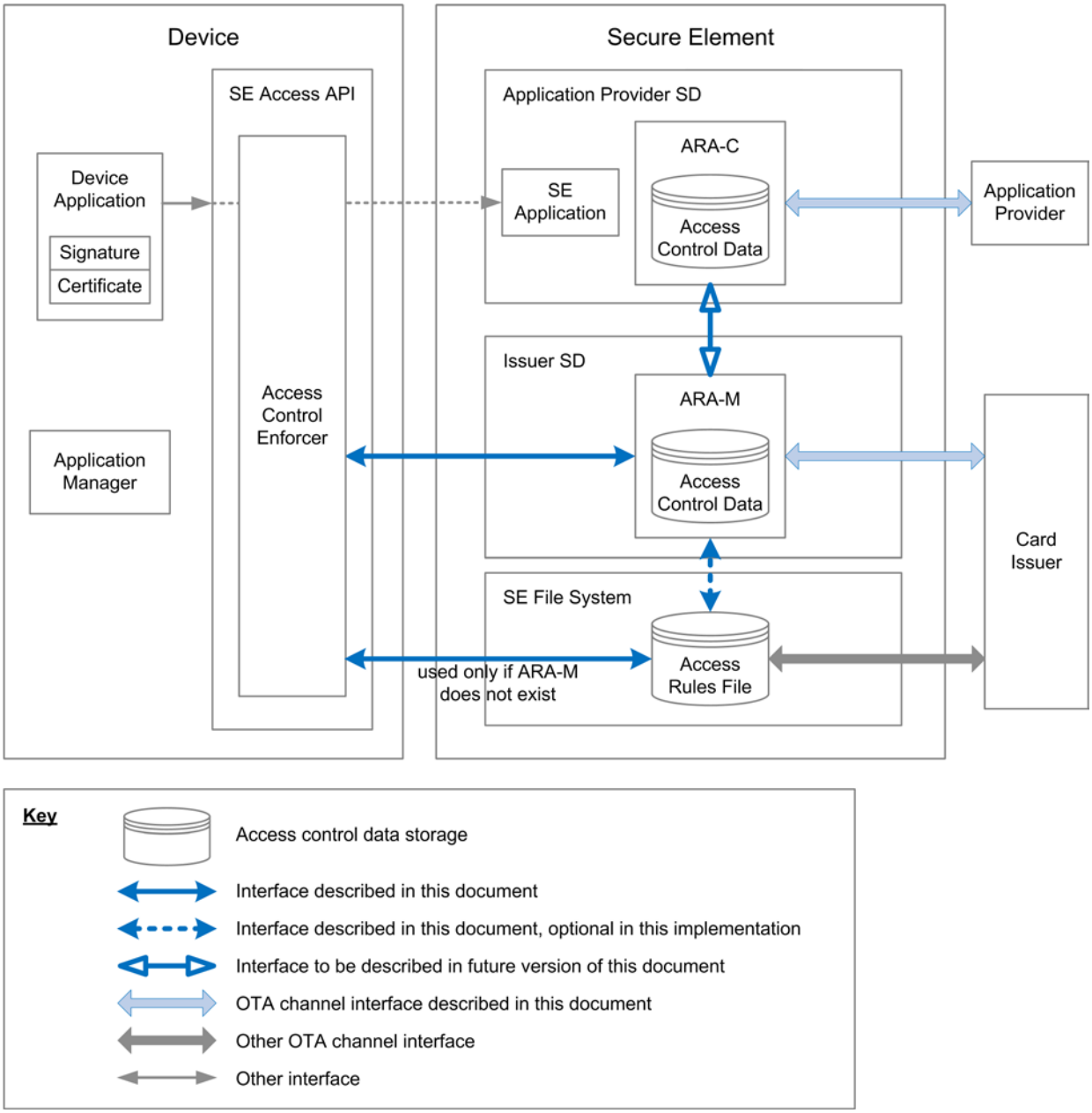
However, in this version of the GlobalPlatform SE Access Control specification, the interface between the ARA-M and the ARA-C is not yet defined. In the current version of this specification, the following items are implementation dependent:

- Authentication of ARA-C by ARA-M
- Authorization to define access rights by ARA-C
- Architecture and interaction between the ARA-M and the ARA-C
- Internal API for registration of ARA-C to ARA-M

2.3 Architecture with Access Rule File (ARF) Support

This specification can be used by any kind of Secure Element (e.g. embedded SE, microSD card with security controller, UICC, etc.). For some existing UICC implementations, access to applications is controlled via a set of elementary files, which are updated using Remote File Management (RFM) rather than RAM. This specification supports that mechanism as well, as illustrated in Figure 2-3.

Figure 2-3: Access Control Architecture with ARF Support

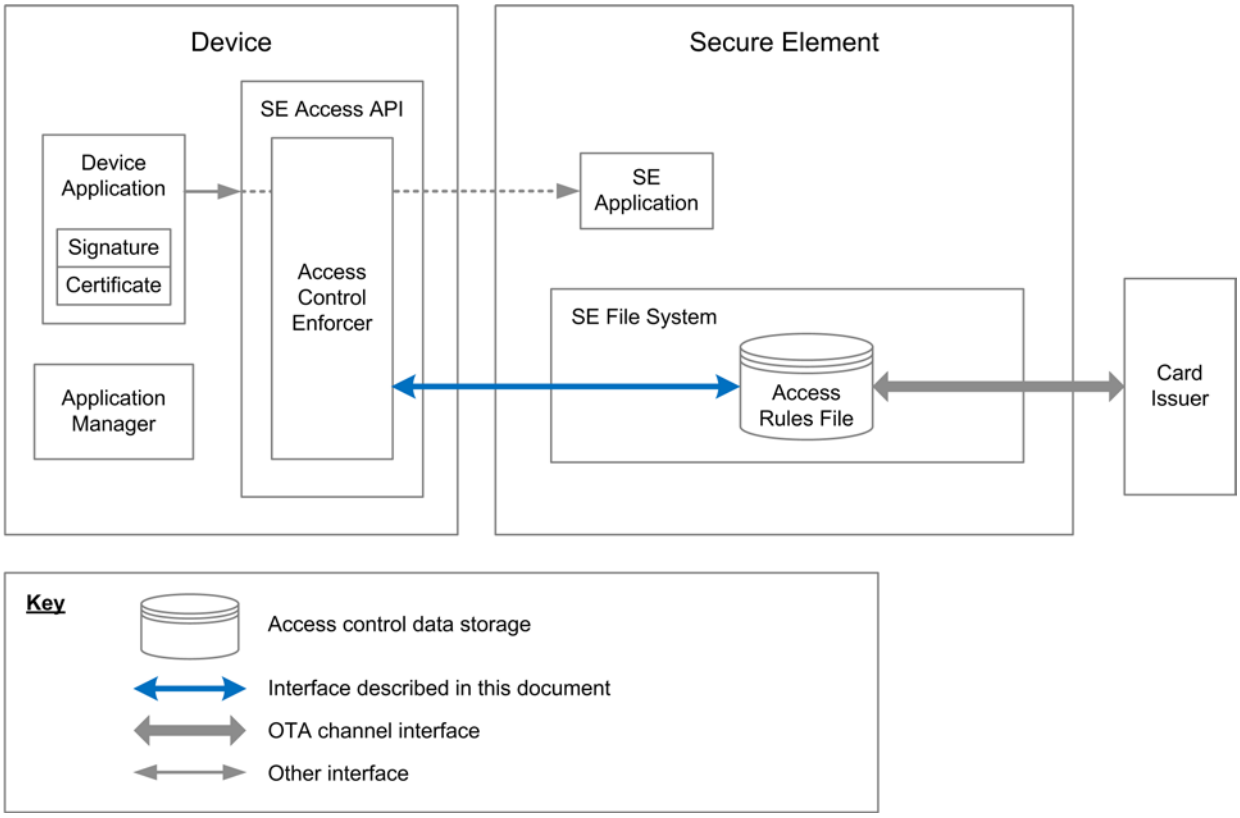


The Card Issuer defines access control rules for the SE applications, and supplies those rules to the ARA-M (as discussed in Chapter 4) or to the Access Rules File (ARF) (as discussed in Chapter 7).

When a device application attempts to access an SE application, the Access Control enforcer shall request the pertinent rules from the ARA-M. The ARA-M shall provide the appropriate rules, whether they are stored on the ARA-M, an ARA-C, or the ARF. (As mentioned in section 2.1, the Access Control enforcer may obtain the full set of rules in advance.) The Access Control enforcer shall permit the access only if the rules indicate that it is acceptable. It is the issuer's choice to decide whether or not the ARA-M has the ARF reading capability.

For the UICC, the following fallback shall be implemented: If the ARA-M is not present, the Access Control enforcer shall retrieve the access rules from the Access Rule Files (ARF), as illustrated in Figure 2-4.

Figure 2-4: Access Control Architecture with Access Rules File System Fallback



For information about migration from the legacy system support described in this section, see Annex F.

3 Access Control Rules

Each access control rule stored on the Secure Element specifies that:

- for a specific SE application, or for all other SE applications on a given SE
- a given device application or all other device applications have access rights to:
  - all APDUs, no APDUs, or selected APDUs
  - all NFC events or no NFC events

Because an access control rule may apply not only to an individual application or to multiple applications, and because separate rules may be defined in different places on the Secure Element (for example, in the ARA-M and in an ARA-C), access control rules may overlap and conflict, and a method must be defined to determine which rule to apply.

3.1 Introduction to Access Control Rule Conflict Resolution

This section discusses access control rule conflict resolution at a high level. Additional detail is provided later in this document.

The priority of a rule is not based on its reading order among other rules.

The policy to manage conflicting rules is based on three basic principles (in order):

1. **Specific rules have priority.**
2. **Rules associated with end-entity certificates have priority (in case of certificate chains).**
3. **Restrictive rules have priority.**

3.1.1 Specific Rules Have Priority

A specific rule is a rule associating explicitly:

- a Secure Element application by specifying its AID or by specifying the implicitly selected application.
- a device application by specifying its certificate.

Note: As a general matter when considering the rules, the implicitly selected application is seen as "a specific application with an unknown AID".

Security Warning: On a GlobalPlatform 2.2 Secure Element supporting the notion of a different implicitly selected application per logical channel, the "implicitly selected application" will apply regardless of the logical channel.

Rules are evaluated as most to least specific as defined in Table 3-1.

Table 3-1: Identifying Rules

Secure Element Application explicitly referenced?	Device Application explicitly referenced?	Priority
yes	yes	highest (specific)
yes	no	high
no	yes	low
no	no	least

3.1.2 End-Entity Certificates Have Priority

If a device application is signed with a certificate within a certificate chain, then during the search for the most specific rules, the search is based first on the certificate used to sign the application (the end entity certificate), then the next certificate up the chain, and so on, until either a certificate is found that has the appropriate level of specificity, or it is determined that no certificate in the chain has that level. Only then does the search proceed to the next lower level of specificity.

This is described further in section 4.3.

3.1.3 Restrictive Rules Have Priority

The most restrictive rules are those that forbid access the device application to access the SE application. Less restrictive rules permit access, but only when using certain APDUs. The least restrictive rules always permit the device application to access the SE application. The most restrictive rules have priority.

3.2 Access Control Rule Conflict Resolution

When several rules apply to the same access request, aggregation and conflict resolution shall be performed either by the ARA-M or by the Access Control enforcer:

- If the Access Control enforcer fetches all the rules from the Secure Element using GET DATA [All], then the Access Control enforcer is responsible for the merging and conflict resolution, if any.
- If the Access Control enforcer fetches the access rules for a particular access request using GET DATA [Specific], it is the responsibility of the ARA-M to merge and resolve the potential conflicts. The ARA-M shall determine whether several rules exist (e.g. in different storage locations within the SE) that apply to the defined reference (consisting of a specific or general device application identifier and a specific or general SE application identifier). Then the ARA-M shall resolve the conflicts, if any, and return access rule data merged as described in section 3.2.1.

3.2.1 Rules Combining

As described in section 3.1, the most specific rules have priority over more general rules. This strict priority shall be enforced by the Access Control enforcer. Thus, the Access Control enforcer shall first look for rules that apply to a specific SE application and a specific device application and shall look for less specific rules only if no specific rule was found. If one or more specific rules apply to a request, a less specific rule shall never be used for that request.[2]

If several rules apply to the same target SE application and have the same priority, then these rules are aggregated and more restrictive rules have priority over more permissive rules. If two rules have equally restrictive data, then both rules (NFC permission, APDU filters) are combined and thus both rules apply.

Table 3-2 summarizes which rule is applied when two rules (R1, R2) conflict. See also Annex D, which provides detailed examples. In this table 'R1+R2' refers to the combination of two rules where APDU filters and NFC event filters are merged.

Table 3-2: Access Control Rules Conflict Resolution

<table>
<tr><th colspan="3" rowspan="3">Conflicting rule resolution</th><th colspan="6">R1</th></tr>
<tr><th colspan="3">All</th><th colspan="3">AID</th></tr>
<tr><th>Never</th><th>APDU filter</th><th>Always</th><th>Never</th><th>APDU filter</th><th>Always</th></tr>
<tr><td rowspan="6">R2</td><td rowspan="3">All</td><td>Never</td><td>R1=R2</td><td>R2</td><td>R2</td><td colspan="3" rowspan="3">R1</td></tr>
<tr><td>APDU filter</td><td>R1</td><td>R1+R2</td><td>R2</td></tr>
<tr><td>Always</td><td>R1</td><td>R1</td><td>R1=R2</td></tr>
<tr><td rowspan="3">AID</td><td>Never</td><td colspan="3" rowspan="3">R2</td><td>R1=R2</td><td>R2</td><td>R2</td></tr>
<tr><td>APDU filter</td><td>R1</td><td>R1+R2</td><td>R2</td></tr>
<tr><td>Always</td><td>R1</td><td>R1</td><td>R1=R2</td></tr>
</table>

If a specific rule associates the hash of a device application with the AID of an SE application, then access to all the other device applications is denied unless a specific rule explicitly exists for this other device application.

[2] The search for the most specific rules is described in detail in section 4.2.2 and section 4.3.

4 Device Interface

The Access Control enforcer shall retrieve the SE access rules from the ARA-M installed on the targeted SE. Therefore the ARA-M provides an interface for retrieving the access rules. On this interface the Access Control enforcer can request either a specific access rule (corresponding to a specific SE application and a specific device application), or the complete set of access rules stored in the SE. The ARA-M shall support both options.

The ARA-M of each SE and the Access Control enforcer shall implement the GET DATA command, as defined in this section, to manage the access rules.

For a UICC, the Access Control enforcer shall implement in addition the following mechanism: If the ARA-M is not present, it shall retrieve the access rules from the Access Rule Files (ARF) as defined in Chapter 7.

The Access Control enforcer is in charge of interpreting the fetched access rules correctly and filtering the access according to these rules.

Access to a UICC is denied in all of the following cases:

- If neither ARA-M nor ARF is present on the UICC
- If the ARA-M is installed but cannot be accessed (being not selectable or locked for instance)
- If the ARA-M is installed but does not provide an access rule explicitly granting access
- If an error occurs during the reading and interpretation of the access rules (as discussed in section 7.3)

Access to any other SE is denied in all of the following cases:

- If the ARA-M is installed but cannot be accessed (being not selectable or locked for instance)
- If the ARA-M is installed but does not provide an access rule explicitly granting access
- If an error occurs during the reading and interpretation of the access rules

4.1 GET DATA Command

The GET DATA command is used to retrieve the access rules from the ARA-M and provides different modes:

Mode 1: Retrieve all access rules stored in the Secure Element.

This mode can be used to cache all access rules on the device to avoid repeatedly retrieving access rules from the Secure Element. Since the GET DATA command in this mode can be very time–consuming, it is recommended that this mode be performed only during the boot process of the device.

Mode 2: Retrieve a refresh tag indicating whether any access rules have been updated.

This mode can be used in conjunction with the previous one to determine whether the cached access rules on the device need to be refreshed. When using a cached version of the rule set, the device enforcer shall check whether a new version of the rules is available prior to applying cached rules.

Mode 3: Retrieve a specific access rule for a defined SE application (identified by the SE application's AID) and a device application (identified by the hash value of the device application's certificate).

This mode is an alternative to mode 1. This mode could be applied if mode 2 indicates that some SE access rules have been updated. Because the Access Control enforcer cannot know which rules have been updated, it shall either apply Mode 1 again to retrieve all access rules, or apply Mode 3 for each specific rule that is required before the device is rebooted.

The ARA-M shall consolidate all the access rules present in the SE (including access rules stored in the file system and the ARA-C).

