Global Platform®

The standard for secure digital services and devices

GlobalPlatform Technology

Secure Element Access Control

Version 1.1.0.10

Public Review

August 2024

Document Reference: GPD_SPE_013

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Tables

Figures

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 [Open Mobile API] or [GP TEE SE API]), 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.

This specification considers two types of device applications: those that run in the Regular Execution Environment (REE, e.g. Android, Windows Phone environment) and those that run in the Trusted Execution Environment (TEE).

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.

In this specification, some elements are identified as "deprecated", and either have been replaced by new functions as noted in their descriptions or have been flagged for later removal. These elements will be removed from the specification in a future version. There is no guarantee that these elements will be present in a future release of this specification. Therefore, it is not recommended to include deprecated elements in new products or to reference them in new specification documents.

Note: In this version of the specification, access rules retrieval using GET DATA [Specific] is deprecated. GET DATA [Specific] is still allowed for ARA-M implementations that support older versions of Access Control Enforcers. It is recommended that Access Control Enforcer implementations use GET DATA [All] to retrieve access rules.

1.1 Audience

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

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of Intellectual Property Rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

The tables below list references applicable to this specification. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1-1: Normative References

Standard / Specification	Description	Ref
GPC_SPE_034	GlobalPlatform Technology Card Specification v2.3	[GP Card Spec]
GPD_SPE_009	GlobalPlatform Technology TEE System Architecture	[GP Sys Arch]
GPD_SPE_010	GlobalPlatform Technology TEE Internal Core API Specification	[GP Internal API]
ETSI TS 102 221	Smart cards; UICC – Terminal interface; Physical and logical characteristics, Release 6, 2004	[102 221]
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 102 622	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 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]

Table 1-2: Informative References

Standard / Specification	Description	Ref
GPC_SPE_007	GlobalPlatform Technology Confidential Card Content Management – Card Specification v2.3 – Amendment A	[GP Amd A]
GPC_SPE_011	GlobalPlatform Technology Remote Application Management over HTTP – Card Specification v2.3 – Amendment B	[GP Amd B]
GPC_SPE_025	GlobalPlatform Technology Contactless Services – Card Specification v2.3 – Amendment C	[GP Amd C]
GPD_SPE_024	GlobalPlatform Technology TEE Secure Element API	[GP TEE SE API]
GPD_SPE_007	GlobalPlatform Technology TEE Client API Specification	[GP TEE Client API]
GPD_SPE_008	GlobalPlatform Technology Secure Element Remote Application Management	[GP SE OTA]
GPD_SPE_075	GlobalPlatform Technology Open Mobile API Specification	[Open Mobile API]
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 3.0	[GSMA]
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 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]

Standard / Specification	Description	Ref
Java Card	Go to the following website for Java Card™ documentation:[1] http://www.oracle.com/technetwork/java/javacard/overview/index.html	[Java Card]
RFC 4122	A Universally Unique IDentifier (UUID) URN Namespace	[RFC 4122]

52

53 ## 1.4 Terminology and Definitions

54 **Table 1-3: Terminology and Definitions**

Term	Definition
Access Control Enforcer (ACE)	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 Applets.
Applet	General term for a Secure Element application: An application as described in GlobalPlatform Card Specification ([GP Card Spec]) that is installed in the SE and runs within the SE.
Application Provider	Entity responsible for the security of an application.
Device application	A third party application running on the open mobile OS.
Mobile device	Any device that includes a secure element, such as a mobile phone.
Regular Execution Environment (REE)	An Execution Environment comprising at least one Regular OS and all other components of the device (IC packages, other discrete components, firmware, and software) that execute, host, and support the Regular OSes (excluding any Secure Components included in the device). From the viewpoint of a Secure Component, everything in the REE is considered untrusted, though from the Regular OS point of view there may be internal trust structures. (Formerly referred to as a *Rich Execution Environment (REE)*.) Contrast *Trusted Execution Environment (TEE)*.
Regular OS	An OS executing in a Regular Execution Environment. May be anything from a large OS such as Linux down to a minimal set of statically linked libraries providing services such as a TCP/IP stack. (Formerly referred to as a Rich OS or Device OS.) Contrast Trusted OS.
Secure Element (SE)	A tamper-resistant secure hardware component that is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models. May exist in any form factor, such as embedded or integrated SE, SIM/UICC, smart card, smart microSD, etc.

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

Term	Definition
Secure Element Access API	An API used by device applications to exchange data with their counterpart applications running in the Secure Element.
Secure Element Issuer	Holds the ultimate responsibility for the GlobalPlatform card. Responsible for developing the card product profile, choosing the platform and application technologies, and designing the card layout. Usually holds a particular Security Domain in the SE: the Issuer Security Domain (ISD).
Security Domain (SD)	Application having the Security Domain privilege. This on-card entity provides support for the control, security, and communication requirements of an off-card entity such as the Card Issuer, an Application Provider, or a Controlling Authority.
Trusted Execution Environment (TEE)	An execution environment that runs alongside but isolated from Execution Environments outside of the TEE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets against a set of defined threats which include general software attacks as well as some hardware attacks, and defines rigid safeguards as to data and functions that a program can access. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. (For more information, see [GP Sys Arch].) Contrast *Regular Execution Environment*.
Universally Unique Identifier (UUID)	An identifier as specified in [RFC 4122].

1.5 Abbreviations and Notations

Table 1-4: Abbreviations

Abbreviation	Meaning
AC	Access Control
ACCF	Access Control Conditions File
ACE	Access Control Enforcer
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
APSD	Application Provider Security Domain
AR	Access Rule
ARA	Access Rule Application
ARA-C	Access Rule Application Client

Abbreviation	Meaning
ARA-M	Access Rule Application Master
AR-DO	Access Rule Data Object
ARF	Access Rule Files
ASN.1	Abstract Syntax Notation One, defined in [X.690]
BER	Basic Encoding Rules
DeviceAppID	Device Application Identifier
DF	Directory File
DO	Data Object (BER encoded TLV)
DODF	Data Object Directory File
EFdir	Application Directory Elementary File
ETSI	European Telecommunications Standards Institute
EVT	Event
GSM	Global System for Mobile Communication
GSMA	GSM Association
HCI	Host Controller Interface
ISD	Issuer Security Domain
ISO	International Organization for Standardization
MF	Master File
NFC	Near Field Communication
ODF	Object Directory File
OID	Object Identifier
oidDO	Object identifier for Data Object
OTA	Over The Air
PKCS	Public Key Cryptography Standards
RAM	Remote Application Management
REE	Regular Execution Environment
RFM	Remote File Management
RID	Registered Application Provider Identifier
SD	Security Domain
SE	Secure Element
SEAC	Secure Element Access Control
SHA-1	Secure Hash Algorithm 1
SHA-256	Secure Hash Algorithm 2 with 256-bit hash length
SW	Status Word

Abbreviation	Meaning
SWP	Single Wire Protocol
TA	Trusted Application
TEE	Trusted Execution Environment
TLV	Tag Length Value
TSM	Trusted Service Manager
UUID	Universally Unique IDentifier

1.6 Revision History

GlobalPlatform technical documents numbered *n*.0 are major releases. Those numbered *n*.1, *n*.2, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered *n.n*.1, *n.n*.2, etc., are maintenance releases that incorporate errata and clarifications; all non-trivial changes are indicated, often with revision marks.

Table 1-5: Revision History

Date	Version	Description
May 2012	1.0	Initial publication.
Sep 2014	1.1	Extension for applications in a Trusted Execution Environment. Extension for applications in Windows Phone 8 environment. New section 2.4, Rule Enforcement, to clarify the rule enforcement by the device. New section 3.1, Device Application Identifier (`DeviceAppID`). The new term `DeviceAppID` was introduced to generalize the mechanism of device identification and replaces the term “Hash” used in the former version of this document. New section 3.2, Specific Features for NFC Access Rule Enforcement, to clarify the basic concept of access control for NFC event. Refinements in sections 3.3 and 3.4 regarding access control rule combination and conflict resolution. Corrections in section 4.2.3, Algorithm for Applying Rules. New section 4.4, Management of the Version of the Device Interface, to manage backward compatibility of versions of this specification. Configuration management data objects were introduced: `Response-ARAM-Config-DO`, in Chapter 4, Device Interface. `Command-Get-Device-Config-DO` and `Response-Device-Config-DO`, in Chapter 5, Remote Interface Based on RAM. `Device-Config-DO`, `ARAM-Config-DO`, and `Device-Interface-Version-DO`, in new section 6.3, Configuration Management Data Objects. In Chapter 5, Remote Interface Based on RAM, clarifications on expected behavior on receipt of `Command-Store-REF-AR-DO` (formerly `Command-Store-AR-DO`). In section 4.1, GET DATA Command, explanation of concurrency of rule retrieval and rule storage. In Chapter 6, General Data Objects, clarification on how a BER-TLV with an unknown tag shall be handled by an ARA. In Chapter 6, `Block-DO` was introduced to perform rule storage or retrieval over several OTA sessions when using SCP80. In section 6.1, Access Rule Reference Data Objects, explanation of support of partial AIDs for SELECT [by name] [first occurrence] / [next occurrence] commands. *— continues —*

<table>
<tr><th>Date</th><th>Version</th><th>Description</th></tr>
<tr><td>Sep 2014</td><td>1.1 (continued)</td><td>In Chapter 7, Structure of Access Rule Files, clarification on the expected interpretation of ARF structures.
GET DATA [Specific] is deprecated in this version.
Changed the names of the following data objects:
<table>
<tr><th>Prior Name</th><th>Name as of v1.1</th></tr>
<tr><td><code>Response-ALL-AR-DO</code></td><td><code>Response-ALL-REF-AR-DO</code></td></tr>
<tr><td><code>Response-RefreshTag-DO</code></td><td><code>Response-Refresh-Tag-DO</code></td></tr>
<tr><td><code>Command-Store-AR-DO</code></td><td><code>Command-Store-REF-AR-DO</code></td></tr>
<tr><td><code>Command-Delete-AR-DO</code></td><td><code>Command-Delete</code></td></tr>
<tr><td><code>Command-UpdateRefreshTag-DO</code></td><td><code>Command-Update-Refresh-Tag-DO</code></td></tr>
<tr><td><code>Command-Register-ClientAIDs-DO</code></td><td><code>Command-Register-Client-AIDs-DO</code></td></tr>
<tr><td><code>Command-Get-AR-DO</code></td><td><code>Command-Get</code></td></tr>
<tr><td><code>Command-GetAll-AR-DO</code></td><td><code>Command-Get-All</code></td></tr>
<tr><td><code>Command-Get-ClientAIDs-DO</code></td><td><code>Command-Get-Client-AIDs-DO</code></td></tr>
<tr><td><code>Command-GetNext-AR-DO</code></td><td><code>Command-Get-Next</code></td></tr>
</table></td></tr>
<tr><td>Feb 2023</td><td>v1.1.0.3</td><td>Committee Review</td></tr>
<tr><td>Feb 2024</td><td>v1.1.0.5</td><td>Member Review</td></tr>
<tr><td>Aug 2024</td><td>v1.1.0.10</td><td>Public Review</td></tr>
<tr><td>TBD</td><td>1.2</td><td>Public Release
• Incorporated prior Errata and Precisions:
○ Added section 7.4, ARF Parsing, clarifying the exact behavior of the Access Control Enforcer if parsing errors occur.
○ In section 3.2, Specific Features for NFC Access Rule Enforcement, clarified the exact behavior of the Access Control Enforcer to retrieve updated NFC rules.
• Updated the reference to OMAPI to GPD_SPE_075.
• Changed the meaning of REE from Rich Execution Environment to Regular Execution Environment.
• Removed references to Windows Phone 8 environment.
• Added support for SHA-256 to derive <code>DeviceAppIDs</code> from the certificate of the device Application Provider (deprecated SHA-1).
• Added support for SHA-256 in ARF.
• Added a Fourth Example to Annex C where both DODF(1) and DODF(2) are present.
• Replaced “Secure Element application” with “Applet” in keeping with GlobalPlatform approved terminology.</td></tr>
</table>

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 Applets.

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

The following types of device applications are considered in this version of the specification:

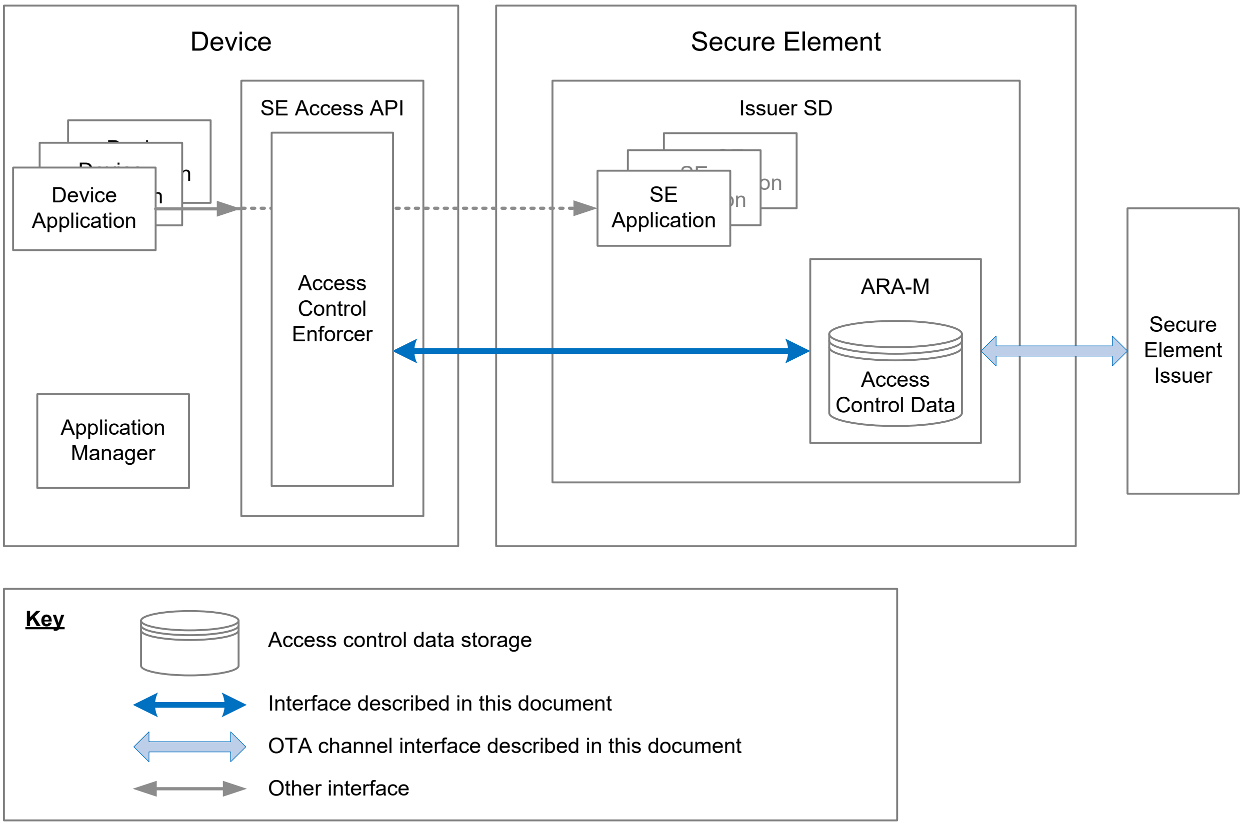
- Applications running in the REE (e.g. Android environments), called REE applications in this document
- Applications running in the TEE environment, called Trusted Applications (TAs)

This chapter illustrates several variations on the system architecture. The following topics are included:

2.1 Rules in Issuer Security Domain Only

In the most basic implementation of this specification, all Access Control rules are defined by the Secure Element 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 Secure Element Issuer defines access control rules for the Applets, and supplies those rules to the Access Rule Application Master (ARA-M). (The Secure Element Issuer may delegate administration to a Trusted Service Manager (TSM).)

When a device application attempts to access an Applet, the ACE 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 Applet which can be selected by a GlobalPlatform-defined AID, as follows:

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

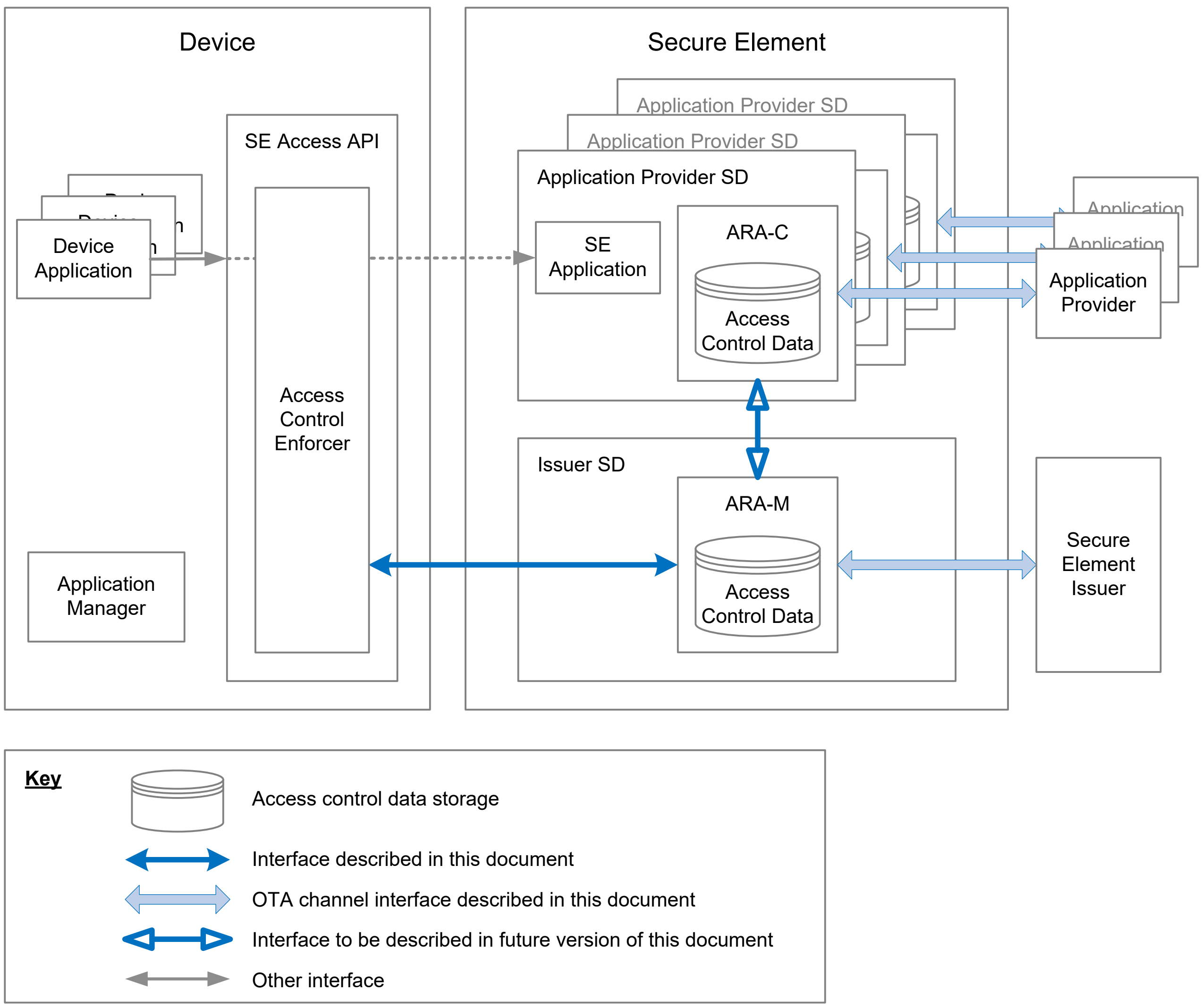
The ARA-M application is unique. Although access rule data can be stored in different locations within the SE (as discussed in the following sections), the ARA-M is in charge of retrieving all available access rules after a request from the ACE on the device.

The interface between the ACE and the ARA-M is described in Chapter 3. The interface between the Secure Element 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 Secure Element 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 Applet, the ACE 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 ACE may obtain the full set of rules in advance.) The ACE shall permit the access only if the rules indicate that it is acceptable.

The interface between the ACE 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 ([GP Card Spec]), 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-Client-AIDs-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:

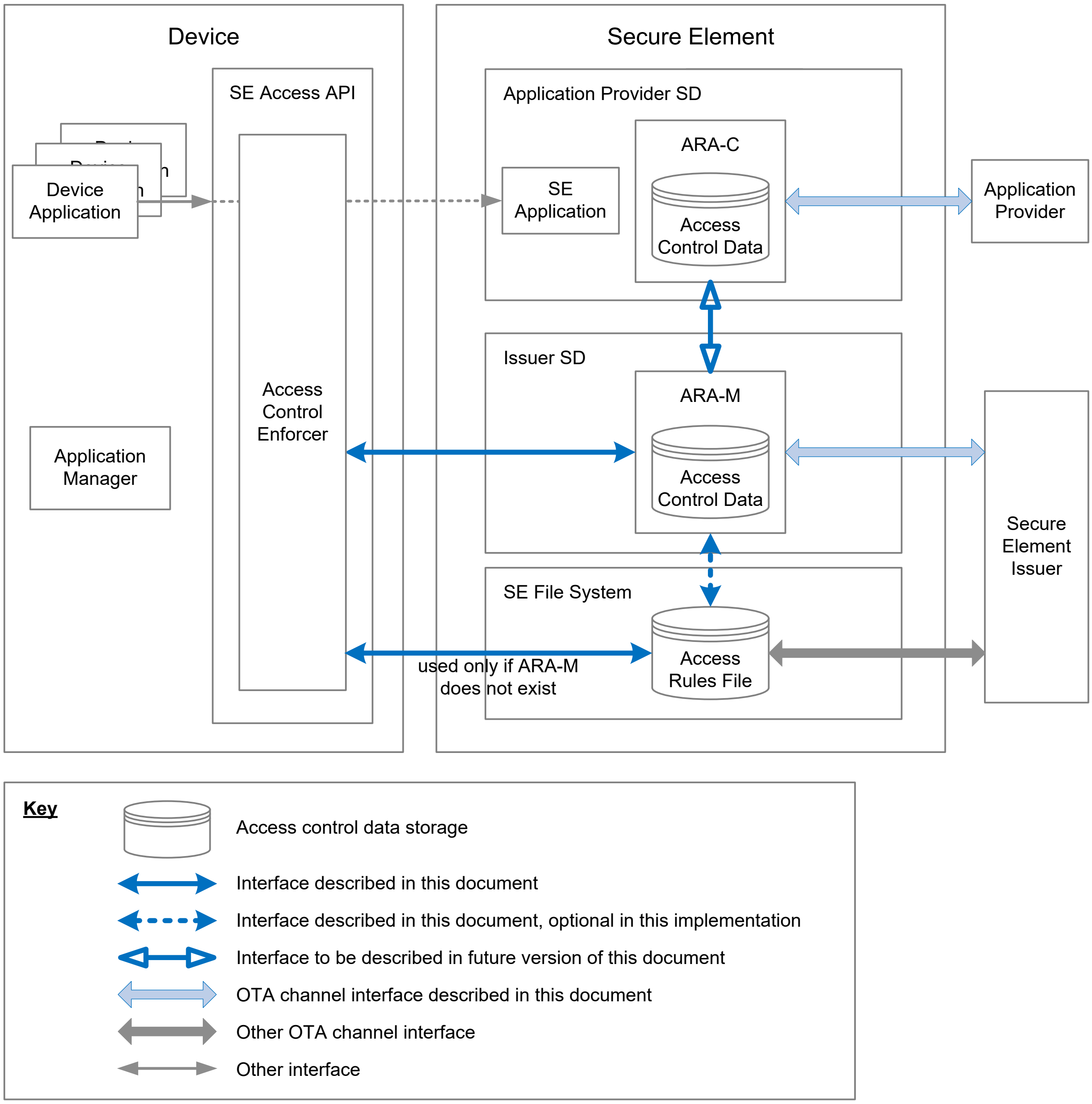
- Authentication of ARA-C by ARA-M
- Authorization to define access rights by ARA-C
- Architecture and interaction between the ARA-M and the ARA-C
- Internal API for registration of ARA-C to ARA-M

In this version of the specification, due to GlobalPlatform card API restrictions, it is not possible for an ARA-M/ARA-C to determine whether an access rule is associated with an Applet belonging to the same SD hierarchy.

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 Remote Application Management (RAM). This specification supports that mechanism as well, as illustrated in Figure 2-3.