Access rules specify that for a given SE application (or all SE applications on a given SE), all or selected device applications have access rights to:

- all APDUs, no APDUs, or selected APDUs
- NFC transaction events or no NFC transaction events

For NFC transaction events, if no rule explicitly specifies NFC permissions, permission shall be granted based on APDU channel rules. A device application authorized to establish an APDU channel with a secure element application is implicitly authorized to receive NFC events from this application.

4.1.1 Command Message

The GET DATA command message shall be coded according to Table 4-1.

Table 4-1: GET DATA Command Message

Code	Value	Meaning
CLA	'80' – '8F', 'C0' – 'CF', or 'E0' – 'EF'	As specified in [GPCardSpec]
INS	'CA'	GET DATA as specified in [GPCardSpec]
P1 P2	One of the following: 'FF 40': All 'FF 50': Specific 'DF 20': Refresh tag 'FF 60': Next	All: Request to obtain all access rules. Specific: Request to obtain specific access rules. Refresh tag: Request to obtain the refresh tag. Next: Request to obtain the remaining bytes which couldn't be fetched with the last command APDU.
LC	Absent or Length of REF-DO	
Data	Absent or REF-DO	Absent: If P1 P2 = [All], [Next], or [Refresh tag] REF-DO: If P1 P2 = [Specific]: REF-DO references a specific access rule.
Le	'00'	Expected length of the returned block

4.1.1.1 Command Message Tags

The GET DATA command can be applied iteratively with subsequent GET DATA commands if the access rule data to be fetched from the ARA is too large for one GET DATA command. The length field of the returned Access Rule Data Object Response-ALL-AR-DO/Response-AR-DO shall always indicate the full length of all expected AR-DOs (even if not all AR-DO bytes are present in the GET DATA response field) so that the Access Control enforcer can determine whether a subsequent GET DATA command is needed. The GET DATA command supports the iteration as follows:

1. GET DATA [All]: Fetches the first bytes of the Response-ALL-AR-DO.
2. GET DATA [Next]: Fetches the next (succeeding) bytes of the Response-ALL-AR-DO.

or:

1. GET DATA [Specific]: Fetches the first bytes of the Response-AR-DO.
2. GET DATA [Next]: Fetches the next (succeeding) bytes of the Response-AR-DO.

To retrieve access rules from the ARA-M, the command GET DATA [All] / GET DATA [Specific] must always be applied as the first command. A GET DATA [Next] command must be rejected by the ARA-M with SW '69 85' if the data retrieve process did not start with a GET DATA [All] / GET DATA [Specific].

4.1.1.2 Command Message Data Objects

If the GET DATA command includes P1 P2 = [Specific], then a REF-DO must be included in the Data field. This REF-DO contains an AID-REF-DO and a Hash-REF-DO which uniquely reference a specific set of access rules assigned for a given SE application (which is identified by its AID) and a device application (which is identified by the hash value of its certificate). REF-DO, AID-REF-DO, and Hash-REF-DO are defined in Chapter 6.

4.1.2 Response Message

The command GET DATA returns the requested access rules in different data objects (depending on the command request) in the response message Data field.

4.1.2.1 Response Message Data Objects

Depending on the GET DATA request, the response message Data field contains a Response-ALL-AR-DO, a Response-AR-DO, or a Response-RefreshTag-DO:

- The Response-ALL-AR-DO is mandatory for a GET DATA [All] request.
- The Response-AR-DO is mandatory for a GET DATA [Specific[request.
- The Response-RefreshTag-DO is mandatory for a GET DATA [Refresh tag] request.

Response-ALL-AR-DO

In response to a GET DATA [All] command, the ARA-M shall return all access rules stored in the Secure Element in the response message Data field within a Response-ALL-AR-DO. The length field of the Response-ALL-AR-DO shall always contain the full length of the values of all the REF-AR-DOs. If the Response-ALL-AR-DO is too large to fit in the GET DATA [All] response, then the remaining Response-ALL-AR-DO bytes can be retrieved using GET DATA [Next] commands. In this case, the response of the GET DATA [Next] doesn't include any Tag and Length fields but only the next bytes of the REF-AR-DO.

Table 4-2: Response-ALL-AR-DO

Tag	Length	Value	Meaning	Presence
'FF 40'	n or 0	$REF\text{-}AR\text{-}DO_1$ | … | $REF\text{-}AR\text{-}DO_x$ or Empty	**Value:** If access rules exist, a concatenation of all REF-AR-DOs on the SE. Empty if access rules do not exist. **Length:** n is the full length of all the REF-AR-DOs. If n is equal to zero, then there are no rules to fetch.	Mandatory

Response-AR-DO

If access rules exist in the Secure Element that corresponds to the REF-DO specified in the GET DATA [Specific] command, then the ARA-M must return those access rules in the response message Data field within a Response-AR-DO. The length field of the Response-AR-DO shall always contain the full length of the DO's value. If the Response-AR-DO is too large to fit in the GET DATA [Specific] response, then the remaining Response-AR-DO bytes can be retrieved using GET DATA [Next] commands. In this case, the response of the GET DATA [Next] doesn't include any Tag and Length fields but only the next bytes of the AR-DO.

Table 4-3: Response-AR-DO

Tag	Length	Value	Meaning	Presence
'FF 50'	n or 0	AR-DO or Empty	**Value:** An AR-DO if the referenced access rules exist. Empty if access rules do not exist for the defined reference. **Length:** n is the full length of the AR-DO. If n is equal to zero, then there are no rules to fetch.	Mandatory

Response-RefreshTag-DO

The GET DATA [Refresh tag] command shall return a Response-RefreshTag-DO containing a refresh tag that indicates whether changes have occurred in the access control data. This refresh tag is an attribute (8-byte random number) of the ARA-M and is newly generated when the ARA-M detects an update of access control data in the Secure Element. The ARA-M shall ensure that the new value is different from the previous one.

Table 4-4: Response-RefreshTag-DO

Tag	Length	Value	Meaning	Presence
'DF 20'	8	RefreshTag	**Value:** RefreshTag is an 8-byte random number. A new RefreshTag value indicates changes in the access control data stored in the SE.	Mandatory

4.1.2.2 Response Message Status Words

A successful execution of the command shall be indicated by status bytes '90 00'.

Table 4-5: GET DATA Response Message Status Words

SW1	SW2	Meaning
'65'	'81'	Memory problem
'67'	'00'	Wrong length in Lc
'69'	'85'	Conditions not satisfied
'6A'	'80'	Incorrect values in the command data
'6A'	'86'	Incorrect P1 P2
'6A'	'88'	Referenced data not found
'6D'	'00'	Invalid instruction
'6E'	'00'	Invalid class

4.2 Access Control Evaluation Steps

This section describes how the Access Control enforcer on the device shall behave when evaluating rules from ARA-M to be compliant with the SE access control policy.

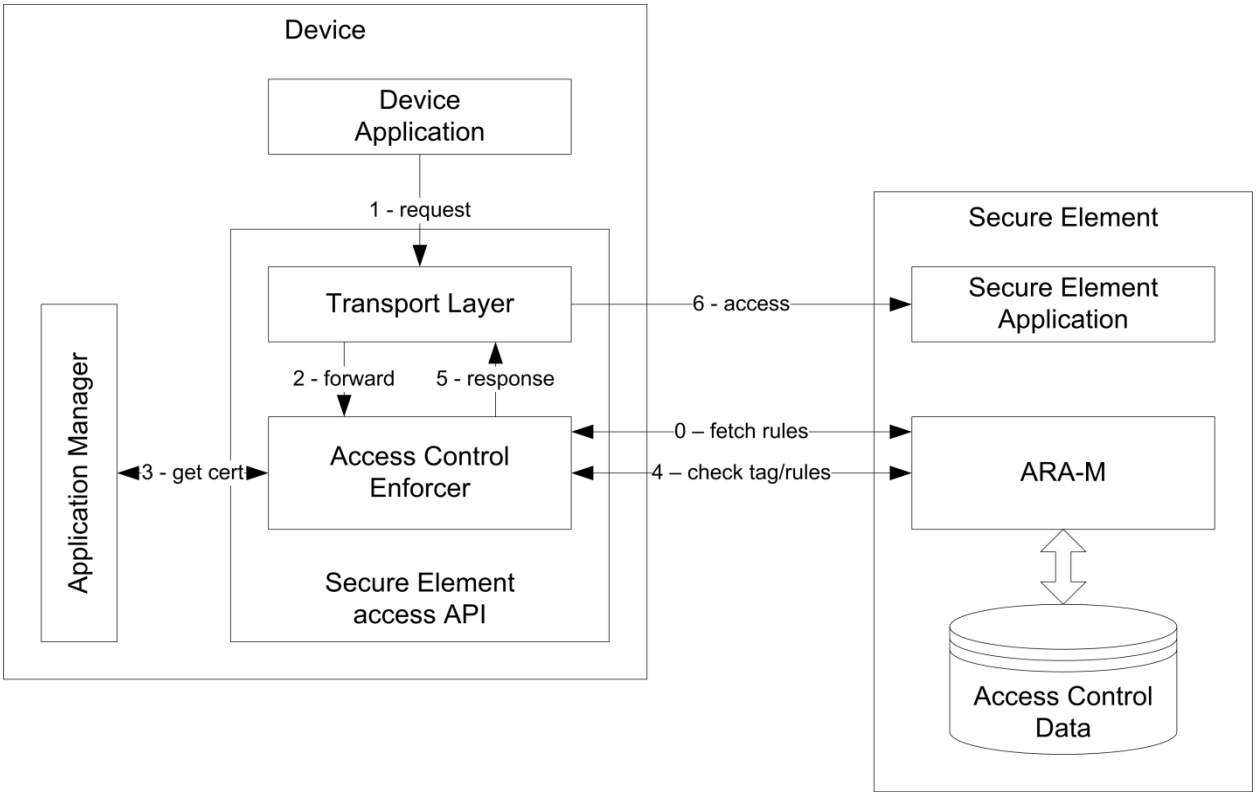
There are two distinct ways that the device can use the access control mechanisms on the Secure Element. Either the device fetches all the rules in advance from the Secure Element, or it queries the Secure Element application each time access is requested. The following sections explain each of those usage scenarios.

Note: This specification does not require that the Access Control enforcer check the validity of any certificate. The developer of the Access Control enforcer may choose to offer that functionality. It is assumed that the device OS providing the application signature to the enforcer can be trusted about the validity of the certificates and the corresponding signatures.

Note: This section discusses access control rules that apply to a device application as though the rules were identified by the application's certificate. In fact, the access control rules are stored and retrieved based on the hash of device application's certificate, not the certificate itself.

4.2.1 Usage with Rules Cached in the Device

Figure 4-1: Device Interface Sequence with Rules Caching



Sequence when the device uses caching of the rules:

Preliminary step: During initialization of the device, the Access Control enforcer fetches rules from all the Secure Elements available, using the GET DATA [All] command. Following this operation, rules must stay associated with the Secure Element they were fetched from, and apply only to access to that Secure Element.

Step 1: The device application uses the Secure Element access API to open a communication channel with an application residing in a Secure Element.

Step 2: This request is forwarded to the Access Control enforcer on the device.

Step 3: The Access Control enforcer retrieves the certificate used to sign the calling device application. If the device application has multiple signatures, the Access Control enforcer retrieves each certificate of each signature. The enforcer then computes the hash (currently SHA-1) of each of these certificates. If the device application is signed by chained certificates, consider the more detailed explanation in section 4.3.

Step 4: The Access Control enforcer fetches the refresh tag using the GET DATA [Refresh tag] from the ARA-M on the targeted Secure Element. If the refresh tag returned is different from the value previously obtained from this Secure Element, then one of the following occurs:

- The Access Control enforcer fetches a new copy of the whole rule set for the targeted Secure Element.
- To minimize response time and maintain a positive user experience, the Access Control enforcer completes the current request using an instant query, as discussed in section 4.2.2, then subsequently fetches the whole rule set for the targeted Secure Element.

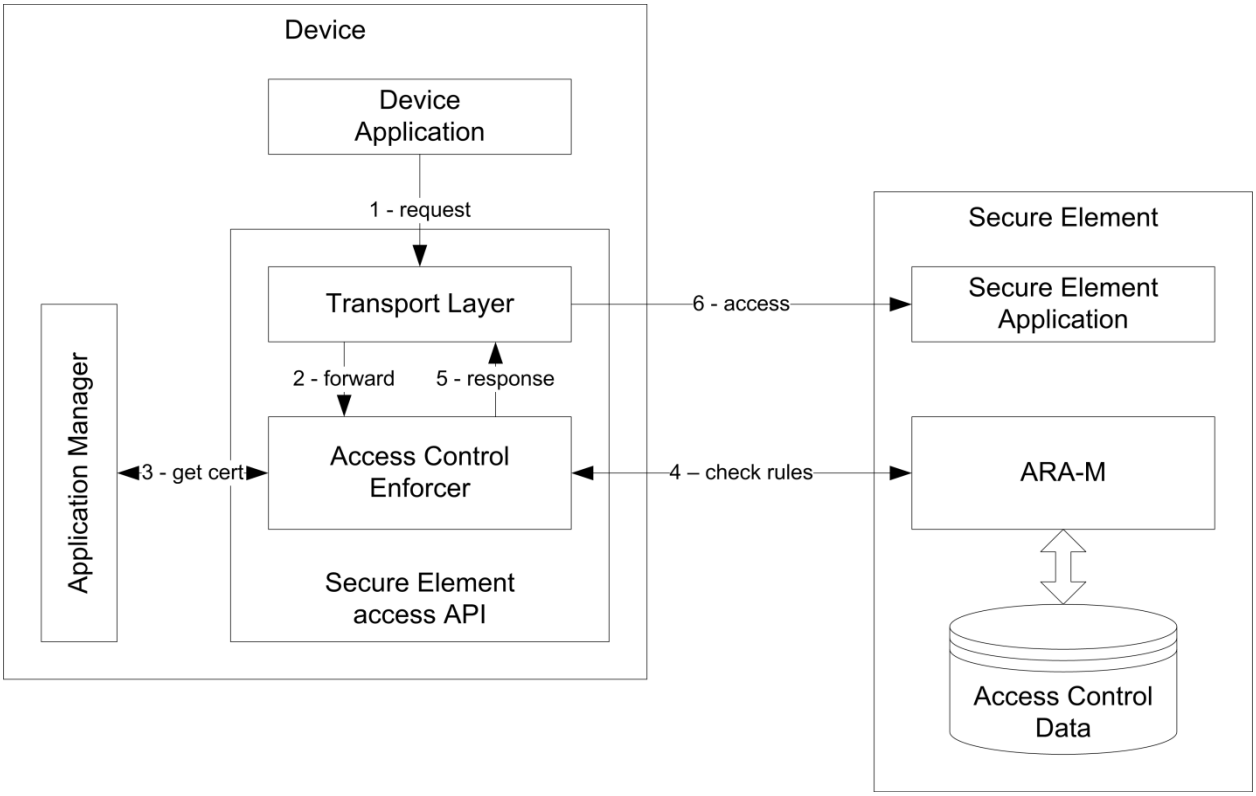
Step 5: The Access Control enforcer evaluates rules based on the calling application certificate hashes and the targeted AID on the Secure Element. See section 4.2.3. If the access is not granted, an error is returned to the calling application and no further action is taken. If the access is granted, then the Secure Element access API performs all operations necessary to open the channel to the Secure Element (e.g. manage channel command and application selection).

Step 6: Upon APDU transmission, the Access Control enforcer applies the APDU filtering applicable to the connection (if any).

Note: This section is also applicable when evaluating rules from ARF on UICC if the ARA-M is not present.

4.2.2 Usage with Instant Query to the ARA-M

Figure 4-2: Device Interface Sequence with Instant Query to the ARA-M



Sequence when the device uses instant query of the rules:

Step 1: The device application uses the Secure Element access API to open a communication channel with an application residing in a Secure Element.

Step 2: This request is forwarded to the Access Control enforcer on the device.

Step 3: The Access Control enforcer retrieves the certificate used to sign the calling device application. If the device application has multiple signatures, the Access Control enforcer retrieves each certificate of each signature. The enforcer then computes the hash (currently SHA-1) of each of these certificates. If the device application is signed by chained certificates, consider the more detailed explanation in section 4.3.

Step 4: The Access Control enforcer interrogates the ARA-M on the targeted Secure Element using the GET DATA [Specific]. The ARA-M returns the requested rules, including APDU filter details, if any. See section 4.2.3.

Step 5: The Access Control enforcer evaluates the rules to determine whether the application is allowed to access the Secure Element application. If the access is not granted, an error is returned to the calling application and no further action is taken. If the access is granted, then the Secure Element access API performs all operations necessary to open the channel to the Secure Element (e.g. manage channel command and application selection).