Figure 2-3: Access Control Architecture with ARF Support

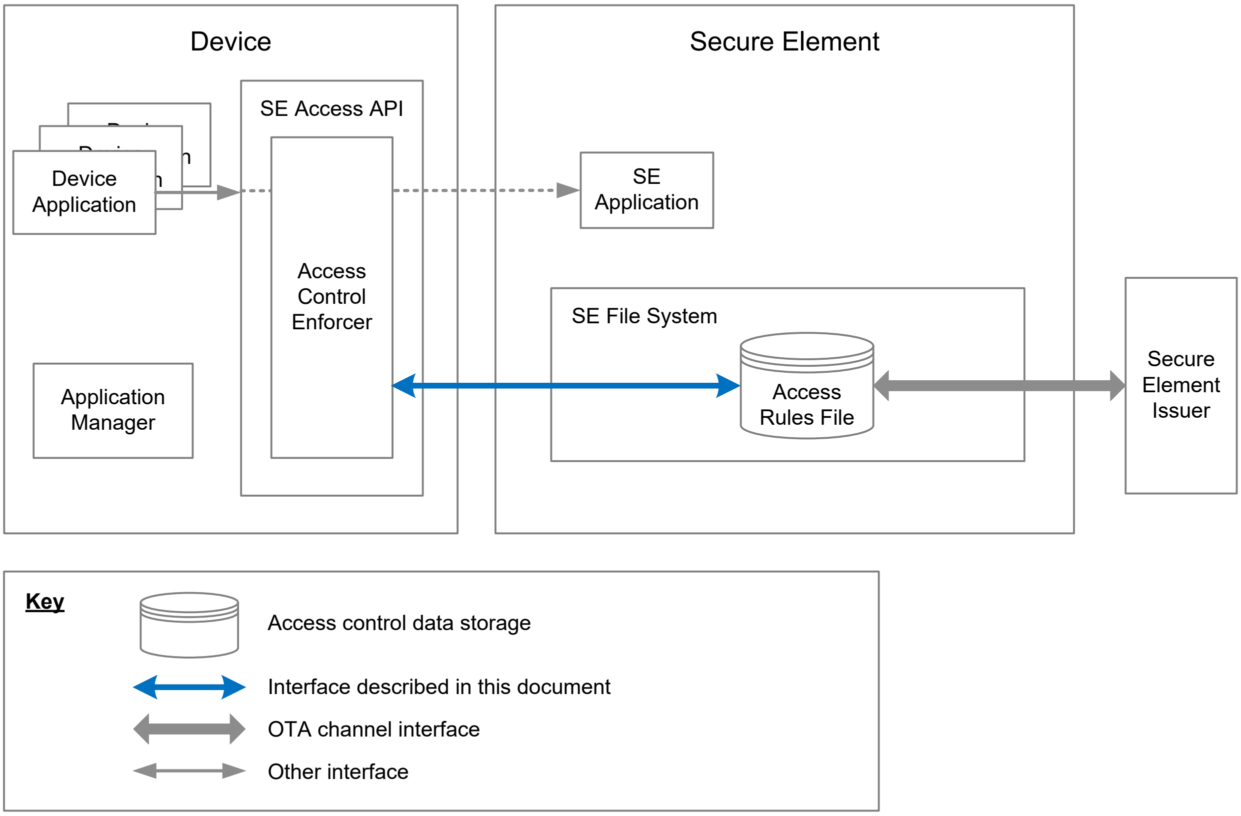


The Secure Element Issuer defines access control rules for the Applets, and supplies those rules to the ARA-M (as discussed in Chapter 4) or to the Access Rule File (ARF) (as discussed in Chapter 7).

When a device application attempts to access an Applet, the ACE 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 ACE may obtain the full set of rules in advance.) The ACE 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 ACE 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 Rule File System Fallback



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

2.4 Rule Enforcement

If an execution environment provides an API for device applications to interact with applications hosted by a Secure Element, this specification can be implemented to prevent unauthorized device applications gaining access to a specific Applet. The device API providing the access to the Secure Element can be the GlobalPlatform Open Mobile API ([Open Mobile API]) that is available in the REE or the GlobalPlatform TEE Secure Element API ([GP TEE SE API]) that is available in the Trusted Execution Environment (TEE). Such an API is called an "SE Access API" in this specification.

In order to be compliant with this specification, the SE Access API shall be connection oriented and shall implement an ACE as defined in this document. When a device application requests to open a connection with a given application in the Secure Element (usually identified by its AID), the SE Access API implementation shall invoke the ACE with the identifier of the device application requesting the connection and the identifier of the Applet to which connection is requested. Then, the ACE is in charge of retrieving the access rules applicable for the corresponding device application and Applet. If the access is granted, the SE Access API connection request shall be accepted and the device application shall be allowed to exchange commands (i.e. APDUs) with the Applet. If the access is not granted, the SE Access API connection request shall be rejected and the device application shall not be able to exchange commands (i.e. APDUs) with the Applet.

In this version of the specification, two types of device execution environment are considered:

- Device execution environments that support device applications signed with a key of the Application Provider. In this case, device applications are provided (i.e. in the application container) with a certificate of the Application Provider. This certificate is verified by the application installer of the Execution Environment. This certificate will be used by the ACE as the application identifier to retrieve the access rule that is applicable for this device application.
- Device execution environments in which device applications are uniquely identified with an application identifier which is included in the application and cannot be forged. This identifier will be used by the ACE to retrieve the access rule that is applicable for this device application. This is typically the case of the Trusted Execution Environment defined in the GlobalPlatform TEE System Architecture ([GP Sys Arch]), where Trusted Applications (TAs) are uniquely identified by their UUID (as defined in [RFC 4122]), as defined in the GlobalPlatform TEE Internal Core API ([GP Internal API]).

More details on device application identifiers are given in section 3.1.

The device may also host several REEs or several TEEs. This specification does not distinguish between multiple REEs or TEEs and a rule assigned to a specific application applies to this application no matter in which execution environment this application is installed.

This specification allows the definition of wildcard rules. Such a wildcard rule can be defined to allow a particular access for all device applications, whether installed in the REE or in the TEE.

3 Access Control Rules

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

- for a specific Applet, or for all other Applets 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 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 Device Application Identifier (DeviceAppID)

An ACE denies or allows device applications access to Applets based on access rules stored on the Secure Element. In order to identify the device applications and to associate device applications and access rules, the ACE uses device application identifiers (called `DeviceAppID` in this specification). The `DeviceAppID` identifies the application, depends on the device runtime, and is included in an access rule.

3.1.1 UUID as DeviceAppID

For Trusted Execution Environments (TEEs), the UUID of each device application is used as its `DeviceAppID`.

3.1.2 Certificate as DeviceAppID

For some REEs, such as Android, the SHA-256 (or deprecated SHA-1) hash value of the certificate of the device Application Provider is used as the `DeviceAppID`.

The following considerations apply when the hash value of a certificate is used as the `DeviceAppID`:

- An Application Provider certificate may be used to sign several applications. If so, a rule based on that certificate will apply to all those applications.
- For a REE application, several `DeviceAppIDs` can exist. This might occur if the REE application contains several signatures based on several certificates or if the REE application is bearing the whole certificate chain alongside the child certificate used to sign the application. In both cases the certificates are available in the application and each certificate has to be considered when determining the access rights of the application.
- For a REE application, `DeviceAppIDs` can exist for each of the supported hash algorithms (SHA-1 and SHA-256 in this version of the specification) to derive `DeviceAppIDs` from certificates.

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

This specification does not require the ACE to check the validity of any certificate. The developer of the ACE may choose to offer that functionality. It is assumed that the Regular OS providing the application certificate to the ACE can be trusted about the validity of the certificates and the corresponding signatures.

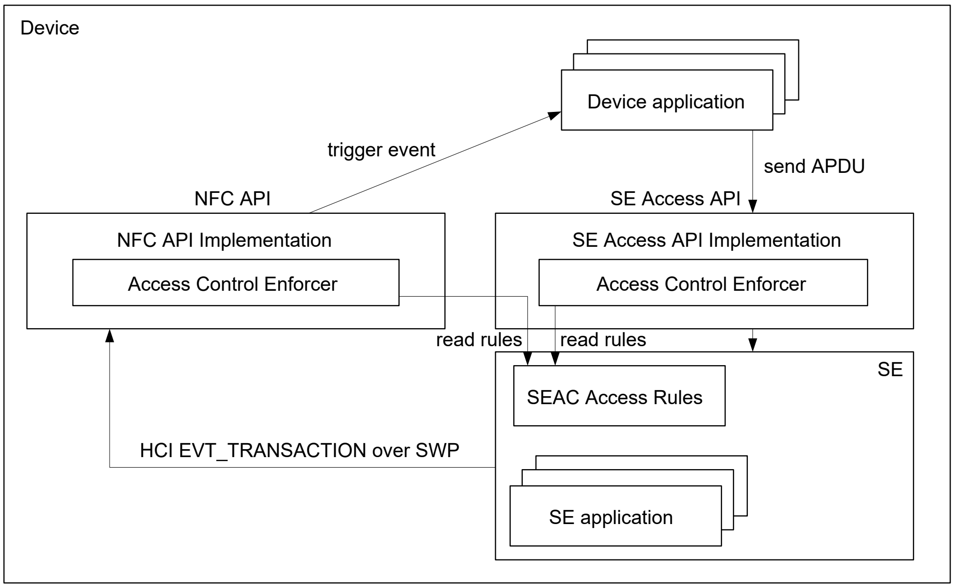
3.1.3 DeviceAppID Retrieval

The way the ACE retrieves the `DeviceAppID` (UUID or hash value of the certificate) is out of scope for this specification because it is specific to the runtime provided by the operating system. The ACE does not perform any check on the `DeviceAppID` provided by the operating system.

3.2 Specific Features for NFC Access Rule Enforcement

The NFC trigger event access rules are usually enforced by the NFC API implementation (also called NFC stack). This implies that the NFC API implementation contains an ACE. This ACE might work completely independently from the ACE in the SE Access API.

Figure 3-1: Example of Architecture for Access Rule Enforcement



The ACE requires capabilities from the NFC API to determine which Secure Element sent the HCI EVT_TRANSACTION. Then the ACE shall apply the NFC rules retrieved from this Secure Element. For the sake of efficiency, the NFC rules shall be cached in the device (the cache could be implemented in the NFC stack or somewhere else in the device). The ACE has to apply these rules following the process defined in section 4.2.1 with the following precisions:

- When an NFC Transaction Event targeting a Device application is received, the ACE shall fetch 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, before evaluating the rules, the ACE shall fetch a new copy of the whole rule set for the targeted Secure Element.

3.3 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 the case of certificate chains).**
3. **Restrictive rules have priority.**

3.3.1 Specific Rules Have Priority

A specific rule is a rule that associates:

- An Applet by specifying its AID or by specifying the implicitly selected application

AND

- A device application by specifying its `DeviceAppID`

All other rules are considered as generic rules. A generic rule is a rule that applies to:

- An Applet by specifying its AID or by specifying the implicitly selected application for device applications not explicitly specified (i.e. no `DeviceAppID` specified in the rule)

OR

- A device application by specifying its `DeviceAppID` for Applet not explicitly specified (i.e. no AID specified in the rule)

OR

- Undefined Applets with undefined device applications (i.e. this rule will be applied when neither the AID nor the `DeviceAppID` is present in any other rule)

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 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: Determining Priority of Rules

Applet explicitly referenced?	Device Application explicitly referenced?	Priority
yes	yes	highest (specific)
yes	no	high (generic)
no	yes	low (generic)
no	no	least (generic)

3.3.2 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.

3.3.3 Restrictive Rules Have Priority

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

3.4 Access Control Rule Combination and 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 ACE:

- If the ACE fetches all the rules from the Secure Element using GET DATA [All], then the ACE is responsible for the merging and conflict resolution, if any.
- If the ACE fetches the access rules for a particular access request using GET DATA [Specific] (deprecated), 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 generic device application identifier and a specific or generic Applet identifier). Then the ARA-M shall resolve the conflicts, if any, and return access rule data merged as described below.
- If the ACE uses certificate hash values as `DeviceAppIDs`, multiple `DeviceAppIDs` represent the same device application. This is because the application might be signed by several certificates, and because all supported hash algorithms (i.e. supported by ARA-M) shall be used to derive all possible `DeviceAppIDs` from the device application.

  When a set of rules exist for the same device application, i.e. when rules exists for `DeviceAppIDs` derived using the SHA-1 (deprecated) and SHA-256 hash algorithms, the ACE shall use the rules associated to the `DeviceAppIDs` that were derived with SHA-256. This is further detailed in section 4.2.3.

Rule conflicts might occur, especially if rules are stored in different locations (ARA-C, ARA-M, or ARF). Potential conflicts shall be avoided by the ARA implementation as much as possible:

Any ARA shall reject provisioning of a rule if a rule for the same target (AID and `DeviceAppID`) already exists in another ARA in the SE. The Service Provider shall be informed with the dedicated status word '6A 89'.

An ARA will not be able to detect conflicts with rules stored in the ARF.

However, in the following scenarios it might happen that rules applying to the same target (AID and `DeviceAppID`) exist in the SE:

- If the rules are pre-installed in an ARA and are available immediately after the ARA instantiation or registration to the ARA-M.

  Note: Pre-installation of access rules in an ARA is not specified in this specification and might not be implemented by ARAs.
- If the ARA-M reads the rules from the ARF.

  Note: The reading of access rules from an ARF by the ARA-M is an optional feature.
- If an ARA-C is locked and later unlocked again and the ARA-M or another ARA-C is updated in the meantime.

If the ARA-M/ARA-C implementation guarantees that only one rule exists in the SE for the same target (AID, `DeviceAppID`) then the ARA-M does not need to consider the following algorithm to solve conflicts or combine rules.

3.4.1 Algorithm to Solve Conflicts or Combine Rules

As described in section 3.3, specific rules have priority over generic rules. This strict priority shall be enforced by the ACE. Thus, the ACE shall first look for rules that apply to the targeted Applet and the targeted device application and shall look for generic rules only if no specific rule was found.[2]

If several rules apply to the same target (AID and `DeviceAppID`), then these rules are aggregated and more restrictive rules have priority over more permissive rules.

The following logic shall apply if several rules apply to the same target:

- If the access rules conflict, 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 the access rules are of different types (i.e. NFC permission, APDU permission), both rules are combined and thus both rules apply.
- If multiple 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 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 policies (never allowed, APDU filter, or always allowed) and NFC event policies (never allowed, always allowed) 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>Always</th><th>APDU filter</th><th>Never</th><th>Always</th><th>APDU filter</th><th>Never</th></tr>
<tr><td rowspan="6">R2</td><td rowspan="3">All</td><td>Always</td><td>R1=R2</td><td>R1</td><td>R1</td><td colspan="3" rowspan="3">R1</td></tr>
<tr><td>APDU filter</td><td>R2</td><td>R1+R2</td><td>R1</td></tr>
<tr><td>Never</td><td>R2</td><td>R2</td><td>R1=R2</td></tr>
<tr><td rowspan="3">AID</td><td>Always</td><td colspan="3" rowspan="3">R2</td><td>R1=R2</td><td>R1</td><td>R1</td></tr>
<tr><td>APDU filter</td><td>R2</td><td>R1+R2</td><td>R1</td></tr>
<tr><td>Never</td><td>R2</td><td>R2</td><td>R1=R2</td></tr>
</table>

The following logic shall apply if rules apply to a specific Applet:

- If only one rule exists which associates the `DeviceAppID` of a device application with the AID of a particular Applet, then access to that Applet from all other device applications is denied.
- If multiple specific rules exist, each of which associates the `DeviceAppID` of a different device application with the AID of the same Applet, then access to that Applet is denied for all device applications for which such a rule does not exist.

[2] The search for the right rules is described in detail in section 4.2.3 and section 4.3.

4 Device Interface

The ACE 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 ACE can request either a specific access rule (corresponding to a specific Applet and a specific device application and a specific `DeviceAppID`) [deprecated], 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 ACE shall implement the GET DATA command, as defined in this section, to manage the access rules.

For a UICC, the ACE 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 ACE is in charge of interpreting the fetched access rules correctly and filtering the access according to these rules.

The ARA-M shall support several logical channels for ACE(s) to retrieve rules. However, the number of supported logical channels might be limited. An ARA-M has to return SW '69 84' on GET DATA if all supported logical channels are already in use. In this case the ACE may retry later to select the ARA-M.

Secure Element is a UICC

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

- The ARA-M is not accessible (e.g. not installed, not selectable, or locked) and the ARF is not present.
- The ARA-M is accessible but does not provide an access rule explicitly granting access.
- The ARA-M is not accessible and the ARF is present but does not provide an access rule explicitly granting access.
- An error occurs during the reading and interpretation of the access rules (as discussed in section 7.3)

In other words, when the SE is a UICC, a device application can access an Applet in all of the following cases:

- The ARA-M is accessible and provides a rule which explicitly allows the access.
- The ARA-M is not accessible but the ARF contains a rule which allows the access.

Secure Element is not a UICC

Access to an SE which is not a UICC is denied in all of the following cases:

- The ARA-M is accessible but does not provide an access rule explicitly granting access.
- An error occurs during the reading and interpretation of the access rules.

In other words, when the SE is not a UICC, a device application can access an Applet in all of the following cases:

- The ARA-M is not accessible on the SE (e.g. not installed, not selectable, or locked).
- The ARA-M is accessible and provides a rule which explicitly allows the access.

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 ACE shall check whether a new version of the rules is available prior to applying cached rules.

Mode 3 (deprecated): Retrieve a specific access rule for a defined Applet (identified by the Applet's AID) and a device application (identified by the device application's `DeviceAppID`).

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 ACE 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 Applet (or all Applets 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 an Applet is implicitly authorized to receive NFC events from this application.

Concurrency of Rule Retrieval and Rule Storage

An ARA shall support concurrency of rule retrieval and rule storage. The following behavior shall apply:

- If a process to retrieve all access rules has been launched with a GET DATA [All] command and all the rules have not yet been retrieved using one or several GET DATA [Next] commands, but an access rule is updated, the GET DATA [Next] command will be rejected by the ARA-M with SW '69 84'. The ACE shall discard the access rules or part of the access rules already retrieved and shall restart the full rule retrieval procedure by sending a GET DATA [All] command.
- If a process to retrieve a specific access rule has been launched with a GET DATA [Specific] (deprecated) command and the whole rule has not yet been retrieved using one or several GET DATA [Next] commands, but an access rule associated with the same `REF-DO` (i.e. concerning the same AID and the same `DeviceAppID`) is updated, the GET DATA [Next] command will be rejected by the ARA-M with SW '69 84'. The ACE shall discard the part of the access rule already retrieved and shall restart the specific access rule retrieval procedure by a sending a GET DATA [Specific] (deprecated) command.

Note that an access rule is considered as updated only once the whole access rule (i.e. the whole `REF-AR-DO`) has been received. It may require several STORE DATA commands if the `REF-AR-DO` is too long to fit a single STORE DATA (`Command-Store-REF-AR-DO`) command.

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 [GP Card Spec]
INS	'CA'	GET DATA as specified in [GP Card Spec]
P1 P2	One of the following: 'FF 40': All 'FF 50': Specific (deprecated) 'DF 20': Refresh tag 'FF 60': Next 'DF 21': Config	All: Request to obtain all access rules. Specific: Request to obtain specific access rules. (deprecated) 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. Config: Sends the configuration of the ACE and requests the configuration of the ARA-M
Lc	Absent or Length of `REF-DO` or Length of `Device-Config-DO`	

Data	Absent
	or
	`REF-DO`
	or
	`Device-Config-DO`
		Absent:
		If P1 P2 = [All], [Next], or [Refresh tag]
		`REF-DO`:
		If P1 P2 = [Specific] (deprecated): `REF-DO` references a specific access rule.
		`Device-Config-DO`:
		If P1 P2 = [Config]: `Device-Config-DO` contains the ACE configuration.

Le	'00'	Expected length of the returned block

Note: Some CLA values might not be implemented by the ARA-M as no Secure Messaging is currently required between the ACE and the ARA-M.

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-REF-AR-DO`/`Response-AR-DO` shall always indicate the full length of all expected `AR-DO`s (even if not all `AR-DO` bytes are present in the GET DATA response field) so that the ACE 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-REF-AR-DO`.
2. GET DATA [Next]: Fetches the next (succeeding) bytes of the `Response-ALL-REF-AR-DO`.

or:

1. GET DATA [Specific] (deprecated): 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] (deprecated) 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 retrieval process did not start with a GET DATA [All] / GET DATA [Specific] (deprecated).

The GET DATA [Config] command shall be sent by the ACE after selection of the ARA-M (i.e. it shall be sent as the next command after the SELECT command); see section 4.4. A GET DATA [Config] command must be rejected by the ARA-M with SW '69 85' if not received just after the SELECT command.

4.1.1.2 Command Message Data Objects

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

If the GET DATA command includes P1 P2 = [Config], then a `Device-Config-DO` must be included in the Data field. This `Device-Config-DO` contains the configuration of the ACE, as 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-REF-AR-DO`, a `Response-AR-DO`, a `Response-Refresh-Tag-DO`, or a `Response-ARAM-Config-DO`:

- The `Response-ALL-REF-AR-DO` is mandatory for a GET DATA [All] request.
- The `Response-AR-DO` is mandatory for a GET DATA [Specific] (deprecated) request.
- The `Response-Refresh-Tag-DO` is mandatory for a GET DATA [Refresh tag] request.
- The `Response-ARAM-Config-DO` is mandatory for a GET DATA [Config] request.

Response-ALL-REF-AR-DO

Note: Formerly named `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-REF-AR-DO`. The length field of the `Response-ALL-REF-AR-DO` shall always contain the full length of the values of all the `REF-AR-DO`s. If the `Response-ALL-REF-AR-DO` is too large to fit in the GET DATA [All] response, then the remaining `Response-ALL-REF-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-REF-AR-DO

Tag	Length	Value	Meaning	Presence
'FF 40'	n or 0	REF-AR-DO$_1$ | … | REF-AR-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. 0 if there are no rules to fetch.	Mandatory

Response-AR-DO (deprecated)

If access rules exist in the Secure Element that corresponds to the `REF-DO` specified in the GET DATA [Specific] (deprecated) 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 data object’s value. If the `Response-AR-DO` is too large to fit in the GET DATA [Specific] (deprecated) 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`. 0 if there are no rules to fetch.	Mandatory

Response-Refresh-Tag-DO

Note: Formerly named `Response-RefreshTag-DO`.

The GET DATA [Refresh tag] command shall return a `Response-Refresh-Tag-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-Refresh-Tag-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

Response-ARAM-Config-DO

The GET DATA [Config] command shall return a `Response-ARAM-Config-DO` containing the configuration of the ARA-M.

If the GET DATA [Config] command returns an error SW, the ACE shall consider that the ARA-M implements version 1.0 of the specification. See section 4.4 for details on Device Interface version management.

Table 4-5: Response-ARAM-Config-DO

Tag	Length	Value	Meaning	Presence
'DF 21'	n	`ARAM-Config-DO`	**Value:** The `ARAM-Config-DO` containing the configuration of the ARA-M. **Length:** n is the full length of the `ARAM-Config-DO`.	Mandatory

4.1.2.2 Response Message Status Words

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

Table 4-6: GET DATA Response Message Status Words

SW1	SW2	Meaning
'65'	'81'	Memory problem
'67'	'00'	Wrong length in Lc
'69'	'84'	Either of the following: • Rules have been updated and must be read again to ensure consistency, or • All the supported logical channels are already in use. **Note:** In either case, the ACE shall read the rules by re-sending the command GET DATA [All] / GET DATA [Specific] (deprecated).
'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

Response to GET DATA [Specific] (deprecated) and GET DATA [All]

ARA-M implementations shall respond to the GET DATA [Specific] (deprecated) and GET DATA [All] commands as follows:

- If no access rule is available, return one of the following:
  - SW '6A 88', Referenced data not found (see Table 4-6)
  - SW '90 00' but a response data with a `Response-AR-DO` of null length ('FF5000') for a GET DATA [Specific] (deprecated)
  - SW '90 00' but a response data with a `Response-ALL-REF-AR-DO` of null length ('FF4000') respectively for a GET DATA [All]

Response to GET DATA [Next]

The ARA-M shall reject a GET DATA [Next] command with SW '69 85' if the data retrieval process did not start with a GET DATA [All] / GET DATA [Specific] (deprecated).

Response to GET DATA [Config]

The ARA-M shall reject a GET DATA [Config] command with SW '69 85' if this command is not the next command after the SELECT command.

Concurrency of Rule Retrieval and Rule Storage

An ARA shall manage the retrieval and storage of rules at the same time.