Step 6: Upon APDU transmission, the Access Control enforcer applies the APDU filtering applicable to the connection (if any).

4.2.3 Algorithm for Applying Rules

The Access Control enforcer shall retrieve the rules that shall be applied by checking for rules associated with the device's certificate according to the algorithm defined below. If not using cache the Access Control enforcer may have to issue several GET DATA [Specific] commands to the ARA-M to ensure that the right rule is retrieved.

The Access Control enforcer uses the following algorithm to retrieve an access rule for the device application (identified by its certificate) and the SE application (identified by its AID):

A) Search for a rule that is specific to the device application and to the SE application with AID:

SearchRuleFor(DeviceApplicationCertificate, AID)

If a rule exists, then apply this rule and stop the rule search.

According to the rule conflict resolution process defined in section 3.2.1, if a specific rule exists that associates another device application with the SE application identified by AID (e.g. there is a rule associating AID with the hash of another device application), then the ARA-M (when using GET DATA [Specific]) or the Access Control Enforcer (when using GET DATA [All]) shall set the result of SearchRuleFor(DeviceApplicationCertificate, AID) to NEVER (i.e. precedence of specific rules over generic rules).

B) If no rule fits condition A: Search for a rule that applies to all device applications not otherwise covered by a specific rule and is specific to the SE application with AID:

SearchRuleFor(<AllDeviceApplications>, AID)

If a rule exists, then apply this rule and stop the rule search.

C) If no rule fits condition A or B: Search for a rule that is specific to the device application and applies to all SE applications not otherwise covered by a specific rule :

SearchRuleFor(DeviceApplicationCertificate, <AllSEApplications>)

If a rule exists, then apply this rule and stop the rule search.

According to the rule conflict resolution process defined in section 3.2.1, if a specific rule exists that associates another device application with the SE application identified by AID (e.g. there is a rule associating AID with the hash of another device application), then the ARA-M (when using GET DATA [Specific]) or the Access Control Enforcer (when using GET DATA [All]) shall set the result of SearchRuleFor(DeviceApplicationCertificate, <AllSEApplications>) to NEVER (i.e. precedence of specific rules over generic rules).

D) If no rule fits condition A, B, or C: Search for a rule that applies to all device applications and to all SE applications not otherwise covered by a specific rule :

SearchRuleFor(<AllDeviceApplications>, <AllSEApplications>)

If a rule exists, then apply this rule.

Note: This section is also applicable when evaluating rules from ARF on UICC if the ARA-M is not present.

4.3 Management of Certificate Chains

As discussed in section 3.1.2, if the device application is signed with a certificate within a chain and if more than one certificate of that chain is associated with access control rules, then the rule associated with the certificate at the lowest hierarchical level in the chain that has an associated rule (which may be the end entity certificate) shall apply.

Figure 4-3: Processing of Chained Certificates

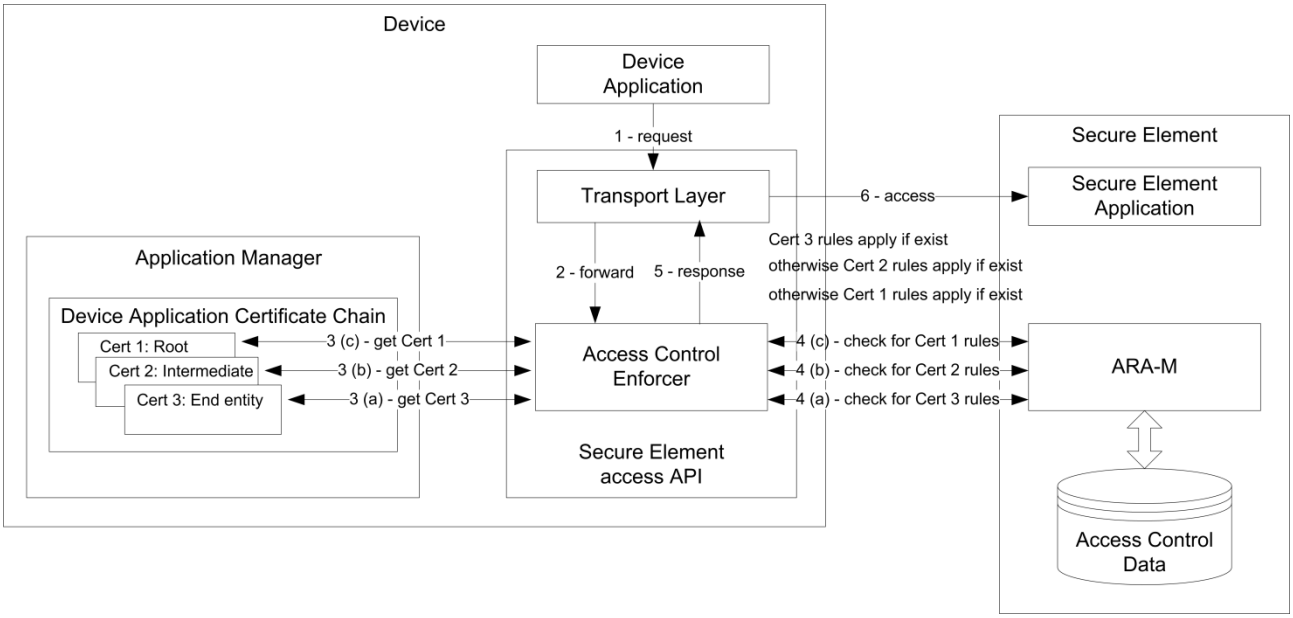


Figure 4-3 shows how the Access Control enforcer queries the rules from the ARA-M when a device application is signed by a certificate within a chain.

In the case of certificate chains, the Access Control enforcer shall retrieve the rules that shall be applied by checking the rules associated with the different certificates of the certificate chain according to the algorithm defined below. The Access Control enforcer may have to issue several GET DATA [Specific] commands to the ARA-M to ensure that the right rule is retrieved.

In steps A) and C) of the procedure in section 4.2.3, the Access Control enforcer uses the following algorithm to retrieve an access rule for the appropriate certificate in the certificate chain.

A) Search for a rule that is specific to the device application and to the SE application with AID:

1. SearchRuleFor(EndEntityCertificate, AID)

   If a rule exists, then apply this rule and stop the rule search.

2. SearchRuleFor(IntermediateCertificate<1>, AID)

   If a rule exists, then apply this rule and stop the rule search.

...

3. SearchRuleFor(IntermediateCertificate<n>, AID)

   If a rule exists, then apply this rule and stop the rule search.

4. SearchRuleFor(RootCertificate, AID)

   If a rule exists, then apply this rule and stop the rule search.

B) If no rule fits condition A: Search for a rule that applies to all device applications and is specific to the SE application with AID:

1. SearchRuleFor(<AllDeviceApplications>, AID)

   If a rule exists, then apply this rule and stop the rule search.

C) If no rule fits condition A or B: Search for a rule that is specific to the device application and applies to all SE applications:

1. SearchRuleFor(EndEntityCertificate, <AllSEApplications>)

   If a rule exists, then apply this rule and stop the rule search.

2. SearchRuleFor(IntermediateCertificate<1>, <AllSEApplications>)

   If a rule exists, then apply this rule and stop the rule search.

...

3. SearchRuleFor(IntermediateCertificate<n>, <AllSEApplications>)

   If a rule exists, then apply this rule and stop the rule search.

4. SearchRuleFor(RootCertificate, <AllSEApplications>)

   If a rule exists, then apply this rule and stop the rule search.

D) If no rule fits condition A, B, or C: Search for a rule that applies to all device applications and to all SE applications:

1. SearchRuleFor(<AllDeviceApplications>, <AllSEApplications>)

   If a rule exists, then apply this rule.

5 Remote Interface Based on RAM

Access rules for a Secure Element can be managed via Remote Application Management (RAM) update commands. Therefore the ARA-M as well as the ARA-C provides a remote interface which allows storing or deleting access rules in the ARA. Any remote management of the access control data should be done only over a secure channel protocol as defined by [GPCardSpec].

Each time some access rules are updated in the SE (either in the ARA-M or the ARA-C) the refresh tag owned by the ARA-M shall be updated. If the refresh tag has been updated, the device shall update the set of access rules previously retrieved via the GET DATA [All] command.

All update operations must be atomic: If an update procedure fails (e.g. due to power loss or communication errors) then the previous state has to be kept by the ARA until a successful update is completed.

Access rules stored in the ARA-M or in an ARA-C can also be retrieved via Remote Application Management (RAM) commands.

5.1 STORE DATA Command

The STORE DATA command is used to store access rules to the ARA-M or an ARA-C for a defined Secure Element application (identified by the SE application's AID) and device application (identified by the hash value of the device application's certificate).

The STORE DATA command can also be used to retrieve the access rules stored to the ARA-M or an ARA-C.

This command can be sent directly to the ARA or through its security domain, using the standard GlobalPlatform INSTALL [for personalization] command.

5.1.1 Command Message

The STORE DATA command message shall be coded according to Table 5-1.

Table 5-1: STORE DATA Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	As specified in [GPCardSpec]
INS	'E2'	STORE DATA as specified in [GPCardSpec]
P1 P2	P1: Reference control parameter P2: Block number	Reference control parameter as specified in [GPCardSpec] with: • b8 indicating if the command contains the last block of a command chain • b5 b4 set to 10, indicating a BER-TLV formatted command Data field • b1 set as follows: 0 for Command-Store-AR-DO Command-Delete-AR-DO Command-UpdateRefreshTag-DO Command-Register-ClientAIDs-DO 1 for Command-Get-AR-DO Command-GetAll-AR-DO Command-Get-ClientAIDs-DO Command-GetNext-AR-DO Block number as specified in [GPCardSpec].
Lc	Length of the DO in the Data field	

Code	Value	Meaning	
Data	One of the following: Command-Store-AR-DO Command-Delete-AR-DO Command-UpdateRefreshTag-DO Command-Register-ClientAIDs-DO Command-Get-AR-DO Command-GetAll-AR-DO Command-Get-ClientAIDs-DO Command-GetNext-AR-DO	Command-Store-AR-DO	Sent to an ARA to store access rules into this ARA of the SE
		Command-Delete-AR-DO	Sent to an ARA to delete access rules from this ARA of the SE
		Command-UpdateRefreshTag-DO	Sent to an ARA to update the refresh tag managed by the ARA-M
		Command-Register-ClientAIDs-DO	Sent to the ARA-M to register an ARA-C identified by its AID.
		Command-Get-AR-DO	Sent to an ARA to retrieve access rules stored in this ARA
		Command-GetAll-AR-DO	Sent to the ARA-M to retrieve all access rules from the SE
		Command-Get-ClientAIDs-DO	Sent to the ARA-M to retrieve the AID of all the ARA-Cs registered to the ARA-M.
		Command-GetNext-AR-DO	Sent to an ARA to retrieve the remaining access rules from this ARA.
Le	Absent or '00'	Not present for: Command-Store-AR-DO Command-Delete-AR-DO Command-UpdateRefreshTag-DO Command-Register-ClientAIDs-DO '00' for: Command-Get-AR-DO Command-GetAll-AR-DO Command-Get-ClientAIDs-DO Command-GetNext-AR-DO	

The STORE DATA command can be applied iteratively with subsequent STORE DATA commands if the Access Rule data objects which shall be stored into the ARA are too large for one STORE DATA command. The Length field of the AR command data object shall always contain the full length of all AR data objects so that the ARA can determine whether a subsequent STORE DATA command is expected. If a subsequent STORE DATA is not received, then the ARA shall ignore the already received AR data objects.

5.1.1.1 Command Message Data Objects

The access rules within an ARA-M/ARA-C can be managed with the access rule command data objects described in this section:

Command-Store-AR-DO

This data object stores access rules to the ARA.

In this case, no response data is expected; therefore, bit 1 (the rightmost bit) of the reference control parameter P1 shall be set to 0 to indicate to the card that the command is an ISO/IEC 7816-4 [7816-4] Case 3 command, and the Le field shall not be present.

Table 5-2: Command-Store-AR-DO

Tag	Length	Value	Meaning	Presence
'F0'	n	REF-AR-DO	Stores the specified access rule to the ARA-M/ARA-C. **Value:** The REF-AR-DO that shall be stored into the ARA-M/ARA-C. If the access rule references (AID and hash) of the REF-AR-DO already exist in the ARA-M/ARA-C, then the corresponding existing access rules must be overwritten by the access rules of the same type (NFC or APDU) in this command. **Length:** n is the full length of all value bytes even if not all value bytes are present in the command Data field. The remaining value bytes can follow in subsequent STORE DATA commands.	Mandatory

Command-Delete-AR-DO

This data object deletes access rules in the ARA.

In this case, no response data is expected; therefore, bit 1 (the rightmost bit) of the reference control parameter P1 shall be set to 0 to indicate to the card that the command is a [7816-4] Case 3 command, and the Le field shall not be present.

Table 5-3: Command-Delete-AR-DO

Tag	Length	Value	Meaning	Presence
'F1'	n or 0	One of the following: AID-REF-DO REF-DO REF-AR-DO Empty	Deletes the specified access rule from the ARA-M/ARA-C. **Value:** AID-REF-DO: All access rules assigned to this AID-REF-DO are deleted. REF-DO: All access rules assigned to this REF-DO are deleted. REF-AR-DO: The APDU-AR-DO and NFC-AR-DO in the REF-AR-DO must be empty. This means a value field does not exist and the length field is set to 0. All access rules that match one of the APDU-AR-DO/NFC-AR-DO tag specified in REF-AR-DO are deleted. Empty: All access rules are deleted. **Length:** n is the full length of all value bytes 0 if no data object is specified.	Mandatory

Command-UpdateRefreshTag-DO

This data object updates the refresh tag managed by the ARA-M.

In this case, no response data is expected; therefore, bit 1 (the rightmost bit) of the reference control parameter P1 shall be set to 0 to indicate to the card that the command is a [7816-4] Case 3 command, and the Le field shall not be present.

Table 5-4: Command-UpdateRefreshTag-AR-DO

Tag	Length	Value	Meaning	Presence
'F2'	0	Empty	Request the ARA-M to update the refresh tag.	Mandatory

Command-Register-ClientAIDs-DO

This data object can be used to register ARA-Cs to the ARA-M. An ARA-C is identified by its AID.

This data object shall only be processed by the ARA-M. The ARA-M shall register all the ARA-Cs identified by the provided AIDs. If a provided AID does not correspond to an installed ARA-C, then this AID shall be ignored by the ARA-M.

Table 5-5: Command-Register-ClientAIDs-DO

Tag	Length	Value	Meaning	Presence
'F7'	n	AID-REF-DO$_1$| … | AID-REF-DO$_x$	Register ARA-Cs to the ARA-M provided these ARA-Cs are installed on the SE. Value: One or more AID-REF-DOs, each corresponding to the AID of an ARA-C to be registered to the ARA-M. Length: n is the full length of all value bytes even if not all value bytes are present in the command Data field. The remaining value bytes can follow in subsequent STORE DATA commands.	Mandatory

Command-Get-AR-DO

This data object is used to retrieve all the access rules stored to the ARA.

In this case, bit 1 (the rightmost bit) of the reference control parameter P1 is set to 1 to indicate to the card that the command is a [7816-4] Case 4 command, the Le field shall be set to '00', and therefore, response data is expected.

Table 5-6: Command-Get-AR-DO

Tag	Length	Value	Meaning	Presence
'F3'	n or 0	Empty or AID-REF-DO	Get the access rules stored in the ARA-M or ARA-C. **Value:** Empty: If this STORE DATA (Command-Get-AR-DO) is sent to an ARA-C, get all the access rules stored to this ARA-C. If this STORE DATA (Command-Get-AR-DO) is sent to the ARA-M, get all the access rules stored to the ARA-M itself. AID-REF-DO: This data object can only be used in a STORE DATA (Command-Get-AR-DO) sent to the ARA-M. The AID-REF-DO shall correspond to the AID of an ARA-C already registered to the ARA-M. In this case, the ARA-M shall return only all the rules stored to this ARA-C. **Length:** n is the full length of the AID-REF-DO. 0 if no AID-REF-DO is specified.	Mandatory

Command-GetAll-AR-DO

This data object is used to retrieve all the access rules stored to the SE. This command can only be sent to the ARA-M.

In this case, bit 1 (the rightmost bit) of the reference control parameter P1 shall be set to 1 to indicate to the card that the command is a [7816-4] Case 4 command, the Le field shall be set to '00', and therefore, response data is expected.

Table 5-7: Command-GetAll-AR-DO

Tag	Length	Value	Meaning	Presence
'F4'	0	Empty	Get all the access rules stored in the SE as sent to the device Access Control enforcer using a GET DATA [All] command.	Mandatory

Command-Get-ClientAIDs-DO

This data object retrieves AIDs of ARA-Cs registered to the ARA-M.