- If an access rule has been updated during the access rule retrieval process, then:
  - The ARA-M shall reject any GET DATA [Next] command with SW '69 84'.
  - The ACE shall discard the access rules or part of the access rules already retrieved and shall restart the full rule retrieval procedure by re-sending either a GET DATA [All] or a GET DATA [Specific] (deprecated) command.

See section 4.1 for detail on concurrency between rule retrieval and rule storage.

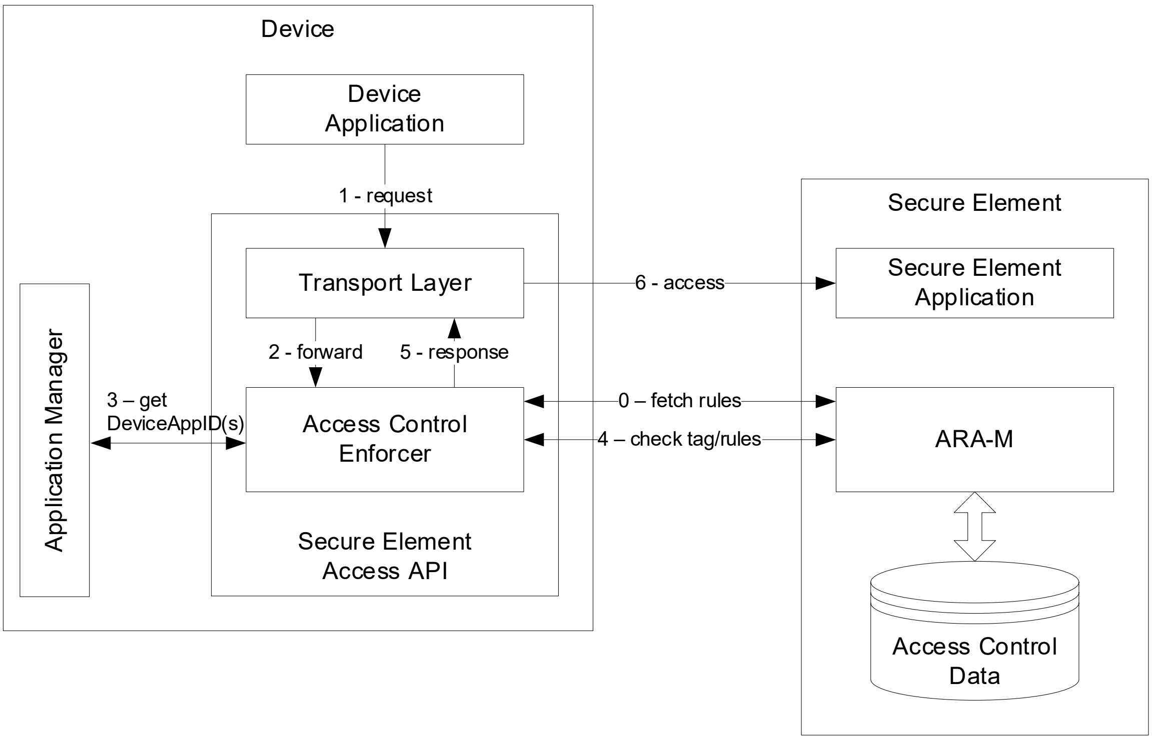
4.2 Access Control Evaluation Steps

This section describes how the ACE 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 Applet each time access is requested. The following sections explain each of those usage scenarios.

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 execution environment, the ACE 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 SE Access API to open a communication channel with an application residing in a Secure Element.

Step 2: This request is forwarded to the ACE on the device.

Step 3: The ACE retrieves the `DeviceAppID`(s) of the calling application. If the device application is running in the REE and has multiple signatures, the ACE retrieves each certificate of each signature. The ACE then computes the hashes (SHA-256 and SHA-1 (deprecated)) 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 ACE 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 the ACE fetches a new copy of the whole rule set for the targeted Secure Element.

Step 5: The ACE evaluates rules based on the calling application's `DeviceAppID(s)` 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 SE Access API performs all operations necessary to open the channel to the Secure Element (e.g. manage channel command and application selection).

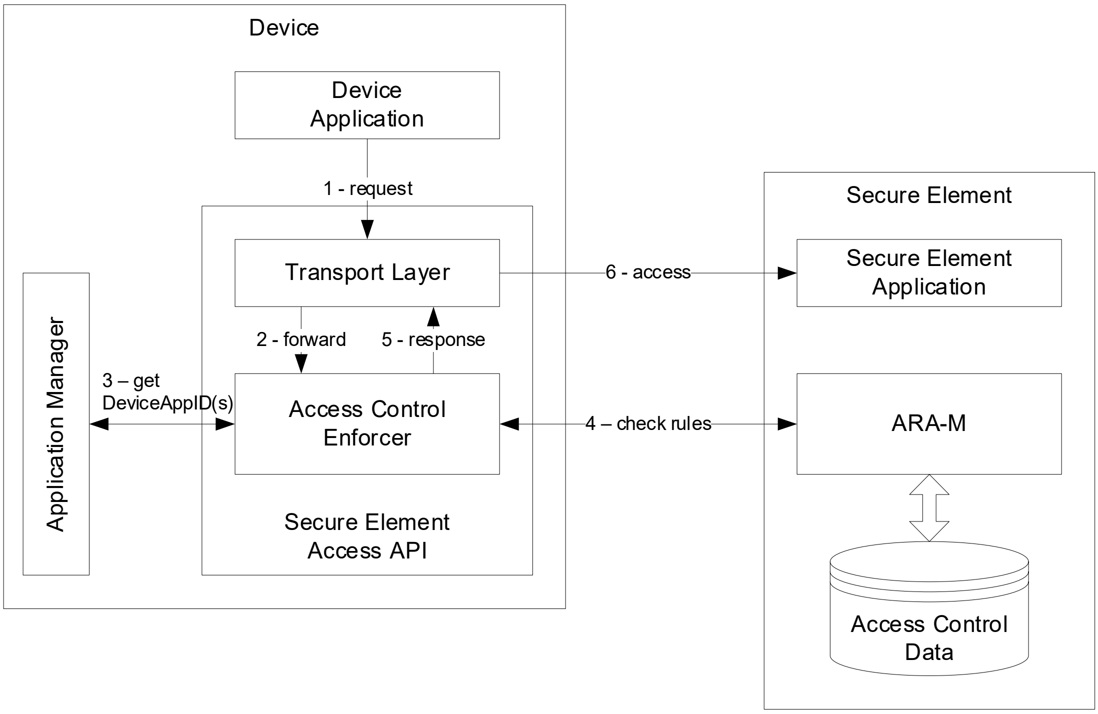
Step 6: Upon APDU transmission, the ACE applies the APDU filtering applicable to the connection (if any).

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 (Deprecated)

Note: This whole section shall be considered as deprecated in this version of the specification. Therefore, it is recommended that the ACE does not use the Instant Query mechanism in new products but instead uses the Rules Cached mechanism described in section 4.2.1.

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 SE Access API to open a communication channel with an application residing in a Secure Element.

Step 2: This request is forwarded to the ACE on the device.

Step 3: The ACE retrieves the `DeviceAppID(s)` of the calling device application. If the device application is running in the REE and has multiple signatures, the ACE retrieves each certificate of each signature. The ACE then computes the hashes (SHA-256 and SHA-1 (deprecated)) 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 ACE interrogates the ARA-M on the targeted Secure Element using the GET DATA [Specific] (deprecated). The ARA-M returns the requested rules, including APDU filter details, if any. See section 4.2.3.

Step 5: The ACE evaluates the rules to determine whether the application is allowed to access the Applet. 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 SE 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 ACE applies the APDU filtering applicable to the connection (if any).

4.2.3 Algorithm for Applying Rules

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

The ACE uses the following algorithm to retrieve an access rule for the device application (identified by one or more DeviceAppIDs) and the Applet (identified by its AID):

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

SearchRuleFor(DeviceAppID, AID)

When the hash value of a certificate is used as the DeviceAppID, this step shall be performed as follows:

- If the ARA-M supports both the SHA-256 and the SHA-1 (deprecated) hash algorithms, search for rules corresponding to the DeviceAppID derived using the SHA-256 algorithm.
  If no rule found, search for rules corresponding to the DeviceAppID derived using the deprecated SHA-1 algorithm.
- If the ARA-M does not support the SHA-256 algorithm, search for rules corresponding to the DeviceAppID derived using the deprecated SHA-1 algorithm.

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

B) If no rule fits condition A: Search for a generic rule that applies to all device applications and the Applet identified by AID:

SearchRuleFor(<AllDeviceApplications>, AID)

B-1) Search for any specific rule that associates another device application with the Applet identified by AID.

According to the rule conflict resolution process defined in section 3.4.1, if a specific rule exists that associates another device application with the Applet identified by AID (e.g. there is a rule associating AID with the DeviceAppID of another device application), then the ARA-M (when using GET DATA [Specific] (deprecated)) or the ACE (when using GET DATA [All]) shall set the result of SearchRuleFor(<AllDeviceApplications>, AID) to NEVER (i.e. precedence of specific rules over generic rules) and stop the rule search.

B-2) Search for any generic rule that associates all device applications with the Applet identified by 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 generic rule that specifies the device application and that applies to all Applets:

SearchRuleFor(DeviceAppID, <AllSEApplications>)

When the hash value of a certificate is used as the DeviceAppID, this step shall be performed as follows:

- If the ARA-M supports both the SHA-256 and the SHA-1 (deprecated) hash algorithms, search for rules corresponding to the DeviceAppID derived using the SHA-256 algorithm.
  If no rule found, search for rules corresponding to the DeviceAppID derived using the deprecated SHA-1 algorithm.
- If the ARA-M does not support the SHA-256 algorithm, search for rules corresponding to the DeviceAppID derived using the deprecated SHA-1 algorithm.

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

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

```
SearchRuleFor(<AllDeviceApplications>, <AllSEApplications>)
```

D-1) Search for any generic rule that associates another device application with all Applets.

According to the rule conflict resolution process defined in section 3.4.1, if a rule exists that associates another device application and applies to all Applets (e.g. there is a rule associating all Applets with the DeviceAppID of another device application), then the ARA-M (when using GET DATA [Specific] (deprecated)) or the ACE (when using GET DATA [All]) shall set the result of SearchRuleFor(<AllDeviceApplications>, <AllSEApplications>) to NEVER and stop the rule search.

D-2) Search for any generic rule that associates all device applications with all Applets.

If a rule exists, then apply this rule.

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

4.3 Managing Certificate Chains

Note: This section applies only to the REE environment, where several certificates can be associated with a REE application. It is not applicable to the TEE environment where the UUID is used to identify the Trusted Application (TA).

As discussed in section 3.3.2, 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.

A chain of certificates is typically embedded in the installation package (called application container in some REE systems) of the device application by the Application Provider.

Figure 4-3: Processing of Chained Certificates

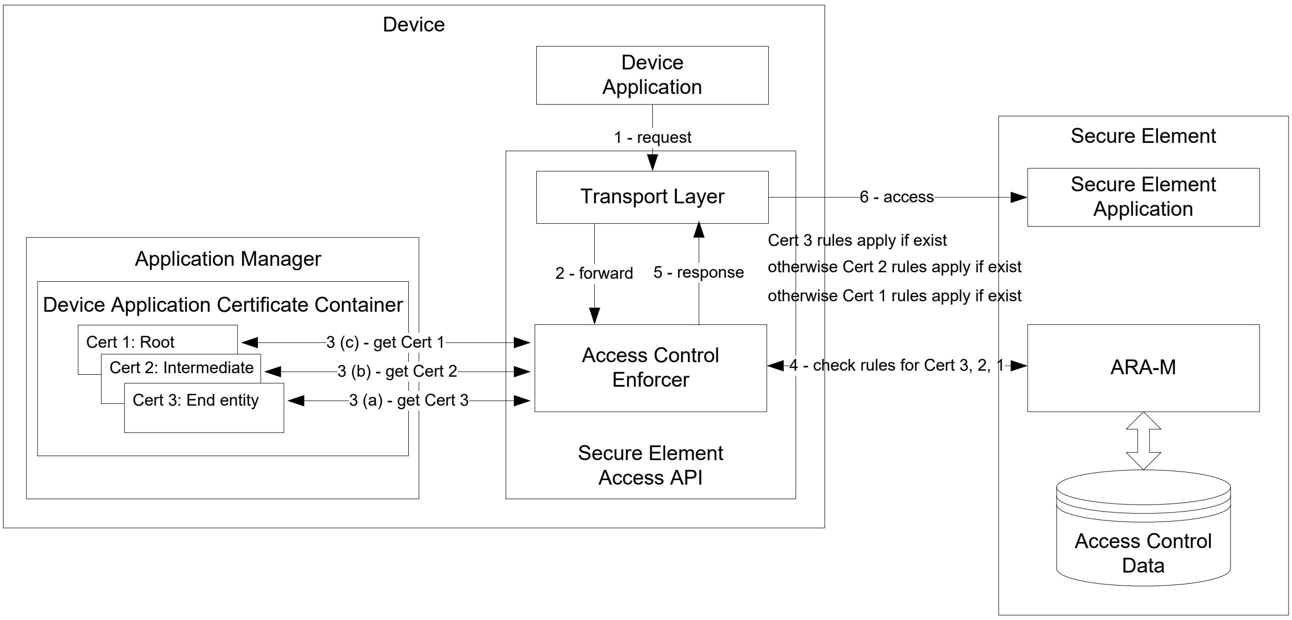


Figure 4-3 shows how the ACE queries the rules from the ARA-M when a device application is signed by a certificate within a chain.