This data object shall only be processed by the ARA-M. The ARA-M shall return the Response-ARAC-AID-DO (as described in Table 5-10) holding a list of AID-REF-DOs indicating the AID of each of the ARA-Cs registered to the ARA-M.

Table 5-8: Command-Get-ClientAIDs-DO

Tag	Length	Value	Meaning	Presence
'F6'	0	Empty		Mandatory

Command-GetNext-AR-DO

This data object is used to retrieve the remaining access rules after a first Command-Get-AR-DO, Command-GetAll-AR-DO, or Command-Get-ClientAIDs-DO when all the requested rules couldn't be fetched in response to the first command.

In this case, bit 1 (the rightmost bit) of the reference control parameter P1 shall be set to 1 to indicate to the card that the command is a [7816-4] Case 4 command, the Le field shall be set to '00', and therefore, response data is expected.

The command Command-GetNext-AR-DO shall be rejected with SW '69 85', if the data retrieve process did not start with a Command-Get-AR-DO, Command-GetAll-AR-DO, or Command-Get-ClientAIDs-DO.

Table 5-9: Command-GetNext-AR-DO

Tag	Length	Value	Meaning	Presence
'F5'	0	Empty	Get the remaining access rules after a first Command-Get-AR-DO, Command-GetAll-AR-DO, or Command-Get-ClientAIDs-DO, when all the requested rules couldn't be fetched in response to this first command	Mandatory

5.1.2 Response Message

5.1.2.1 Response Message Data Objects

If the command data object used in the STORE DATA request is Command-Store-AR-DO, Command-Delete-AR-DO, Command-UpdateRefreshTag-DO, or Command-Register-ClientAIDs-DO, then there is no response data.

If the command data object used in the STORE DATA request is Command-Get-AR-DO or Command-GetAll-AR-DO, then the response message Data field contains a Response-ALL-AR-DO as defined in Table 4-2:

- In response to STORE DATA (Command-Get-AR-DO), Response-ALL-AR-DO includes all the access rules stored in the current ARA.
- In response to STORE DATA (Command-GetAll-AR-DO), Response-ALL-AR-DO includes all the access rules stored in the SE as sent to the device Access Control enforcer in response to the command GET DATA [All].

If the command data object used in the STORE DATA request is Command-Get-ClientAIDs-DO, then the response message Data field contains a Response-ARAC-AID-DO as defined in Table 5-10. The Response-ARAC-AID-DO includes the AIDs of all the ARA-Cs registered to the ARA-M.

In each case (Command-Get-AR-DO, Command-GetAll-AR-DO, or Command-Get-ClientAIDs-DO), if the Response-ALL-AR-DO or Response-ARAC-AID-DO is too large to fit in the STORE DATA response, then the remaining bytes can be retrieved using STORE DATA (Command-GetNext-AR-DO) commands.

In response to STORE DATA (Command-GetNext-AR-DO), the response data includes the remaining bytes of the Response-ALL-AR-DO or Response-ARAC-AID-DO that couldn't be fetched in response to the initial STORE DATA (Command-Get-AR-DO), STORE DATA (Command-GetAll-AR-DO), or STORE DATA (Command-Get-ClientAIDs-DO). The response of the STORE DATA (Command-GetNext-AR-DO) doesn't include any Tag Length fields but only the next bytes of the Response-ALL-AR-DO or Response-ARAC-AID-DO.

Response-ARAC-AID-DO

In response to STORE DATA (Command-Get-ClientAIDs-DO), the ARA-M shall return the AID of each of the ARA-Cs currently registered within a Response-ARAC-AID-DO.

Table 5-10: Response-ARAC-AID-DO

Tag	Length	Value	Meaning	Presence
'FF 70'	N	AID-REF-DO$_1$ | … | AID-REF-DO$_x$ or Empty	Value: The list of the AIDs of all the ARA-Cs registered to the ARA-M. Empty if no ARA-C is registered to the ARA-M. Length: n is the length of all the AIDs returned. 0 if no ARA-C is registered to the ARA-M.	Mandatory

5.1.2.2 Response Message Status Words

A successful execution of the command shall be indicated by status bytes '90 00'.

Table 5-11: STORE DATA Response Message Status Words

SW1	SW2	Meaning
'63'	'81'	Rule successfully stored but an access rule already exists for this target
'65'	'81'	Memory problem
'67'	'00'	Wrong length in Lc
'69'	'82'	Security status not satisfied
'69'	'85'	Conditions not satisfied
'6A'	'80'	Incorrect values in the command data
'6A'	'84'	Not enough memory space
'6A'	'86'	Incorrect P1 P2
'6A'	'88'	Referenced data not found
'6A'	'89'	Conflicting access rule already exists in the Secure Element
'6D'	'00'	Invalid instruction
'6E'	'00'	Invalid class

6 General Data Objects

When storing and retrieving access rules from and to the Secure Element, the access rules and access rule references shall be coded in BER-TLV data objects to indicate the type and length of values.

In order to anticipate future evolutions, the Access Control enforcer and the ARA-M/ARA-C shall ignore unknown BER-TLV objects as long as the mandatory data objects described in this specification are made available to perform the action requested from the ARA-M, ARA-C, or Access Control enforcer components.

In the definitions below, when a DO is constructed from other DOs, the order as defined in the DO description shall be enforced.

6.1 Access Rule Reference Data Objects

The GET DATA and STORE DATA command require a set of data objects in the Data field for referencing and assigning the access rule data which shall be read from the SE or stored into the SE.

The AID and hash reference data objects assign access rules to a specific SE application and device application. If access rules shall be stored into the ARA or retrieved from the ARA it is necessary to specify the SE application and device application to create a unique assignment to these access rules. The SE application is uniquely identified by the AID specified in the AID reference data object (AID-REF-DO) whereas the device application is uniquely identified by the hash value of its certificate specified in the hash reference data object (Hash-REF-DO).

AID-REF-DO

The AID-REF-DO shall be used to store and retrieve the corresponding access rules for an SE application (which is identified by its AID) to and from the ARA. Two different AID reference data objects exist and one of these can be chosen and applied for a GET DATA and STORE DATA command:

Table 6-1: AID-REF-DO

Tag	Length	Value	Meaning
'4F'	5-16 or 0	AID or Empty	**Value:** AID: Identifies the specific SE application for which rules are to be stored or retrieved Empty: Indicates that the rules to be stored or retrieved are associated with all SE applications not covered by a specific rule **Length:** 5-16 for an AID according to ISO/IEC 7816-5 0 for empty value field
'C0'	0	Empty	Implicitly selected application (all channels)

Hash-REF-DO

The Hash-REF-DO shall be used to store and retrieve the corresponding access rules for a device application (which is identified by the hash value of its certificate) to and from the ARA:

Table 6-2: Hash-REF-DO

Tag	Length	Value	Meaning
'C1'	20 or 0	Hash or Empty	**Value:** Hash: Identifies the specific device application for which rules are to be stored or retrieved Empty: Indicates that the rules to be stored or retrieved are associated with all device applications not covered by a specific rule **Length:** 20 for 20-byte SHA-1 hash value 0 for empty value field

REF-DO

The REF-DO contains an AID-REF-DO and a Hash-REF-DO which uniquely reference a specific set of access rules assigned for a given SE application (which is identified by its AID) and a device application (which is identified by the hash value of its certificate)..

Table 6-3: REF-DO

Tag	Length	Value	Meaning
'E1'	n	AID-REF-DO | Hash-REF-DO	**Value:** A concatenation of an AID-REF-DO and a Hash-REF-DO. **Length:** n is the total length of all data objects in Value.

REF-AR-DO

The REF-AR-DO contains an access rule data object and its corresponding references for the SE application (AID reference) and device application (hash reference).

Table 6-4: REF-AR-DO

Tag	Length	Value	Meaning
'E2'	n	REF-DO | AR-DO	**Value:** A concatenation of a REF-DO and an AR-DO. The REF-DO must correspond to the succeeding AR-DO. **Length:** n is the total length of all data objects in Value.

6.2 Access Rule Data Objects

The ARA in the Secure Element can store and retrieve access rules for APDU access and for NFC event access, which are defined in different access rule data objects.

AR-DO

The AR-DO contains one or two access rules of type APDU or NFC.

Table 6-5: AR-DO

Tag	Length	Value	Meaning
'E3'	n	One of the following: APDU-AR-DO NFC-AR-DO APDU-AR-DO | NFC-AR-DO	**Value:** An APDU-AR-DO, or an NFC-AR-DO, or a concatenation of an APDU-AR-DO and an NFC-AR-DO. **Length:** n is the total length of all data objects in value.

APDU-AR-DO

An APDU access rule data object defines an access rule for APDU access. The APDU access can either be restricted by a general rule based on an “access is NEVER/ALWAYS allowed” policy or by a specific rule based on APDU filters which defines the range of allowed APDUs more precisely.

Table 6-6: APDU-AR-DO

Tag	Length	Value	Meaning
'D0'	1 or n*8	One of the following: '00' '01' APDU filter 1 | … | APDU filter n	**Value:** Contains a general APDU access rule: NEVER ('00'): APDU access is not allowed ALWAYS('01'): APDU access is allowed or Contains a specific APDU access rule based on one or more APDU filter(s): APDU filter: 8-byte APDU filter consists of: 4-byte APDU filter header (defines the header of allowed APDUs, i.e. CLA, INS, P1, and P2 as defined in [7816-4]) 4-byte APDU filter mask (bit set defines the bits which shall be considered for the APDU header comparison) An APDU filter shall be applied to the header of the APDU being checked, as follows: if((APDUHeader & APDU_filter_mask) == APDU_filter_header) then allow APDU **Length:** 1 if value contains a general APDU access rule. n*8 if value contains a specific APDU access rule, where n is the number of APDU filters included in value.

NFC-AR-DO

In the NFC use case, a mobile device application gathers information from its associated card application using the SE access API. However, when the card application wants to trigger its associated mobile application, it sends an HCI EVT_TRANSACTION according to ETSI TS 102 622 [102 622] over SWP to the device. This event is handled by the device, which starts the corresponding device application. In some other implementations, this event is generated by the selection of the application on the Secure Element, and then publicized by the NFC driver stack on the device. Disclosure of such events to malicious applications can lead to phishing and denial of service attacks.

To prevent this, it shall be possible to use the device application's signature to authorize device applications to receive NFC events issued by the Secure Element application.

An NFC event data object defines an access rule for generating NFC events for a specific device application. The NFC event access can be restricted by a rule based on an "event access is NEVER/ ALWAYS allowed" policy.

Table 6-7: NFC-AR-DO

Tag	Length	Value	Meaning
'D1'	1	'00', '01'	**Value:** Contains an NFC event access rule: NEVER ('00'): NFC event access is not allowed. ALWAYS ('01'): NFC event access is allowed.

If no rule specifies access for the NFC event, then the Access Control enforcer shall allow the sending of the NFC event to the targeted device application in all cases, unless this device application is not allowed to access the SE application which is the source of the transactionStructure of Access Rule Files (ARF).

7 Structure of Access Rule Files (ARF)

7.1 Background

The Access Rule Files (ARF) can be stored in a PKCS#15 file structure in the Secure Element. The following sections define the file system structure. The ASN.1 object description is based on the implicit ASN.1 encoding (see [X.690]), except when explicitly mentioned (e.g. AID).

7.1.1 File Paths

All the paths used for the access control mechanism described in this specification (DODF, ACMF, and ACRF files) shall be relative paths from a PKCS#15 directory. The full path starting from the UICC MF shall not be used. The path encoding is defined in [PKCS15].

7.1.2 File Padding

If needed, the end of the files shall be padded using 0xFF, and only 0xFF, bytes. The parsing engine used to manage the files involved in the access control mechanism shall support such padding.

7.1.3 PKCS#15 Selection

Selection of the PKCS#15 file structure or application is not within the scope of this specification and can be managed in different ways by the devices.

However a recommended way to perform the selection of the PKCS#15 application is to use the following sequence:

Step 1: the device sends a SELECT_BY_NAME command with PKCS#15 AID (A0 00 00 00 63 50 4B 43 53 2D 31 35). If the select is successful, the device can start reading PKCS#15 files (ODF, DODF…)

Step 2: if the previous select fails, the device sends SELECT commands to select the MF and the EF DIR, and then reads the EF DIR in order to locate an entry with the PKCS#15 AID. If a matching entry is found, the device must select the PKCS#15 DF path, and then it can start reading PKCS#15 files (ODF, DODF…)

7.1.4 PKCS#15 DODF

The PKCS#15 DODF used as the entry point to the access control data has an OidDO entry with an OID that must be in the GlobalPlatform scope (under {iso(1) member-body(2) country-USA(840) Global-Platform(114283)},).

The registered OID for this specification is: {iso(1) member-body(2) country-USA(840) Global-Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}

According to [PKCS15], the DODF shall include a DataType with an oidDO entry. The DataType oidDO entry is defined as PKCS15Object {CommonDataObjectAttributes, NULL, OidDO}. The presence of the oidDO entry is mandatory in the DODF.

The OidDO structure of the DODF DataType oidDo entry shall contain an id value set to the registered OID defined above and a value that is a path structure to the Access Control Main File. This path structure is defined with the following ASN.1 syntax:

```
Path ::= SEQUENCE {
  path OCTET STRING,
  index INTEGER (0..65535) OPTIONAL,
  length [0] INTEGER (0..65535) OPTIONAL
}( WITH COMPONENTS {..., index PRESENT, length PRESENT}|
WITH COMPONENTS {..., index ABSENT, length ABSENT})
-- the path of the Access Control Main File (as per PKCS#15)
```

The CommonDataObjectAttributes structure of the DODF DataType oidDo entry may contain an applicationName or an applicationOID. The applicationName and the applicationOID are considered as informative in this specification. The detection of the presence of the ARF structure shall be done checking that the id value of the DataType oidDO entry is the registered OID defined above.

7.1.5 The Access Control Main File (ACMF)

The Access Control Main File or ACMF (referenced from the DODF) has the following structure:

Table 7-1: Access Control Main File (ACMF)

Identifier: 'xxxx'	Structure: transparent file	Mandatory
File length: n bytes	Update activity: low	
Access Conditions: READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATE ADM		

Bytes	Description	M/O	Length
1 to n	AccessControlMainFile	M	n bytes

There shall be only one ACMF file per Secure Element. If a Secure Element contains several ACMF files, then the security shall be considered compromised and the Access Control enforcer shall forbid access to all the Secure Element applications for, and only for, this specific Secure Element.

The AccessControlMainFile object is related to the Secure Element that contains it.

The AccessControlMainFile object contains the refresh tag and the path to the rules (i.e. the path to the Access Control Rules File); additional fields may be added in future versions of this specification.

The AccessControlMainFile object is defined using the following ASN.1 syntax:

```
-- The access control main file object
AccessControlMainFile ::= SEQUENCE {
  -- the refresh tag
  refreshTag OCTET STRING (SIZE(8)),

  -- the path to the access control rules file
  rulesFile Path,

  -- RFU
  ...
}
```

7.1.6 The Access Control Rules File (ACRF)

The rules are stored in the Access Control Rules File or ACRF (referenced from the ACMF), which has the following structure:

Table 7-2: Access Control Rules File (ACRF)

Identifier: 'xxxx'	Structure: transparent file	Mandatory
File length: n bytes	Update activity: low	
Access Conditions: READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATE ADM		

Bytes	Description	M/O	Length
1 to n	List of Rules	M	n bytes

There shall be only one ACRF file per Secure Element. If a Secure Element contains several ACRF files, then the security shall be considered compromised and the Access Control enforcer shall forbid access to all the Secure Element applications for, and only for, this specific Secure Element.

Each Rule object explicitly or implicitly identifies a set of Secure Element applications, and refers to the Access Control Conditions File that describes how these applications can be accessed.

The Rule object is defined using the following ASN.1 syntax:

```
-- An access control rule entry
Rule ::= SEQUENCE {
  -- the target of this policy entry,
  target Target,

  -- the path to the access control conditions file applicable
  -- for this target
  conditionsFile  Path,

  -- RFU
  ...
}
```

The Target object indicates whether the rule applies to one Secure Element application (identified by its AID), or the implicitly selected application, or all other applications (all the Secure Element applications that are not explicitly protected by a specific rule).