In the case of certificate chains, the ACE 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 ACE may have to issue several GET DATA [Specific] (deprecated) 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 ACE 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 Applet 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 generic rule that applies to all device applications and the Applet identified by 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 generic rule specifies the device application and that applies to all Applets:

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 generic rule that applies to all device applications and to all Applets:

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

   If a rule exists, then apply this rule.

4.4 Managing the Version of the Device Interface

The ACE shall send a GET DATA [Config] to the ARA-M after the selection of the ARA-M (i.e. GET DATA [Config] command shall be sent right after the SELECT command selecting the ARA-M) in order to inform the ARA-M of the version of the Device Interface implemented by the ACE. In response to this command the ARA-M will send back the version of the Device Interface it implements.

4.4.1 Managing Backward Compatibility with Previous Versions

ACEs and ARA-Ms deployed in the field might implement different versions of this specification. This section describes how ACE or ARA-M that follow this current version of the specification shall identify ACE and ARA-M following previous versions of this specification.

4.4.1.1 Identifying ARA-M Version

The GET DATA [Config] command was not defined in version 1.0 of the specification. An ACE that implements this version of the specification can use that fact to manage interoperability with an ARA-M implementing a former version of the Device Interface:

- If the ARA-M rejects the GET DATA [Config] command with an error SW (for example SW '6A 86', Incorrect P1 P2), the ACE shall consider that the ARA-M implements the Device Interface as defined in version 1.0 of this specification.
- If the ARA M responds to the GET DATA [Config] command with specific version information, the ACE shall consider that the ARA M implements the Device Interface version reported in ARAM-Config-DO. If the reported version is 1.2.0, the ARA-M supports the current version of the specification; otherwise it supports version 1.1 of this specification.

4.4.1.2 Identifying Access Control Enforcer Version

The GET DATA [Config] command was not defined in version 1.0 of the specification. An ARA-M that implements this version of the specification can use that fact to manage interoperability with an ACE implementing a former version of the Device Interface.

- If the ARA-M does not receive a GET DATA [Config] command right after the ARA-M selection (i.e. SELECT command), the ARA-M shall consider that the ACE implements version 1.0 of the Device Interface.
- If the ARA-M receives a GET DATA [Config] command right after the ARA-M selection (i.e. SELECT command), the ARA-M shall consider the version reported in Device-Config-DO. If the reported version is 1.2.0, the ACE supports the current version of the specification; otherwise it supports version 1.1 of this specification.

4.4.1.3 Managing ARA-M Backward Compatibility

When the ACE following this version of the specification has identified an ARA-M following version 1.0 or 1.1 of this specification, it should not search for rules for `DeviceAppIDs` derived via SHA-256 from the device application certificate. This means that the ACE should not attempt to provide SHA-256 hashes for `DeviceAppIDs` in `DeviceAppID-REF-DO` when sending GET DATA [Specific] (deprecated) commands.

4.4.1.4 Managing ACE Backward Compatibility

When an ARA-M following this version of the specification has identified an ACE following version 1.0 of this specification:

- In the case of concurrency of rule retrieval and rule storage, the ARA-M shall send the error SW '69 85' rather than the error SE '69 84' defined in section 4.1.
- The ARA-M should not return ARA rules containing SHA-256 `DeviceAppIDs` on receipt of a GET DATA [All] command, or a GET DATA [Specific] (deprecated) command with an empty `DeviceAppID-REF-DO`.

When an ARA-M following this version of the specification has identified an ACE following version 1.1 of this specification:

- The ARA-M should not return ARA rules containing SHA-256 `DeviceAppIDs` on receipt of a GET DATA [All] command, or a GET DATA [Specific] (deprecated) command with an empty `DeviceAppID-REF-DO`.

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 and the ARA-C each provide 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 [GP Card Spec].

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 ARA must keep the previous state until a successful update is completed.

Access rules stored in the ARA-M or in an ARA-C can also be retrieved via RAM commands.

RAM commands can be accomplished by the TSM directly targeting the Secure Element via OTA (using SCP80 or SCP81 as defined respectively in [GP Card Spec] and in GlobalPlatform Remote Application Management over HTTP ([GP Amd B])), or via a Remote Admin Agent on the device (as defined in GlobalPlatform Secure Element Remote Application Management ([GP SE OTA])).

In some cases remote application management can also be performed via a standalone device application (e.g. in a TEE); however, that is not standardized within GlobalPlatform. If a device application is used to forward RAM commands to the Secure Element, this access of this device application to the Secure Element shall be explicitly granted by an access rule already stored in the Secure Element.

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 Applet (identified by the Applet's AID) and device application (identified by the device application's `DeviceAppID`).

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

Note: In the case of SCP80, the maximum length of a `REF-AR-DO` which can be stored in an ARA depends on the size of some input buffers of the SE, and the retrieval of the access rules is limited to the buffer size of the SE.

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.

Note: When the remote management on the ARA can be performed through different interfaces, the ARA shall terminate an already running STORE DATA session on a given interface if a STORE DATA command is received on another interface.

Table 5-1: STORE DATA Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	As specified in [GP Card Spec] **Note:** Only CLA '80' is applicable with SCP80 and SCP81.
INS	'E2'	STORE DATA as specified in [GP Card Spec]

P1 P2	P1: Reference control parameter
	P2: Block number
		Reference control parameter as specified in [GP Card Spec] with:
		• b8 indicating whether 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-REF-AR-DO`
		`Command-Delete`
		`Command-Update-Refresh-Tag-DO`
		`Command-Register-Client-AIDs-DO`
		1 for `Command-Get`
		`Command-Get-All`
		`Command-Get-Client-AIDs-DO`
		`Command-Get-Next`
		`Command-Get-Device-Config-DO`
		Block number as specified in [GP Card Spec].

Lc	Length of the DO in the Data field	

Code	Value	Meaning	
Data	Block-DO | Command_Type or Command_Type Command_Type is one of the following: Command-Store-REF-AR-DO Command-Delete Command-Update-Refresh-Tag-DO Command-Register-Client-AIDs-DO Command-Get Command-Get-All Command-Get-Client-AIDs-DO Command-Get-Next Command-Get-Device-Config-DO	Block-DO	Used to specify the data block sent to the ARA, when the STORE DATA is transmitted over SCP80 and some buffer sizes are limited. The presence of Block-DO is only applicable to Command-Store-REF-AR-DO, Command-Get, Command-Get-All, and Command-Get-Next.
		Command-Store-REF-AR-DO	Sent to an ARA to store access rules to this ARA of the SE
		Command-Delete	Sent to an ARA to delete access rules from this ARA of the SE
		Command-Update-Refresh-Tag-DO	Sent to an ARA to update the refresh tag managed by the ARA-M
		Command-Register-Client-AIDs-DO	Sent to the ARA-M to register an ARA-C identified by its AID.
		Command-Get	Sent to an ARA to retrieve access rules stored in this ARA
		Command-Get-All	Sent to the ARA-M to retrieve all access rules from the SE
		Command-Get-Client-AIDs-DO	Sent to the ARA-M to retrieve the AID of all the ARA-Cs registered to the ARA-M.
		Command-Get-Next	Sent to an ARA to retrieve the remaining access rules from this ARA.
		Command-Get-Device-Config-DO	Sent to the ARA-M to retrieve the configuration of the ACE.

Code	Value	Meaning

Le	Absent
	or
	'00'
		Not present for:
		`Command-Store-REF-AR-DO`
		`Command-Delete`
		`Command-Update-Refresh-Tag-DO`
		`Command-Register-Client-AIDs-DO`
		'00' for:
		`Command-Get`
		`Command-Get-All`
		`Command-Get-Client-AIDs-DO`
		`Command-Get-Next`
		`Command-Get-Device-Config-DO`

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-REF-AR-DO

Note: Formerly named `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-REF-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 in the ARA-M/ARA-C. **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

Storage of a rule

A STORE DATA (`Command-Store-REF-AR-DO`) command can contain only a single `REF-AR-DO`.

If the `REF-AR-DO` to be stored in the ARA cannot be transmitted in a single STORE DATA command:

- The initial STORE DATA (`Command-Store-REF-AR-DO`) command can be followed by subsequent STORE DATA commands.
- The access rule shall be stored in the ARA only when subsequent STORE DATA commands containing the whole `REF-AR-DO` have been received. That is, the ARA shall drop the already received parts of the `REF-AR-DO` in either of the following cases:
  - No subsequent STORE DATA command is received and the whole `REF-AR-DO` has not been received.
  - P1 is set to "last block" in a subsequent STORE DATA command before all the value bytes of the `REF-AR-DO` have been received.

Storage of several rules

If several `REF-AR-DO`s are to be stored, each `REF-AR-DO` needs to be transmitted using a separate STORE DATA (`Command-Store-REF-AR-DO`) command. These STORE DATA (`Command-Store-REF-AR-DO`) commands can be transmitted in a single INSTALL [for personalization] session.

Atomicity

The ARA-M/ARA-C shall ensure that the operation to update a `REF-AR-DO` and to update the refresh tag owned by the ARA-M is atomic. In other words, a rule downloaded over several STORE DATA commands (`REF-AR-DO` too long to fit a single STORE DATA (`Command-Store-REF-AR-DO`) command) will be stored in the ARA-M/ARA-C and the refresh tag updated, only once the whole rule has been received.

Rule overwrites in the current ARA

If a STORE DATA (`Command-Store-REF-AR-DO`) command contains a `REF-DO` already used by an existing access rule defined in the current ARA (i.e. concerning the same AID and the same `DeviceAppID`), the command is successful and the following behavior shall apply:

- If the access rule already existing in the ARA contains only an `APDU-AR-DO` and the rule being stored contains only an `APDU-AR-DO`, then the access rule stored in the ARA will be overwritten by the content of the `APDU-AR-DO` provided in the `Command-Store-REF-AR-DO`. Similarly, if the access rule already existing in the ARA contains only an `NFC-AR-DO` and the rule being stored contains only an `NFC-AR-DO`, then the access rule stored in the ARA will be overwritten by the content of the `NFC-AR-DO` provided in the `Command-Store-REF-AR-DO`.
- If the access rule already existing in the ARA contains only an `APDU-AR-DO` and the rule being stored contains only an `NFC-AR-DO`, then the access rule stored in the ARA will be updated by adding an `NFC-AR-DO` as provided in the `Command-Store-REF-AR-DO`, and the existing `APDU-AR-DO` remains unchanged. Similarly, if the access rule already existing in the ARA contains only an `NFC-AR-DO` and the rule being stored contains only an `APDU-AR-DO`, then the access rule stored in the ARA will be updated by adding an `APDU-AR-DO` as provided in the `Command-Store-REF-AR-DO` command, and the existing `NFC-AR-DO` remains unchanged.
- If the access rule already existing in the ARA contains an `APDU-AR-DO` and an `NFC-AR-DO` and the rule being stored also contains an `APDU-AR-DO` and an `NFC-AR-DO`, then the access rule stored in the ARA will be overwritten by content of the `APDU-AR-DO` and `NFC-AR-DO` provided in the `Command-Store-REF-AR-DO`.

Table 5-3 shows all possible overwrite scenarios.