It is defined using the following ASN.1 syntax:

```
-- An access control target: either a named application,
-- the implicitly selected application, or all other applications
Target ::= CHOICE {
  -- the AID of the targeted Secure Element application
  aid [0]EXPLICIT AID,

  -- the (unnamed) default selected Secure Element
  -- application (applies to implicitly selected application
  -- on all logical channels)
  default [1]NULL,
```

```
    -- identifies all other applications
    -- that are not referenced in another rule
    others [2]NULL,

    -- RFU
    ...
}
```

The AID object uses the following ASN.1 syntax:

```
-- as per ISO7816-5
AID ::= OCTET STRING
```

7.1.7 The Access Control Conditions File (ACCF)

The conditions referred to by a Rule object are stored in the Access Control Conditions File or ACCF.

The conditions are expressed as a list of entries, each entry containing a SHA-1 hash of a certificate identifying an authorized application issuer.

If this file is empty, then any Rule object pointing to this file is denying all device applications access to the Secure Element applications pointed to by the Rule object.

If this file contains a condition without a certificate hash, then any Rule object pointing to this file is granting any device application access to the Secure Element applications pointed to by the Rule object.

Table 7-3: Access Control Conditions File (ACCF)

Identifier: 'xxxx'	Structure: transparent file	Mandatory
File length: n bytes	Update activity: low	
Access Conditions: READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATE ADM		

Bytes	Description	M/O	Length
1 to n	List of Conditions	M	n bytes

The Condition object is defined using the following ASN.1 syntax:

```
-- A Condition entry
Condition ::= SEQUENCE {
  -- the hash of the certificate of the authorized entity;
  -- if not indicated, then the Rule pointing to this Condition
  -- applies to all the device applications
  cert   CertHash OPTIONAL,
  accessRules     [0]AccessRules OPTIONAL,

  -- RFU
  ...
}
-- SHA1 of the certificate of the authority being granted access
CertHash   ::= OCTET STRING (SIZE(20))

-- Each type of AccessRule can occur not more than once
-- in this sequence
AccessRules ::= SEQUENCE OF AccessRule

AccessRule ::=CHOICE {
apduAccessRule    [0]APDUAccessRule,
nfcAccessRule           [1]NFCAccessRule
}

APDUAccessRule ::= CHOICE {
   apduPermission [0] APDUPermission,
   apduFilter [1] APDUFilters
   -- RFU
   ...
```

```
}

-- TRUE means APDU access is allowed,
-- FALSE means APDU access is not allowed
APDUPermission ::= BOOLEAN

-- Each APDU filter is a 8 byte octet string:
-- 4-byte header and 4-byte mask
APDUFilters ::= SEQUENCE OF APDUFilter
APDUFilter ::= OCTET STRING (SIZE(8))

NFCAccessRule ::= CHOICE {
   nfcPermission [0] NFCPermission,
   -- RFU
   ...
}

-- TRUE means NFC event is allowed,
-- FALSE means NFC event is not allowed
NFCPermission ::= BOOLEAN
```

When retrieving rules from the Access Rule Files (ARF), the APDU permission or the NFC event policy may not be defined in an access rule because these policies are optional. If they are not defined, the ARA-M shall consider the following:

- If an APDU permission policy has to be checked for a specific AID or for other AIDs by the enforcer and there is no APDU policy specified in the ARF, then the ARA-M or the Access Control Enforcer shall interpret the missing policy as an APDU permission ALWAYS allowed policy.
- If an NFC event policy has to be checked for a specific AID or for other AIDs by the enforcer and there is no NFC policy specified in the ARF, then the ARA-M or the Access Control Enforcer shall interpret the missing policy as an NFC event ALWAYS allowed policy.

7.2 ASN.1 Definition

This section includes all ASN.1 types, values, and information object class definitions contained in this document, in the form of the ASN.1:

```
module PKCS-15
GP ACP { iso(1) member-body(2) country-USA(840) Global-
Platform(114283) device(200) seAccessControl(1)
accessControlMainFile(1) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

AID ::= OCTET STRING

CertHash ::= OCTET STRING (SIZE(20))

APDUPermission ::= BOOLEAN

NFCPermission ::= BOOLEAN

APDUFilter ::= OCTET STRING (SIZE(8))


AccessControlMainFile ::= SEQUENCE {
refreshTag OCTET STRING (SIZE(8)),
rulesFile Path
}

Target ::= CHOICE {
aid [0] EXPLICIT AID,
default [1] NULL,
others [2]NULL
}

Rule ::= SEQUENCE {
target Target,
conditionsFile Path
}

Condition ::= SEQUENCE {
      cert   CertHash OPTIONAL,
      accessRules [0]AccessRules OPTIONAL
}

AccessRules ::= SEQUENCE OF AccessRule

AccessRule ::=CHOICE {
apduAccessRule    [0]APDUAccessRule,
nfcAccessRule     [1]NFCAccessRule
}
```

```
APDUAccessRule ::= CHOICE {
      apduPermission [0] APDUPermission,
      apduFilter [1] APDUFilter
}


APDUFilters ::= SEQUENCE OF APDUFilter

NFCAccessRule ::= CHOICE {
      nfcPermission [0] NFCPermission
}

END
```

7.3 File System Validity

When the device is determining whether access control rules are available in the file system of the Secure Element, different cases may occur:

The File System is correctly provisioned when either of the following is true:

- There is a PKCS#15 application (selectable using the standard AID) in the Secure Element and the OID of the access control mechanism is specified in the DODF provisioning file.
- If the Secure Element is a UICC, a PKCS#15 application/file structure is referenced in the Secure Element EFdir and the OID of the access control mechanism is specified in the DODF provisioning file.

The File System has no access rules (as described in this specification) when any of the following is true:

- There is no PKCS#15 application (selectable using the standard AID) and if the Secure Element is a UICC, there is no EFdir file.
- There is no PKCS#15 application (selectable using the standard AID) and if the Secure Element is a UICC, no PKCS#15 application/file structure is referenced in its EFdir.
- There is a PKCS#15 provisioning in the Secure Element (either an application or a file structure) and the OID of the access control mechanism is not referenced in a valid DODF file.

All other cases shall be treated as parsing error cases, which should lead to denying access to the whole Secure Element.

Annex A Summary of Data Object Nesting

This annex summarizes the nesting of data objects in the GET DATA command and response, and the STORE DATA command and response. For detailed information about the messages and data objects, see Chapters 4 through 6.

Where a data object name is shown in bold blue letters, the full nesting follows.

Table A-1: Data Object Nesting in GET DATA Command

Command	P1 P2	Data	Value		
GET DATA	[All]	absent			
	[Specific]	**REF-DO**	AID-REF-DO | Hash-REF-DO		
				AID-REF-DO	AID
					Empty
				Hash-REF-DO	Hash
					Empty
	[Refresh tag]	absent			
	[Next]	absent			

Table A-2: Data Object Nesting in GET DATA Response

P1 P2 of GET DATA Command	GET DATA Response Includes			
[All]	**Response-ALL-AR-DO**			
		Empty		
		REF-AR-DO$_1$ | ... | REF-AR-DO$_x$, first bytes		
			REF-DO | AR-DO	
[Specific]	**Response-AR-DO**			
		Empty		
		AR-DO, first bytes		
			APDU-AR-DO	'00' (NEVER)
				'01' (ALWAYS)
				APDU filter 1 | ... | APDU filter n
				4-byte APDU filter header | 4-byte APDU filter mask
			NFC-AR-DO	'00' (NEVER)
				'01' (ALWAYS)
			APDU-AR-DO | NFC-AR-DO	
[Refresh tag]	**Response-RefreshTag-DO**			
		RefreshTag (8-byte random number)		
[Next]				
		next bytes of REF-AR-DO		
		next bytes of AR-DO		

Table A-3: Data Object Nesting in STORE DATA Command

Command	Data	Value
STORE DATA	**Command-Store-AR-DO**	REF-AR-DO
	Command-Delete-AR-DO	AID-REF-DO
		REF-DO
		REF-AR-DO
		Empty
	Command-UpdateRefreshTag-DO	Empty
	Command-Register-ClientAIDs-DO	AID-REF-DO$_1$ | … | AID-REF-DO$_x$, first bytes
	Command-Get-AR-DO	Empty
		AID-REF-DO
	Command-GetAll-AR-DO	Empty
	Command-Get-ClientAIDs-DO	Empty
	Command-GetNext-AR-DO	Empty

Table A-4: Data Object Nesting in STORE DATA Response

STORE DATA Command	STORE DATA Response Includes	
(Command-Get-AR-DO)	**Response-ALL-AR-DO**	
		Empty
		REF-AR-DO$_1$ | ... | REF-AR-DO$_x$, first bytes
(Command-GetAll-AR-DO)	Response-ALL-AR-DO	
(Command-Get-ClientAIDs-DO)	**Response-ARAC-AID-DO**	
		Empty
		AID-REF-DO$_1$ | ... | AID-REF-DO$_x$, first bytes
(Command-GetNext-AR-DO)		
		Next bytes of REF-AR-DO
		Next bytes of AID-REF-DO

Annex B Data Object Tags

This annex lists the tags assigned to all data objects defined in this specification.

Table B-1: Data Object Tags

Tag	Data Object	
'4F'	AID-REF-DO	Specific application or all SE applications not covered by a specific rule
'C0'	AID-REF-DO	Implicitly selected application (all channels)
'C1'	Hash-REF-DO	
'D0'	APDU-AR-DO	
'D1'	NFC-AR-DO	
'DF 20'	Response-RefreshTag-DO	
'E1'	REF-DO	
'E2'	REF-AR-DO	
'E3'	AR-DO	
'F0'	Command-Store-AR-DO	
'F1'	Command-Delete-AR-DO	
'F2'	Command-UpdateRefreshTag-AR-DO	
'F3'	Command-Get-AR-DO	
'F4'	Command-GetAll-AR-DO	
'F5'	Command-GetNext-AR-DO	
'F6'	Command-Get-ClientAIDs-DO	
'F7'	Command-Register-ClientAIDs-DO	
'FF 40'	Response-ALL-AR-DO	
'FF 50'	Response-AR-DO	
'FF 70'	Response-ARAC-AID-DO	

Annex C Example of Access Control Data

For the sake of simplicity, in the following examples, AIDs are all in the form of A0 00 00 01 51 xx, and the certificate hashes are fake values like 111…, 222…, 333…, etc.

C.1 First Example

In this example, the setup is:

AID1 = A0 00 00 01 51 01	→ access denied for all apps	→ conditions 1
AID2 = A0 00 00 01 51 02	→ access allowed for 1 app (hash1)	→ conditions 2
AID3 = A0 00 00 01 51 03	→ access allowed for 1 app (hash1)	→ conditions 2
Any other AIDs	→ access allowed for all apps	→ conditions 3

Here's a summary of the PKCS#15 file system personalization:

```
Hierarchical view (file system):

MF (3F00)
|-EF DIR (2F00)                --> reference DF PKCS-15
|
|-DF PKCS-15 (7F50)
  |-ODF (5031)                 --> reference DODF
  |-DODF (5207)                --> reference EF ACMain
  |-EF ACMain (4200)           --> reference EF ACRules
  |-EF ACRules (4300)          --> reference EF ACConditions...
  |-EF ACConditions1 (4310)
  |-EF ACConditions2 (4311)
  |-EF ACConditions3 (4312)


Hierarchical view (PKCS#15 application):

PKCS#15 application (AID: A0 00 00 00 63 50 4B 43 53 2D 31 35)
  |-ODF (5031)                 --> reference DODF
  |-DODF (5207)                --> reference EF ACMain
  |-EF ACMain (4200)           --> reference EF ACRules
  |-EF ACRules (4300)          --> reference EF ACConditions...
  |-EF ACConditions1 (4310)
  |-EF ACConditions2 (4311)
  |-EF ACConditions3 (4312)


EF DIR: 3F00/2F00

Based on this ASN.1 syntax:
DIRRecord ::= [APPLICATION 1] SEQUENCE {
   aid [APPLICATION 15] OCTET STRING,
```

```
    label [APPLICATION 16] UTF8String OPTIONAL,
    path [APPLICATION 17] OCTET STRING,
    ddo [APPLICATION 19] DDO OPTIONAL
}

aid PKCS-15 = A0 00 00 00 63 50 4B 43 53 2D 31 35
label = "PROVISIONING" = 50 52 4F 56 49 53 49 4F 4E 49 4E 47
path = 3F00/7F50

binary coding:
61 22 4F 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 50 0C 50 52 4F 56 49 53 49
4F 4E 49 4E 47 51 04 3F 00 7F 50


ODF: 3F00/7F50/5031

References file 5207.

Binary coding
A7 06 30 04 04 02 52 07


DODF: 3F00/7F50/5207

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840) Global-
Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}
http://www.oid-info.com/get/1.2.840.114283
==> HEX encoding = 2A 86 48 86 FC 6B 81 48 01 01

application name = "GP SE Acc Ctl" (example)
path to EF ACMain = 4200

binary coding:
A1 29 30 00 30 0F 0C 0D 47 50 20 53 45 20 41 63 63 20 43 74 6C A1 14 30 12
06 0A 2A 86 48 86 FC 6B 81 48 01 01 30 04 04 02 42 00


EF ACMain: 3F00/7F50/4200

Refresh tag value is 01 02 03 04 05 06 07 08
path to EF ACRules = 4300

binary coding:
30 10 04 08 01 02 03 04 05 06 07 08 30 04 04 02 43 00


EF ACRules: 3F00/7F50/4300

AID1 --> EFConditions 4310 --> access denied for all apps
AID2 --> EFConditions 4311 --> access allowed for 1 app (hash1)
AID3 --> EFConditions 4311 --> access allowed for 1 app (hash1)
*    --> EFConditions 4312 --> access allowed for all apps
```

```
binary coding:
30 10  A0 08 04 06 A0 00 00 01 51 01  30 04 04 02 43 10
30 10  A0 08 04 06 A0 00 00 01 51 02  30 04 04 02 43 11
30 10  A0 08 04 06 A0 00 00 01 51 03  30 04 04 02 43 11
30 08  82 00                          30 04 04 02 43 12
```

EF ACConditions1: 3F00/7F50/4310 (access denied for all apps)

```
binary coding:
(empty file)
```

EF ACConditions2: 3F00/7F50/4311 (access allowed for 1 app)

```
Hash1 has the value 111…

binary coding:
30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
```

EF ACConditions3: 3F00/7F50/4312 (access allowed for all apps)

```
binary coding:
30 00
```

C.2 Second Example

In this example, the setup is:

AID1 = A0 00 00 01 51 01	→ access allowed for all apps	→ conditions 1
AID2 = A0 00 00 01 51 02	→ access allowed for 1 app (hash1)	→ conditions 2
AID3 = A0 00 00 01 51 03	→ access allowed for 3 apps (h1, h2, h3)	→ conditions 3
AID4 = A0 00 00 01 51 04	→ access denied for all apps	→ conditions 4
AID5 = A0 00 00 01 51 05	→ access denied for all apps	→ conditions 4
Any other AIDs	→ access denied for all apps	→ conditions 4

Here's a summary of the PKCS#15 file system personalization:

```
Hierarchical view (file system):

MF (3F00)
|-EF DIR (2F00)                --> reference DF PKCS-15
|
|-DF PKCS-15 (7F50)
  |-ODF (5031)                 --> reference DODF
  |-DODF (5207)                --> reference EF ACMain
  |-EF ACMain (4200)           --> reference EF ACRules
  |-EF ACRules (4300)          --> reference EF ACConditions...
  |-EF ACConditions1 (4310)
  |-EF ACConditions2 (4311)
  |-EF ACConditions3 (4312)
  |-EF ACConditions4 (4313)


Hierarchical view (PKCS#15 application):

PKCS#15 application (AID: A0 00 00 00 63 50 4B 43 53 2D 31 35)
  |-ODF (5031)                 --> reference DODF
  |-DODF (5207)                --> reference EF ACMain
  |-EF ACMain (4200)           --> reference EF ACRules
  |-EF ACRules (4300)          --> reference EF ACConditions...
  |-EF ACConditions1 (4310)
  |-EF ACConditions2 (4311)
  |-EF ACConditions3 (4312)
  |-EF ACConditions4 (4313)


EF DIR: 3F00/2F00

Based on this ASN.1 syntax:
DIRRecord ::= [APPLICATION 1] SEQUENCE {
   aid [APPLICATION 15] OCTET STRING,
   label [APPLICATION 16] UTF8String OPTIONAL,
   path [APPLICATION 17] OCTET STRING,
```

```
   ddo [APPLICATION 19] DDO OPTIONAL
}

aid PKCS-15 = A0 00 00 00 63 50 4B 43 53 2D 31 35
label = "PROVISIONING" = 50 52 4F 56 49 53 49 4F 4E 49 4E 47
path = 3F00/7F50

binary coding:
61 22 4F 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 50 0C 50 52 4F 56 49 53 49
4F 4E 49 4E 47 51 04 3F 00 7F 50


ODF: 3F00/7F50/5031

References file 5207.

binary coding:
A7 06 30 04 04 02 52 07


DODF: 3F00/7F50/5207

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840) Global-
Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}
http://www.oid-info.com/get/1.2.840.114283
==> HEX encoding = 2A 86 48 86 FC 6B 81 48 01 01

application name = "GP SE Acc Ctl" (example)
path to EF ACMain = 4200

binary coding:
A1 29 30 00 30 0F 0C 0D 47 50 20 53 45 20 41 63 63 20 43 74 6C A1 14 30 12
06 0A 2A 86 48 86 FC 6B 81 48 01 01 30 04 04 02 42 00


EF ACMain: 3F00/7F50/4200

Refresh tag value is 01 02 03 04 05 06 07 08
path to EF ACRules = 4300

binary coding:
30 10 04 08 01 02 03 04 05 06 07 08 30 04 04 02 43 00


EF ACRules: 3F00/7F50/4300

AID1 --> EFConditions 4310 --> access allowed for all apps
AID2 --> EFConditions 4311 --> access allowed for 1 app (h1)
AID3 --> EFConditions 4312 --> access allowed for 3 apps (h1, h2, h3)
AID4 --> EFConditions 4313 --> access denied for all apps
AID5 --> EFConditions 4313 --> access denied for all apps
*    --> EFConditions 4313 --> access denied for all apps
```

```
binary coding:
30 10  A0 08 04 06 A0 00 00 01 51 01  30 04 04 02 43 10
30 10  A0 08 04 06 A0 00 00 01 51 02  30 04 04 02 43 11
30 10  A0 08 04 06 A0 00 00 01 51 03  30 04 04 02 43 12
30 10  A0 08 04 06 A0 00 00 01 51 04  30 04 04 02 43 13
30 10  A0 08 04 06 A0 00 00 01 51 05  30 04 04 02 43 13
30 08  82 00                          30 04 04 02 43 13


EF ACConditions: 3F00/7F50/4310 (access allowed for all apps)

binary coding:
30 00


EF ACConditions: 3F00/7F50/4311 (access allowed for 1 app)

binary coding:
30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11


EF ACConditions: 3F00/7F50/4312 (access allowed for 3 apps)

binary coding:
30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
30 16 04 14 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
30 16 04 14 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33


EF ACConditions: 3F00/7F50/4313 (access denied for all apps)

binary coding:
(empty file)
```

C.3 Third Example

In this example, the setup is:

Default AID	→ access allowed for 1 app (hash0)	→ conditions 1
AID1 = A0 00 00 01 51 01	→ access allowed for 1 app (hash1)	→ conditions 2
AID2 = A0 00 00 01 51 02	→ access allowed for 1 app (hash2) + APDU Filter	→ conditions 3
AID3 = A0 00 00 01 51 03	→ access allowed for all apps	→ conditions 4

Here's a summary of the PKCS#15 file system personalization:

```
Hierarchical view (file system):

MF (3F00)
|-EF DIR (2F00)                --> reference DF PKCS-15
|
|-DF PKCS-15 (7F50)
  |-ODF (5031)                 --> reference DODF
  |-DODF (5207)                --> reference EF ACMain
  |-EF ACMain (4200)           --> reference EF ACRules
  |-EF ACRules (4300)          --> reference EF ACConditions...
  |-EF ACConditions1 (4380)
  |-EF ACConditions2 (4381)
  |-EF ACConditions3 (4382)
  |-EF ACConditions4 (4383)


Hierarchical view (PKCS#15 application):

PKCS#15 application (AID: A0 00 00 00 63 50 4B 43 53 2D 31 35)
  |-ODF (5031)                 --> reference DODF
  |-DODF (5207)                --> reference EF ACMain
  |-EF ACMain (4200)           --> reference EF ACRules
  |-EF ACRules (4300)          --> reference EF ACConditions...
  |-EF ACConditions1 (4380)
  |-EF ACConditions2 (4381)
  |-EF ACConditions3 (4382)
  |-EF ACConditions4 (4383)


EF DIR: 3F00/2F00

Based on this ASN.1 syntax:
DIRRecord ::= [APPLICATION 1] SEQUENCE {
   aid [APPLICATION 15] OCTET STRING,
   label [APPLICATION 16] UTF8String OPTIONAL,
   path [APPLICATION 17] OCTET STRING,
   ddo [APPLICATION 19] DDO OPTIONAL
}
```

```
aid PKCS-15 = A0 00 00 00 63 50 4B 43 53 2D 31 35
label = "PROVISIONING" = 50 52 4F 56 49 53 49 4F 4E 49 4E 47
path = 3F00/7F50

binary coding:
61 22 4F 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 50 0C 50 52 4F 56 49 53 49
4F 4E 49 4E 47 51 04 3F 00 7F 50


ODF: 3F00/7F50/5031

References file 5207.

binary coding:
A7 06 30 04 04 02 52 07


DODF: 3F00/7F50/5207

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840) Global-
Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}
http://www.oid-info.com/get/1.2.840.114283
==> HEX encoding = 2A 86 48 86 FC 6B 81 48 01 01

application name = "GP SE Acc Ctl" (example)
path to EF ACMain = 4200

binary coding:
A1 29 30 00 30 0F 0C 0D 47 50 20 53 45 20 41 63 63 20 43 74 6C A1 14 30 12
06 0A 2A 86 48 86 FC 6B 81 48 01 01 30 04 04 02 42 00


EF ACMain: 3F00/7F50/4200

Refresh tag value is 01 02 03 04 05 06 07 08
path to EF ACRules = 4300

binary coding:
30 10 04 08 01 02 03 04 05 06 07 08 30 04 04 02 43 00


EF ACRules: 3F00/7F50/4300

Default AID --> EFConditions 4380 --> access allowed for 1 app (h0)
AID1        --> EFConditions 4381 --> access allowed for 1 app (h1)
AID2        --> EFConditions 4382 --> access allowed for 1 app (h2)...
AID3        --> EFConditions 4383 --> access allowed for all apps

binary coding:
30 08  81 00                          30 04 04 02 43 80
30 10  A0 08 04 06 A0 00 00 01 51 01  30 04 04 02 43 81
30 10  A0 08 04 06 A0 00 00 01 51 02  30 04 04 02 43 82
```

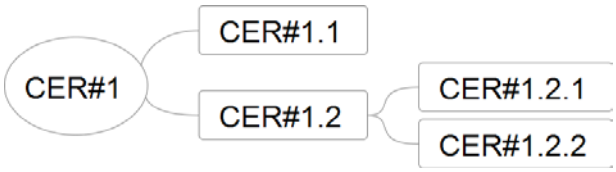
```
30 10  A0 08 04 06 A0 00 00 01 51 03  30 04 04 02 43 83


EF ACConditions: 3F00/7F50/4380 (access allowed for 1 app)

Hash0 has the value 000…

binary coding:
30 16 04 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00


EF ACConditions: 3F00/7F50/4381 (access allowed for 1 app)

Hash1 has the value 111…

binary coding:
30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11


EF ACConditions: 3F00/7F50/4382 (access allowed for 1 app)

Hash2 has the value 222…
APDU filter : 80 F2 00 00 / FF FF FF FF + 80 CA 00 00 / FF FF 00 00
NFC event : NEVER

binary coding:
30 35 04 14 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 A0
1D A0 16 A1 14 04 08 80 F2 00 00 FF FF FF FF 04 08 80 CA 00 00 FF FF 00 00
A1 03 80 01 00

EF ACConditions: 3F00/7F50/4383 (access allowed for all apps)

binary coding:
30 00
```

Annex D Rules Conflict Management Examples

Table D-1 shows how the Access Control enforcer shall apply combinations of the rules provided by the ARA-M.

Terms

Rxx	A certain rule defined in the Secure Element, regardless of where it is stored.
CER#y	A certain certificate.
CER#x.y	A certain certificate within a certificate chain



CER#y*	Used to specify a certain certificate or any child certificate thereof.
All	Indicates a general reference which applies to all device applications which are not covered by a specific reference.
Ref	The reference to an access rule (AR): Can be a specific reference CER (hash value of the device certificate (CER-Ref)) or the general reference “for all device applications” (ALL-Ref).
(Ref)-AR-APDU-ALWAYS	The access rule containing the “APDU always allowed” policy for a device application.
(Ref)-AR-APDU-NEVER	The access rule containing the “APDU never allowed” policy for a device application.
(Ref)-AR-APDU-Filter	The access rule containing the APDU filter for a device application.
(Ref)-AR-NFC-ALWAYS	The access rule containing the “NFC event always allowed” policy for a device application.
(Ref)-AR-NFC-NEVER	The access rule containing the “NFC event never allowed” policy for a device application.
NOT EXIST	Access rules for the SE application are not defined. This means the access rules contain no reference—neither “aid”, “default”, nor “others”—to the corresponding SE application. Thus, no access conditions are assigned to this SE application and all access is denied.

Conditions

- A specific rule has a higher priority than general rules. This means: if an access condition with a CER exists, then a further existing access condition which applies for all device applications (that is, a rule in which Hash-REF-DO is empty) shall be ignored.
- If the access rules contain neither access rules assigned to a certificate hash that matches to the device application’s certificate nor a rule that applies for all device applications, then the access shall be denied.

- Only the rule with the highest priority shall apply, based on the following priority orders:

  NEVER (APDU) > APDU filter > ALWAYS (APDU)

  NEVER (NFC) > ALWAYS (NFC)

- If aggregated access rules contain APDU filters, then these shall be combined per OR operation. This means an APDU is allowed if one of these filters matches:

  AR1-APDU filter || AR2-APDU filter || AR3-APDU filter || AR4-APDU filter

Table D-1: Rules Conflict Management

SE App.#1	SE App.#2	Other AIDs (general rule)	Access Result for the Secure Element Applications
R0: (CER#1-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with CER#1*are granted access to SE App. #1. • Access to any other SE application is denied.
R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (CER#2-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with CER#1* or with CER#2* are granted access to SE App. #1. • Access to any other SE application is denied.
R0: (CER#1-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Access to any SE application is denied.
R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (CER#1-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Access to SE App. #1 is denied because NEVER rule has higher priority. • Access to any other SE application is denied.
R0: (CER#1-Ref)-AR-APDU-Filter	NOT EXIST	NOT EXIST	• Device applications signed with CER#1* are granted access to SE App. #1, but only with an APDU matching the APDU filter. • Access to any other SE application is denied.
R0: (CER#2-Ref)-AR-APDU-NEVER	R0: (CER#2-Ref)-AR-APDU-Filter	R0: (CER#2-Ref)-AR-APDU-ALWAYS	• All device applications are denied access to SE App #1. • Device applications signed with CER#2* are granted access to SE App. #2, but only with an APDU matching the APDU filter. • Device applications signed with CER#2* are granted access to any other SE application.
R0: (CER#1-Ref)-AR-APDU-NEVER R1: (CER#2-Ref)-AR-APDU-Filter	NOT EXIST	NOT EXIST	• Device applications signed with CER#2* are granted access to SE App. #1, but only with an APDU matching the APDU filter. • Access to any other SE application is denied.

SE App.#1	SE App.#2	Other AIDs (general rule)	Access Result for the Secure Element Applications
R0: (CER#1.2-Ref)-AR-APDU-Filter R1: (CER#1.2.1-Ref)-AR-APDU-Filter	NOT EXIST	NOT EXIST	• Device applications signed with CER#1.2.1* are granted access to SE App. #1, but only with an APDU matching the R1 APDU filter. • Device applications signed with CER#1.2* are granted access to SE App. #1, but only with an APDU matching the R0 APDU filter. • Access to any other SE application is denied.
R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (ALL-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Device applications signed with CER#1* have an access to SE App. #1. • Access to any other SE application is denied
R0: (CER#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with CER#1* have no access to SE App. #1 • All other device applications don't have access to SE App. #1 because there is a specific rule protecting access to SE app #1
R0: (CER#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Access to all SE application is denied
R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (ALL-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with CER#1* have access to SE App. #1 • All other device applications don't have access to any SE application (including SE App#1)
NOT EXIST	NOT EXIST	R0: (ALL-Ref)-AR-APDU-ALWAYS	• All device applications have access to all SE applications
R0: (ALL-Ref)-AR-APDU-NEVER	R0: (ALL-Ref)-AR-APDU-NEVER	R0: (ALL-Ref)-AR-APDU-ALWAYS	• Access to the SE App. #1 and SE App. #2 is denied for all device applications • All other SE applications can be access by all device applications
R0: (ALL-Ref)-AR-APDU-ALWAYS	R0: (ALL-Ref)-AR-APDU-ALWAYS	R0: (ALL-Ref)-AR-APDU-NEVER	• Access to the SE App. #1 and SE App. #2 is granted for all device applications • All other SE applications cannot be accessed by any device application.

SE App.#1	SE App.#2	Other AIDs (general rule)	Access Result for the Secure Element Applications
R0: (CER#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-ALWAYS	R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (ALL-Ref)-AR-APDU-NEVER	R0: (CER#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-ALWAYS	• A device application signed with CER#1 or any other certificate has no access to SE App. #1. • A device application signed with CER#1 is allowed to access SE App. #2. • All other device applications have no access to SE App. #2 • Access to all other SE applications is denied for all device applications because there is a specific rule denying the access to all other SE applications.
R0: (CER#1-Ref)-AR-APDU-NEVER R1: (CER#2-Ref)-AR-APDU-NEVER R2: (CER#3-Ref)-AR-APDU-NEVER R3: (ALL-Ref)-AR-APDU-ALWAYS	R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (CER#2-Ref)-AR-APDU-NEVER R2: (CER#3-Ref)-AR-APDU-NEVER R3: (ALL-Ref)-AR-APDU-ALWAYS	R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (CER#2-Ref)-AR-APDU-ALWAYS R2: (CER#3-Ref)-AR-APDU-ALWAYS R3: (ALL-Ref)-AR-APDU-NEVER	• SE App. #1 can never be accessed • SE App. #2 can only be accessed by device applications signed with CER#1* • All other SE application can be access by all device applications which are signed with CER#1, CER#2, CER#3
R0: (CER#1-Ref)-AR-APDU-ALWAYS R1: (CER#2-Ref)-AR-APDU-ALWAYS R2: (CER#3-Ref)-AR-APDU-ALWAYS	R0: (CER#4-Ref)-AR-APDU-ALWAYS R1: (CER#5-Ref)-AR-APDU-ALWAYS R2: (CER#6-Ref)-AR-APDU-ALWAYS R3: (ALL-Ref)-AR-APDU-NEVER	R0: (CER#7-Ref)-AR-APDU-ALWAYS R1: (CER#8-Ref)-AR-APDU-ALWAYS R2: (CER#9-Ref)-AR-APDU-ALWAYS R3: (ALL-Ref)-AR-APDU-NEVER	• SE App. #1 can be access by all device applications which are signed with CER#1, CER#2, CER#3 • SE App. #2 can be access by all device applications which are signed with CER#4, CER#5, CER#6 • All other SE applications can only be accessed by device applications signed with CER#7, CER#8, CER#9
NFC Events Management			
R0: (CER#1-Ref)-AR-NFC-ALWAYS	R0: (CER#1-Ref)-AR-APDU-Filter	NOT EXIST	• Device applications signed with CER#1* can receive NFC events from SE App. #1. • Device applications signed with CER#1* are granted access to SE App. #2, but only with an APDU matching the APDU filter. • Device applications signed with CER#1 can receive NFC events from SE App. #2. • Access to any other SE application is denied.

SE App.#1	SE App.#2	Other AIDs (general rule)	Access Result for the Secure Element Applications
R0: (CER#1-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with CER#1* can receive NFC events from SE App. #1, because this application is allowed APDU access to SE App. #1. • Other device applications cannot receive NFC events.
R0: (CER#1-Ref)-AR-NFC-ALWAYS R1: (CER#1-Ref)-AR-NFC-NEVER	NOT EXIST	NOT EXIST	• NFC events cannot be received by any device application.
R0: (CER#1-Ref)-AR-NFC-NEVER R1: (CER#1-Ref)-AR-APDU-ALWAYS	NO EXIST	NOT EXIST	• NFC events cannot be received by any device application. • Device applications signed with CER#1* are granted APDU access to SE App. #1. • Access to any other SE application is denied.
R0: (CER#1-Ref)-AR-NFC-ALWAYS R1: (CER#1-Ref)-AR-APDU-ALWAYS	NO EXIST	NOT EXIST	• Device applications signed with CER#1* can receive NFC events. • Other device applications cannot receive NFC events. • Device applications signed with CER#1* are granted APDU access to SE App. #1. • Access to any other SE application is denied.
NO EXIST	NO EXIST	NO EXIST	• Access to any SE application is denied

Annex E APDU Process Flows

The following diagrams give an overview of the APDU flows.