Table 5-3: Overwrite Scenarios

Rule in ARA	Rule in STORE DATA	Result
NFC-AR-DO$_1$	APDU-AR-DO$_2$	(APDU-AR-DO$_2$, NFC-AR-DO$_1$)
APDU-AR-DO$_1$	NFC-AR-DO$_2$	(APDU-AR-DO$_1$, NFC-AR-DO$_2$)
APDU-AR-DO$_1$	APDU-AR-DO$_2$	(APDU-AR-DO$_2$)
NFC-AR-DO$_1$	NFC-AR-DO$_2$	(NFC-AR-DO$_2$)
(APDU-AR-DO$_1$, NFC-AR-DO$_1$)	APDU-AR-DO$_2$	(APDU-AR-DO$_2$, NFC-AR-DO$_1$)
(APDU-AR-DO$_1$, NFC-AR-DO$_1$)	NFC-AR-DO$_2$	(APDU-AR-DO$_1$, NFC-AR-DO$_2$)
NFC-AR-DO$_1$	(APDU-AR-DO$_2$, NFC-AR-DO$_2$)	(APDU-AR-DO$_2$, NFC-AR-DO$_2$)
APDU-AR-DO$_1$	(APDU-AR-DO$_2$, NFC-AR-DO$_2$)	(APDU-AR-DO$_2$, NFC-AR-DO$_2$)
(APDU-AR-DO$_1$, NFC-AR-DO$_1$)	(APDU-AR-DO$_2$, NFC-AR-DO$_2$)	(APDU-AR-DO$_2$, NFC-AR-DO$_2$)

Conflict detection between ARA instances

If a STORE DATA (`Command-Store-REF-AR-DO`) command contains a `REF-AR-DO` with a `REF-DO` which exactly corresponds to the `REF-DO` of an access rule already defined in another ARA (i.e. concerning the same AID and the same `DeviceAppID`), then the STORE DATA (`Command-Store-REF-AR-DO`) command is rejected with SW '6A 89'.

Specific versus generic rules

If a specific rule is to be stored and a generic rule with the same AID or `DeviceAppID` already exists (in the same ARA or in another ARA), then the specific rule will be stored.

If a generic rule is to be stored and a specific rule with the same AID or `DeviceAppID` already exists, then the generic rule will be stored.

ARA-M not available

A STORE DATA (`Command-Store-REF-AR-DO`) must be rejected by an ARA-C with the error SW '69 85' if an ARA-M does not exist in the SE, or an existing ARA-M is disabled, or the ARA-C is not connected to an existing ARA-M.

Management of unknown BER-TLVs

If a STORE DATA (`Command-Store-REF-AR-DO`) command contains a `REF-AR-DO` including all required data objects as defined in this specification and also unknown BER-TLV tags, the ARA-M/ARA-C shall discard the unknown BER-TLV objects and all associated data and respond with the warning SW '63 82'. If required data objects are missing, the ARA-M/ARA-C shall discard the whole rule (`REF-AR-DO`) and answer with the error SW '6A 80'.

Command-Delete

Note: Formerly named Command-Delete-AR-DO.

This data object deletes access rules in this 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.

If the Command-Delete refers to a rule already deleted, then the response shall be SW '6A 88'.

Table 5-4: Command-Delete

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 current ARA.
			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 data objects in the `REF-AR-DO` (`APDU-AR-DO`, `NFC-AR-DO`, or both) must be empty (tag present, length field set to 0, value field empty).
			The effect on the access rule depends on whether the `REF-AR-DO` contains an `APDU-AR-DO`, an `NFC-AR-DO`, or both, as described in Table 5-5.
			Empty:
			All access rules are deleted.
			Length:
			n is the full length of all value bytes
			0 if no data object is specified.
				Mandatory

Table 5-5: REF-AR-DO in Command-Delete

`REF-AR-DO` contains:	If an `APDU-AR-DO` exists in the access rule:	If an `NFC-AR-DO` exists in the access rule:
`APDU-AR-DO` only	The `APDU-AR-DO` is deleted.	The `NFC-AR-DO` remains unchanged.
`NFC-AR-DO` only	The `APDU-AR-DO` remains unchanged.	The `NFC-AR-DO` is deleted.
`NFC-AR-DO` and `APDU-AR-DO`	The `APDU-AR-DO` is deleted.	The `NFC-AR-DO` is deleted.

Command-Update-Refresh-Tag-DO

Note: Formerly named `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 5-6: Command-Update-Refresh-Tag-DO

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

Command-Register-Client-AIDs-DO

Note: Formerly named `Command-Register-ClientAIDs-DO`.

This data object can be used to register an ARA-C to the ARA-M. An ARA-C is identified by its AID. Several ARA-Cs can be registered by submitting this data object several times with the AIDs of the different ARA-Cs. (Registration of multiple ARA-Cs in a single data object is deprecated.)

This data object shall only be processed by the ARA-M. The ARA-M shall register the ARA-C(s) identified by the provided AID(s). If a provided AID does not correspond to an installed ARA-C, then this AID shall be ignored 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 5-7: Command-Register-Client-AIDs-DO

Tag	Length	Value	Meaning	Presence
'F7'	n	`AID-REF-DO`$_1$ | … | `AID-REF-DO`$_x$ (deprecated) or `AID-REF-DO`	Register an ARA-C to the ARA-M provided the ARA-C is installed on the SE. [Deprecated: Register ARA-Cs to the ARA-M provided these ARA-Cs are installed on the SE.] Value: `AID-REF-DO` corresponding to the AID of an ARA-C to be registered to the ARA-M. [Deprecated: More than one `AID-REF-DO`, 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

Note: Formerly named `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 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 5-8: Command-Get

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`) is sent to an ARA-C, get all the access rules stored to this ARA-C. If this STORE DATA (`Command-Get`) 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`) 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-Get-All

Note: Formerly named `Command-GetAll-AR-DO`.

This data object is used to retrieve all the access rules stored to the SE. This data object 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 5-9: Command-Get-All

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

Command-Get-Client-AIDs-DO

Note: Formerly named `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 5-13) holding a list of `AID-REF-DO`s indicating the AID of each of the ARA-Cs registered 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 5-10: Command-Get-Client-AIDs-DO

Tag	Length	Value	Meaning	Presence
'F6'	0	Empty	Get the AIDs of all ARA-Cs registered to the ARA-M.	Mandatory

Command-Get-Next

Note: Formerly named `Command-GetNext-AR-DO`.

This data object is used to retrieve the remaining data after a first `Command-Get`, `Command-Get-All`, `Command-Get-Client-AIDs-DO`, or `Command-Get-Device-Config-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.

If the data retrieval process did not start with a `Command-Get`, `Command-Get-All`, `Command-Get-Client-AIDs-DO`, or `Command-Get-Device-Config-DO`, then the `Command-Get-Next` shall be rejected with SW '69 85'.

Table 5-11: Command-Get-Next

Tag	Length	Value	Meaning	Presence
'F5'	0	Empty	Get the remaining data after a first `Command-Get`, `Command-Get-All`, `Command-Get-Client-AIDs-DO`, or `Command-Get-Device-Config-DO` when all the requested rules couldn't be fetched in response to this first command	Mandatory

Note: An ARA shall manage the retrieval and storage of rules at the same time. For the RAM interface:

If an access rule has been updated during the access rule retrieval process, then:

- The ARA shall reject the STORE DATA (`Command-Get-Next`) command with SW '69 84'.
- The remote management entity shall discard the access rules or part of the access rules already retrieved and shall restart the full rule retrieval procedure by re-sending either a STORE DATA (`Command-Get-All`) or a STORE DATA (`Command-Get`) command. See section 4.1 for detail on concurrency between rule retrieval and rule storage.

Command-Get-Device-Config-DO

This data object is used to retrieve the configuration of the ACE implemented in the device or the list of the different configurations if several ACEs are implemented in the device. This configuration includes the version of the Device Interface implemented by the ACE.

This data object shall only be processed by 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 5-12: Command-Get-Device-Config-DO

Tag	Length	Value	Meaning	Presence
'F8'	0	Empty	Get the configuration(s) of the ACE(s) implemented in the device	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-REF-AR-DO`, Command-Delete, `Command-Update-Refresh-Tag-DO`, or `Command-Register-Client-AIDs-DO`, then there is no response data.

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

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

If the command data object used in the STORE DATA request is `Command-Get-Client-AIDs-DO`, then the response message Data field contains a Response-ARAC-AID-DO as defined in Table 5-13. The Response-ARAC-AID-DO includes the AIDs of all the ARA-Cs registered to the ARA-M.

If the command data object used in the STORE DATA request is `Command-Get-Device-Config-DO`, then the response message Data field contains a `Response-Device-Config-DO` as defined in Table 5-14.

In each case (`Command-Get`, `Command-Get-All`, `Command-Get-Client-AIDs-DO`, or `Command-Get-Device-Config-DO`), if the `Response-ALL-REF-AR-DO`, Response-ARAC-AID-DO, or `Response-Device-Config-DO` is too large to fit in the STORE DATA response, then the remaining bytes can be retrieved using STORE DATA (`Command-Get-Next`) commands.

In response to STORE DATA (`Command-Get-Next`), the response data includes the remaining bytes of the `Response-ALL-REF-AR-DO`, Response-ARAC-AID-DO, or `Response-Device-Config-DO` that couldn't be fetched in response to the initial STORE DATA (`Command-Get`), STORE DATA (`Command-Get-All`), STORE DATA (`Command-Get-Client-AIDs-DO`), or STORE DATA (`Command-Get-Device-Config-DO`). The response to the STORE DATA (`Command-Get-Next`) doesn't include any Tag Length fields but only the next bytes of the `Response-ALL-REF-AR-DO`, Response-ARAC-AID-DO, or `Response-Device-Config-DO`.

Response-ARAC-AID-DO

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

Table 5-13: 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

Response-Device-Config-DO

In response to STORE DATA (`Command-Get-Device-Config-DO`), the ARA-M shall return the configuration of the ACEs that have accessed the ARA-M since the last device initialization. More specifically, the ARA-M shall return a `Response-Device-Config-DO` (as defined in Table 5-14) as follows:

- If no ACE has accessed the ARA-M since the last device initialization, the `Response-Device-Config-DO` shall be empty.
- If a single ACE has accessed the ARA-M since the last device initialization, the `Response-Device-Config-DO` shall contain the configuration of that ACE.
- If several ACEs have accessed the ARA-M since the last device initialization, the `Response-Device-Config-DO` shall contain a list of the different configurations implemented by those ACEs.
    - If one of the ACEs did not sent a `Device-Config-DO` by using the command GET DATA [Config], the ARA-M shall assume that this ACE implements v1.0 of this specification and shall append a `Device-Config-DO` with the `Device-Interface-Version-DO` '01 00 00' in the `Response-Device-Config-DO`.
    - If several ACEs implement the same `Device-Config-DO`, the ARA-M shall append only one instance of this `Device-Config-DO` in the `Response-Device-Config-DO`.
    - A maximum of five `Device-Config-DO`s can be returned. If more than five `Device-Config-DO`s are required, only the first five `Device-Config-DO`s will be returned in the `Response-Device-Config-DO`.

Table 5-14: Response-Device-Config-DO

Tag	Length	Value	Meaning	Presence
'FF 7F'	n	`Device-Config-DO` or `Device-Config-DO`$_1$ |…| `Device-Config-DO`$_x$ or Empty	**Value:** If a single ACE has accessed the ARA-M since the last device initialization, `Device-Config-DO` containing the configuration of that ACE. If several ACEs have accessed the ARA-M since the last device initialization, list of `Device-Config-DO`s containing no more than five different configurations implemented by those ACEs. Empty if no ACE has accessed the ARA-M since the last Secure Element power up. **Length:** n is the length of all the `Device-Config-DO`s. 0 if no ACE has accessed the ARA-M since the last Secure Element power up.	Mandatory

5.1.2.2 Response Message Status Words

If the STORE DATA command is rejected due to inconsistencies or wrong coding within the data objects defined in this specification, the ARA-M/ARA-C shall reject the STORE DATA command with the appropriate error SW as defined in Table 5-15.

Block-DO

If `Block-DO` is received, but is not supported by the ARA-M/ARA-C, SW '69 81' shall be returned.

The presence of a `Block-DO` is only applicable for a STORE DATA (`Command-Store-REF-AR-DO`, `Command-Get`, `Command-Get-All`, or `Command-Get-Next`). If any other STORE DATA (Command-Delete, `Command-Update-Refresh-Tag-DO`, `Command-Register-Client-AIDs-DO`, `Command-Get-Client-AIDs-DO`, or `Command-Get-Device-Config-DO`) includes a `Block-DO`, the ARA-M/ARA-C shall reject the STORE DATA command with the SW '69 85'.