E.1 Device Interface APDU Flow

Figure E-1 shows how the Access Control enforcer (Off-card Entity) shall retrieve all Access Rules via the ARA-M, as described in section 4.2.1.

Figure E-1: Access Control Rules Retrieval from ARA-M

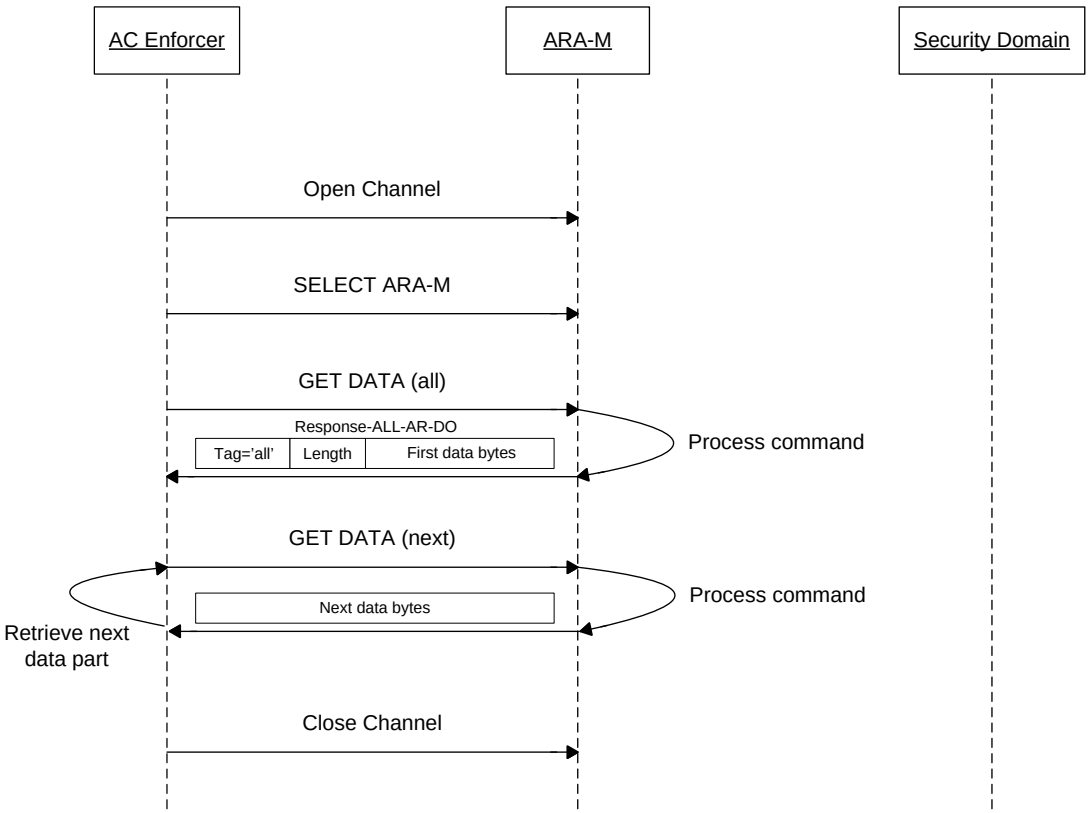


Figure E-2 shows how the Access Control enforcer (Off-card Entity) shall retrieve specific Access Rules from the ARA-M, as described in section 4.2.2. (See Figure E-3 for a more complete version of the same process.)

Figure E-2: Specific Access Rule Retrieval from ARA-M

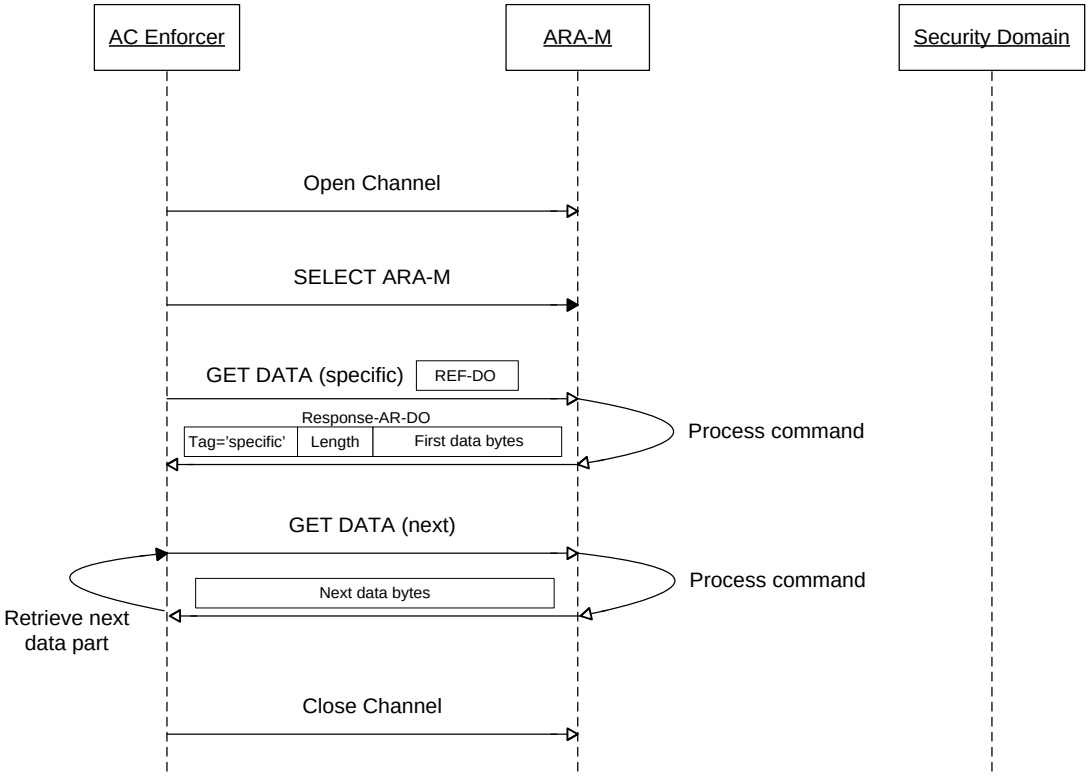


Figure E-3 shows how the Access Control enforcer (Off-card Entity) shall retrieve specific Access Rules from the ARA-M with the whole retrieval sequence, as described in section 4.2.2.

Figure E-3: Access Rule Retrieval Sequence

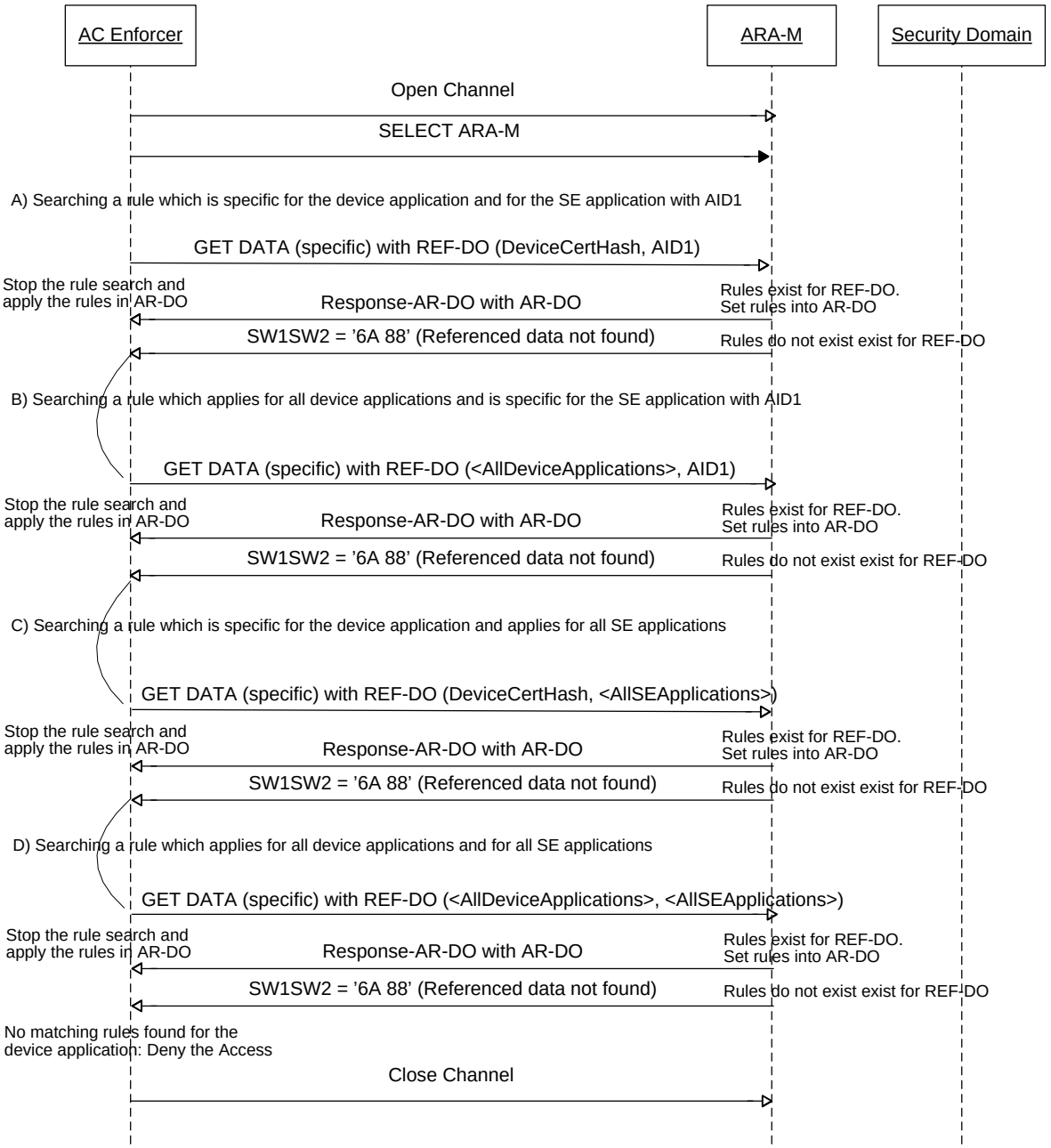
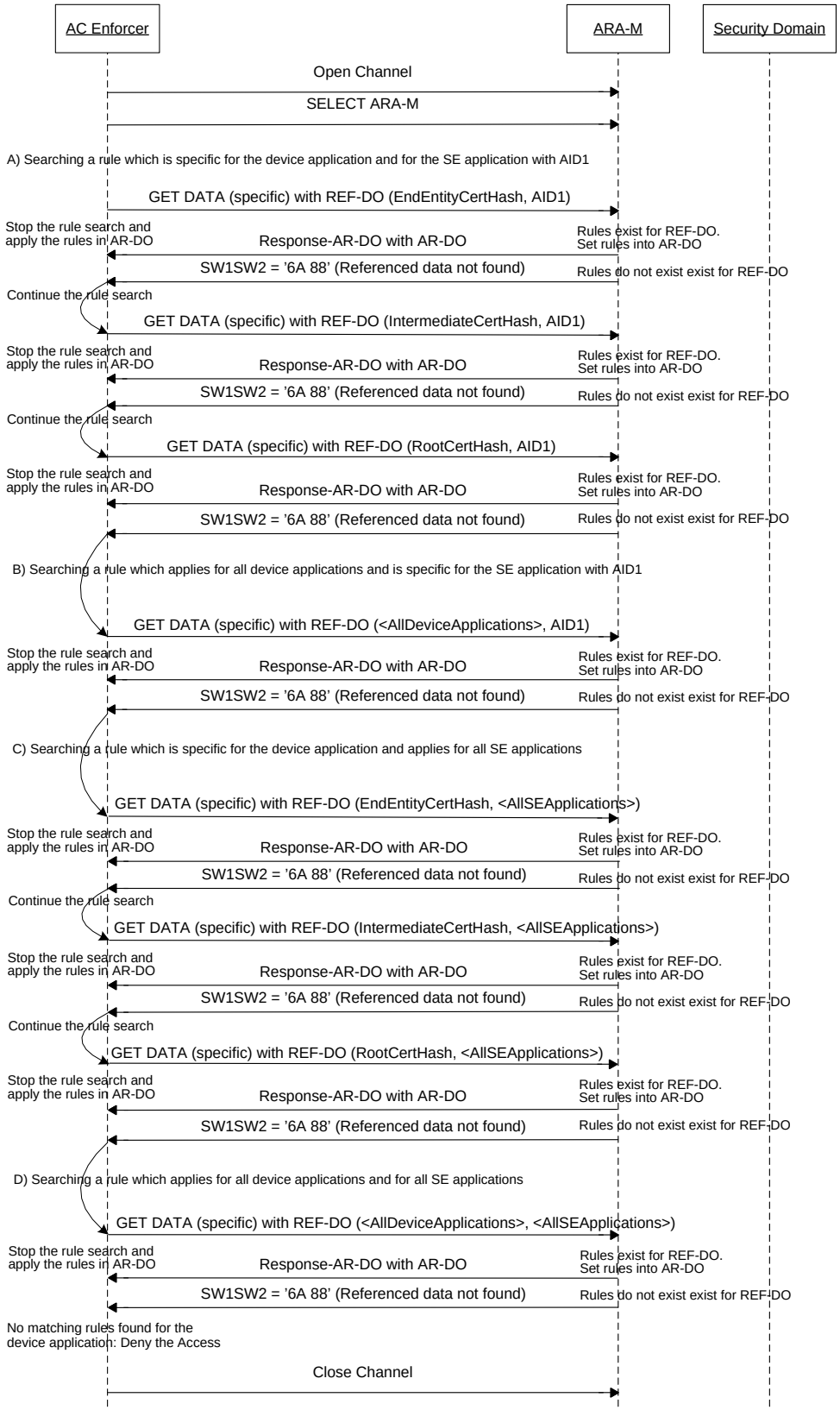


Figure E-4 shows how the Access Control enforcer (Off-card Entity) shall retrieve specific Access Rules from the ARA-M when the device application has a certificate chain.

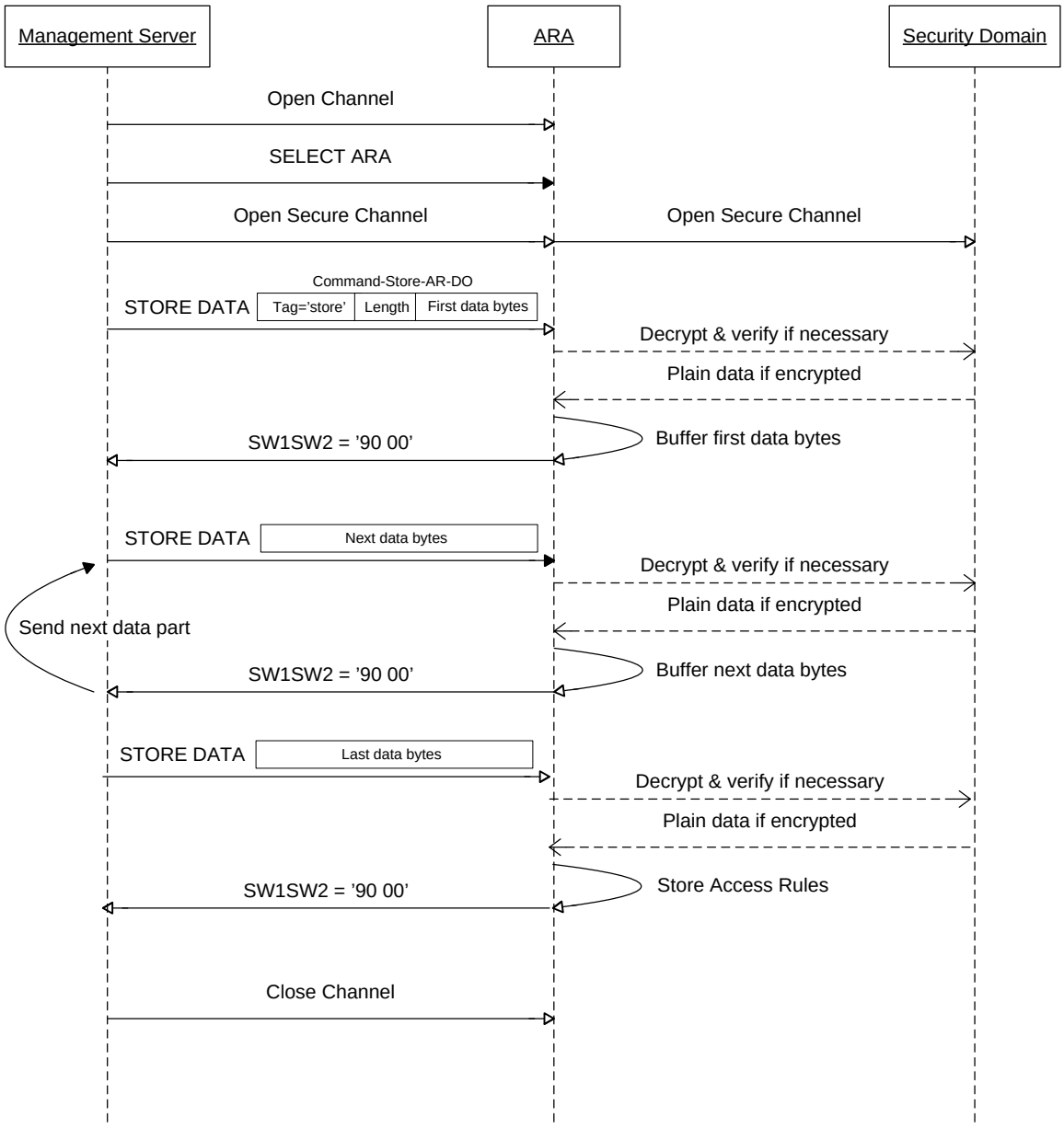
Figure E-4: Chained Certificate Querying



E.2 Remote Interface Based on RAM APDU Flow

Figure E-5 shows how an off-card entity (e.g. remote administration server) can store Access Rule data to the ARA (ARA-M or ARA-C) by sending a STORE DATA (Command-Store-AR-DO) command directly to the ARA.

Figure E-5: OTA Provisioning Directly to ARA



The deletion of Access Rule data can be performed in the same way with a STORE DATA (Command-Delete-AR-DO). Since the value of Command-Delete-AR-DO is always short, this command can always be transferred with one APDU.

Figure E-6: OTA Deletion Directly to ARA

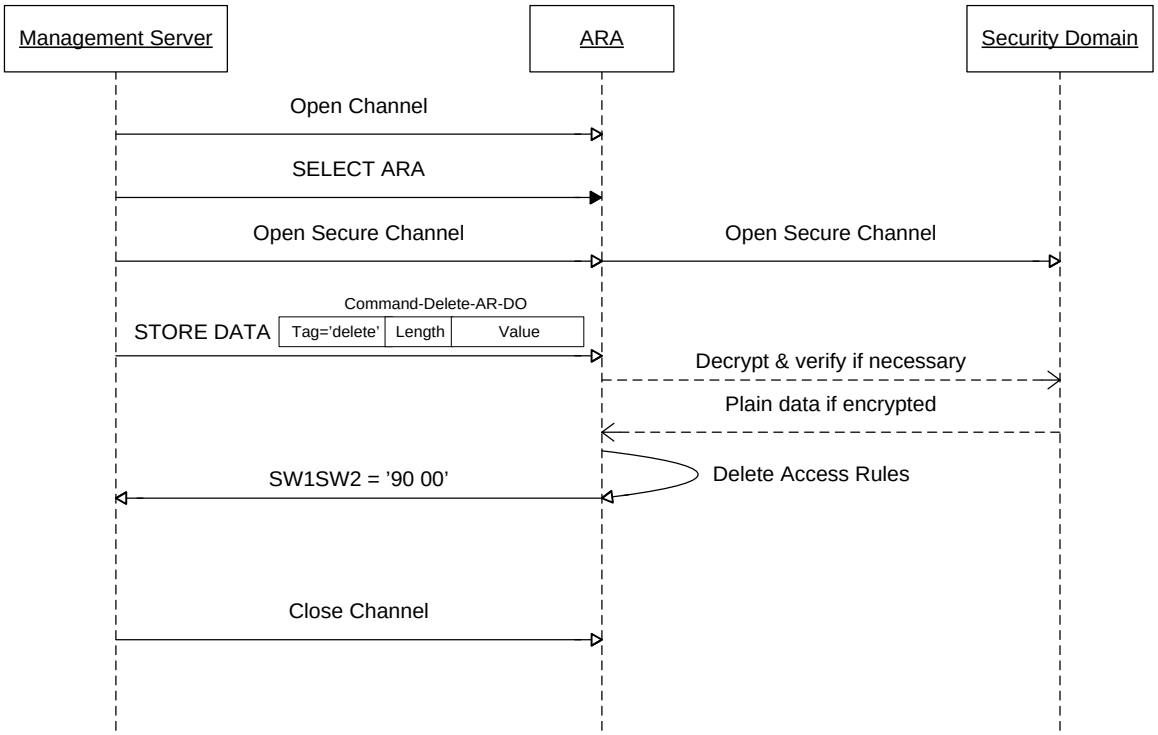
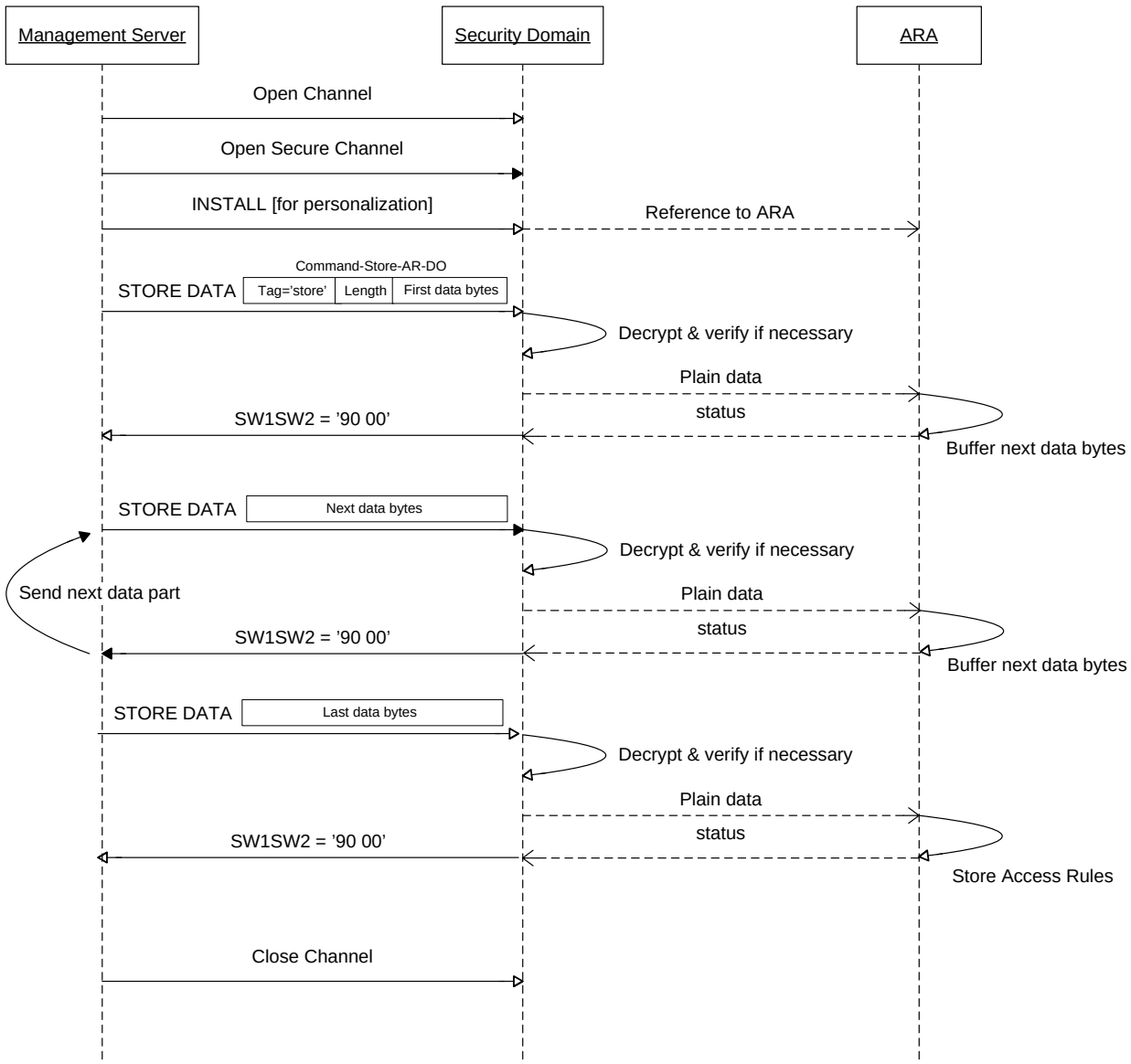


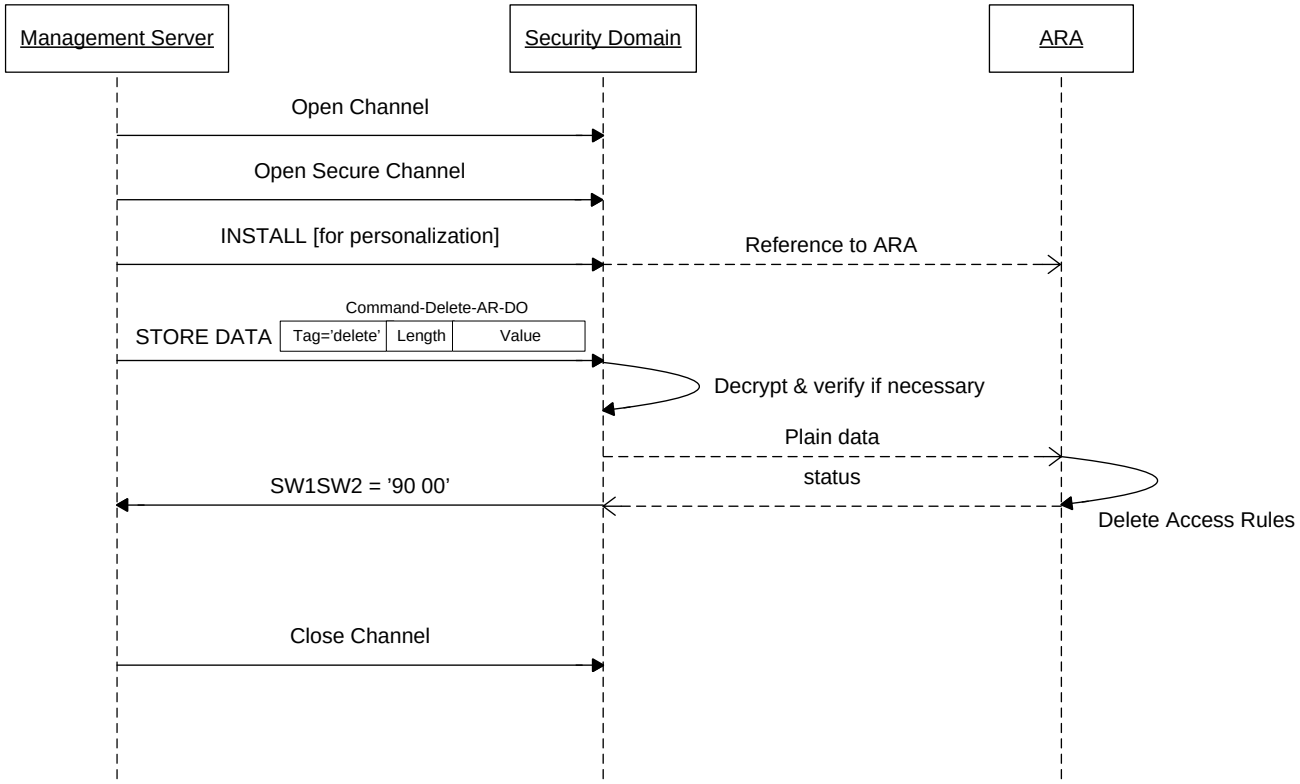
Figure E-7 shows how an off-card entity (e.g. remote administration server) can store Access Rule data to the ARA (ARA-M or ARA-C) by sending an INSTALL [for personalization] command and then a STORE DATA (Command-Store-AR-DO) command to the associated SD.

Figure E-7: OTA Provisioning Through Security Domain



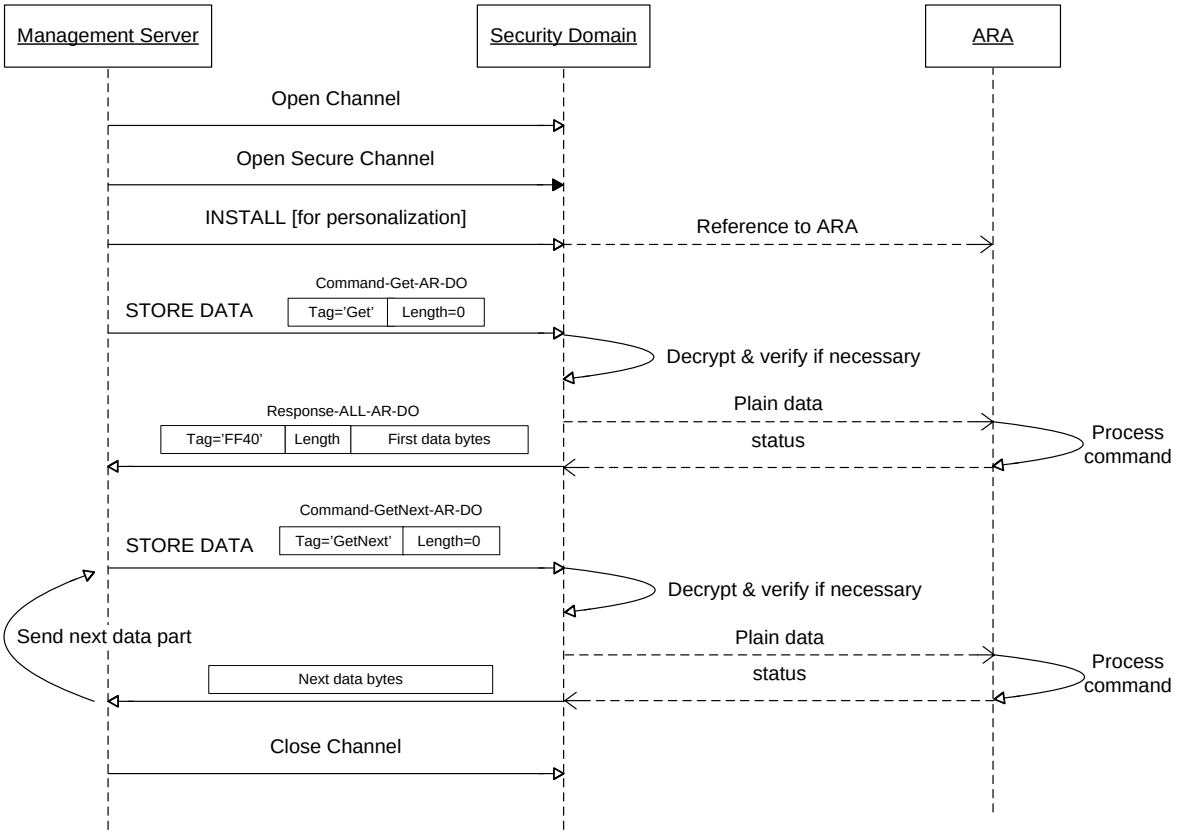
The deletion of Access Rule data can be performed in the same way with a STORE DATA (Command-Delete-AR-DO). Since the value of Command-Delete-AR-DO is always short this command can always be transferred with one APDU.

Figure E-8: OTA Deletion Through Security Domain



The retrieval of Access Rule data can be performed in the same way with a STORE DATA (Command-Get-AR-DO) or a STORE DATA (Command-GetAll-AR-DO). When the requested rules couldn't be fetched in response to this first command, a command STORE DATA (Command-GetNext-AR-DO) can be performed to retrieve the remaining data.

Figure E-9: OTA Rules Retrieval Through Security Domain



Annex F Migration Scenarios for the UICC

Prior to the publication of this specification, GSMA issued a document where access to applications on a UICC is defined by a set of elementary files [GSMA]. This solution is identical to ARF defined in Chapter 7. GSMA indicated that a long term solution would be based on this GlobalPlatform specification.

To guarantee compatibility with UICCs issued according to [GSMA] with ARF only, this specification makes it mandatory that the Access Control enforcer must support fallback to ARF for a UICC.

The following transition scenarios for UICCs can be expected:

- A UICC that is deployed with ACF only can be upgraded to the solution in this specification by loading and installing the ARA applets over-the-air.
- A UICC issuer may decide to install (at production or over-the-air) an ARA-M applet that is able to evaluate the ARF rules. This would allow a Network Operator that uses (only) RFM to manage its UICCs to continue to do so and to provide its rules to the ARF file system.
- Another UICC issuer may decide to install (at production or over-the-air) an ARA-M applet that is not able to evaluate the ARF rules, migrate all rules from the ARF to the ARA-M, and remove the ARF completely from the UICC.