Note: As `Block-DO` was not defined in version 1.0 of the specification, an ARA-M/ARA-C implementing v1.0 of this specification may reject a STORE DATA with a `Block-DO` with an error SW '6A 80'.

Unknown BER-TLVs

If the STORE DATA (`Command-Store-REF-AR-DO`) command includes one or more unknown BER-TLVs inside the `REF-AR-DO` but all other consistency checks are performed successfully, the ARA-M/ARA-C shall discard the unknown BER-TLVs in the `REF-AR-DO` and all associated data, store the rest of the `REF-AR-DO`, and respond with the warning SW '63 82'.

A successful execution of the command (other than one that requires the warning SW '63 82') shall be indicated by status bytes '90 00'.

Table 5-15: STORE DATA Response Message Status Words

SW1	SW2	Meaning
'63'	'81'	Rule successfully stored but an access rule already exists for this target (deprecated – see note following table)
'63'	'82'	Rule successfully stored but the access rule contained at least one unknown BER-TLV that has been discarded
'65'	'81'	Memory problem
'67'	'00'	Wrong length in Lc
'69'	'81'	DO is not supported by the ARA-M/ARA-C
'69'	'82'	Security status not satisfied
'69'	'84'	Rules have been updated and must be read again to ensure consistency
'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

Note: Some ARA vendors may use SW '63 81', which was defined in v1.0 of this specification, but the behavior of the ARA in this case was implementation dependent. With the clarification of conflict detection between ARA instances in section 5.1.1.1, SW '63 81' is no longer applicable.

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

Tags

The ranges of tag values listed in Table 6-1 are reserved for data objects in this specification and shall not be used in additional BER-TLVs (considered as unknown BER-TLVs in this specification) supported in customized ARA and ACE implementations.

Table 6-1: Reserved Data Object Tags for GlobalPlatform SEAC Specification

Tag Ranges Reserved for GlobalPlatform SEAC Specification	
• '4F'	• 'E0' to 'EB'
• 'C0' to 'CB'	• 'F0' to 'FB'
• 'D0' to 'DB'	• 'FF 1F' to 'FF 7F'
• 'DF 1F' to 'DF 7F'	

Currently defined data object tags are summarized in Annex B.

Unknown BER-TLVs

When receiving a BER-TLV with an unknown tag, the ACE and the ARA-M/ARA-C shall discard the BER-TLV object and all data within it.

If the ARA-M/ARA-C implements additional BER-TLVs for custom purpose, those additional BER-TLVs might be processed by a customized ACE implementation. However, this specification guarantees that:

- A non-customized ARA will be able to manage customized access rules received from a customized Remote Administration server: The customized part of the access rule will be discarded.
- A non-customized ACE will be able to manage customized access rules received from a customized ARA-M: The customized part of the access rule will be discarded.

Those additional BER-TLVs are considered as unknown BER-TLVs in this specification and might be supported by customized ARA and ACE implementations. The handling of such additional BER-TLVs, to be implemented by a customized ARA and ACE, depends on individual schemes and is completely out of the scope of this GlobalPlatform specification.

Additional BER-TLVs to be defined for custom purpose have to use tags that comply with the following table:

Table 6-2: Reserved Data Object Tags for Proprietary Usage

Tag Ranges Reserved for Proprietary Usage	
• 'CC' to 'CE'	• 'EC' to 'EE'
• 'DC' to 'DE'	• 'FC' to 'FE'
• 'DF 80' to 'DF 8F'	• 'FF 80' to 'FF 8F'

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 in the SE.

The AID and `DeviceAppID` reference data objects assign access rules to a specific Applet and device application. If access rules shall be stored in the ARA or retrieved from the ARA it is necessary to specify the Applet and device application to create a unique assignment to these access rules. The Applet 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 `DeviceAppID` specified in the `DeviceAppID` reference data object (`DeviceAppID-REF-DO`).

AID-REF-DO

The `AID-REF-DO` shall be used to store and retrieve the corresponding access rules for an Applet (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 6-3: AID-REF-DO

Tag	Length	Value	Meaning
'4F'	5-16 or 0	AID or Empty	**Value:** AID: Identifies the specific Applet for which rules are to be stored or retrieved. This AID can also be a partial AID (containing for example only the RID, as defined in [7816-5]). Empty: Indicates that the rules to be stored or retrieved are associated with all Applets not covered by a specific rule **Length:** 5-16 for an AID according to [7816-5] 0 for empty value field
'C0'	0	Empty	Implicitly selected application (all channels)

When trying to find a rule corresponding to an Applet, the AID targeted can be a full AID or a partial AID. A partial AID is used to perform SELECT [by name] [first occurrence] command and then one or more SELECT [by name] [next occurrence] commands as defined in [7816-4]. In both cases, full AID and partial AID, the ARA-M (when ACE uses GET DATA [Specific] (deprecated)) or the ACE (when using GET DATA [All]) shall only consider rules having an `AID-REF-DO` with an AID value matching exactly with the AID given in the `AID-REF-DO` of the GET DATA [Specific] (deprecated) or specified by the device application to the ACE through the SE Access API.

Note: When a device application is selecting Applets using selection by partial AID, access to the Applets by the device application will only be granted if there is an access rule defined for this partial AID.

DeviceAppID-REF-DO

The `DeviceAppID-REF-DO` shall be used to store and retrieve the corresponding access rules for a device application (which is identified by the `DeviceAppID`) to and from the ARA:

Table 6-4: DeviceAppID-REF-DO

Tag	Length	Value	Meaning
'C1'	32, 20, or 0	`DeviceAppID` or Empty	**Value:** `DeviceAppID`: ○ Identifies the specific device application that the rules apply to. ○ Contains one of following values: ▪ Hash of the certificate of the Application Provider: Used in most cases when the application is running in the REE. ▪ Unique identifier (this could be a TA UUID) of the application: Used when the application is running in the TEE or when it is running in the REE but the certificate is not an appropriate identifier (see section 3.1.2). This unique identifier shall be padded with 'FF' in order to provide a length of 20 bytes. Empty: Indicates that the rules apply to all device applications not covered by a specific rule. **Length:** 32 for 32-byte SHA-256 hash value 20 for 20-byte SHA-1 hash value (deprecated) 20 for unique identifier (padded with 'FF' if necessary) 0 for empty value field

To allow transitions from SHA-1 (deprecated) to SHA-256, support for both SHA-1 (deprecated) and SHA-256 hash values is mandatory for ARA and ACE compliant with this version of the document.

While support for SHA-1 is deprecated for future versions, support for SHA-256 hash values will remain mandatory for ARA and ACE compliant to future Device Interface Versions.

REF-DO

The `REF-DO` contains an `AID-REF-DO` and a `DeviceAppID-REF-DO` which uniquely reference a specific set of access rules assigned for a given Applet (which is identified by its AID) and a device application (which is identified by the `DeviceAppID`).

Table 6-5: REF-DO

Tag	Length	Value	Meaning
'E1'	n	`AID-REF-DO` | `DeviceAppID-REF-DO`	**Value:** A concatenation of an `AID-REF-DO` and a `DeviceAppID-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 Applet (AID reference) and device application (`DeviceAppID` reference).

Table 6-6: 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 6-7: 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 generic 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 6-8: 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 generic 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((APDU_header & APDU_filter_mask) == APDU_filter_header)` then allow APDU **Length:** 1 if value contains a generic 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 Applet using the SE Access API. However, when the Applet 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 `DeviceAppID` to authorize device applications to receive NFC events issued by the Applet.

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 6-9: 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 NFC event access rule exists, then the ACE shall not allow the sending of the NFC event to the targeted device application, unless this device application is explicitly allowed by an APDU access rule to access the Applet which is referenced by the AID parameter in the HCI EVT_TRANSACTION as defined in [102 622].

6.3 Configuration Management Data Objects

The GET DATA and STORE DATA commands require a set of data objects in the Data field for retrieving the configuration of the ACE and of the ARA-M.

Device-Config-DO

The ACE shall use the `Device-Config-DO` to inform the ARA-M of its configuration, when sending the GET DATA [Config] command.

The TSM can retrieve the `Device-Config-DO` by sending the STORE DATA (`Command-Get-Device-Config-DO`) command to the ARA-M.

In this version of the specification, `Device-Config-DO` contains only the version of the Device Interface implemented by the ACE (`Device-Interface-Version-DO`).

Table 6-10: Device-Config-DO

Tag	Length	Value	Meaning
'E4'	5	`Device-Interface-Version-DO`	**Value:** Version of the Device Interface implemented by the ACE.

ARAM-Config-DO

The ARA-M shall use the `ARAM-Config-DO` to inform the ACE of its configuration, in response to the GET DATA [Config] command.

In this version of the specification, `ARAM-Config-DO` contains only the version of the Device Interface implemented by the ARA-M (`Device-Interface-Version-DO`).

Table 6-11: ARAM-Config-DO

Tag	Length	Value	Meaning
'E5'	5	`Device-Interface-Version-DO`	**Value:** Version of the Device Interface implemented by the ARA-M.

Device-Interface-Version-DO

The `Device-Interface-Version-DO` shall be used to identify which version of the Device Interface is implemented either by the ACE or the ARA-M.

The version of the device interface is incremented only when there is a change in the specification of the Device Interface defined in Chapter 4.

Table 6-12: Device-Interface-Version-DO

Tag	Length	Value	Meaning
'E6'	3	Version	**Value:** Version of the Device Interface implemented by the ARA-M or the ACE. Version is a 3-byte number ‘xyz’, corresponding to the coding of a version x.y.z. In this version of the specification, v1.2.0, the Version shall be set to '01 02 00'.

Block-DO

The `Block-DO` is used to specify a data block which shall be stored in the ARA or retrieved from an ARA. The data block is specified by offset and length. `Block-DO` may be included in STORE DATA (`Command-Store-REF-AR-DO`), STORE DATA (`Command-Get-All`), STORE DATA (`Command-Get`), and STORE DATA (`Command-Get-Next`) as an additional data object. `Block-DO` allows adaptation to certain payload size constraints for transmissions. The Remote Administration Server shall use `Block-DO` only if many access rules or long access rules need to be transmitted over SCP80 and the incoming or the outgoing buffer size is too small to perform this transmission within one INSTALL [for personalization] session. In this case data can be stored or retrieved block by block over several INSTALL [for personalization] sessions by using `Block-DO`.

Table 6-13: Block-DO

Tag	Length	Value	Meaning
'E7'	3	offset | length	**3 bytes: 2 bytes offset, 1 byte length** The offset and length specify the data block of access rule data which shall be: ○ stored in the session of STORE DATA (`Command-Store-REF-AR-DO`) or ○ retrieved in the session of STORE DATA (`Command-Get-All`) or STORE DATA (`Command-Get`) and the following STORE DATA (`Command-Get-Next`). The 2 bytes of offset are stored in big-endian (high byte first) order.

Usage of Block-DO for the storage of an access rule:

- For a `Block-DO` included in the initial STORE DATA (`Command-Store-REF-AR-DO`), the offset must be set to 0.
- For a `Block-DO` included in any subsequent STORE DATA (`Command-Store-REF-AR-DO`):
  - If the offset is not equal to the (offset + length) of the previous command, the ARA shall return SW '6A 80'. However, the ARA shall not reject a repetition of the previous command.
  - If the offset exceeds the length of the remaining part of the `REF-AR-DO` being stored, the ARA returns SW '6A 80'.
- The length in the `Block-DO` is the length of the part of the `REF-AR-DO` being stored.

  Note: When computing the length of the `Block-DO`, the Remote Administration Server shall consider that additional bytes are needed and also have to fit into the SE incoming buffer:
  - Additional bytes for the ETSI TS 102 225 ([102 225]) command packet header and ETSI TS 102 226 ([102 226]) command script format, as required by SCP80
  - Additional bytes for coding of the STORE DATA (`Command-Store-REF-AR-DO`)
- If the length is not equal to the length of the part of the `REF-AR-DO` being stored, the ARA returns SW '6A 80'.
- If a rule storage session has been initiated with a STORE DATA (`Command-Store-REF-AR-DO`) including a `Block-DO`, all the following STORE DATA () shall include a `Block-DO`, otherwise the ARA shall return SW '69 85'.

Usage of Block-DO for the retrieval of access rules:

- The offset in a `Block-DO` included in a STORE DATA (`Command-Get-All`) or STORE DATA (`Command-Get`) must be set to 0.
- If the offset in a STORE DATA (`Command-Get-Next`) is not equal to the (offset + length) of the previous command, the ARA shall return SW '6A 80'. However, the ARA shall not reject a repetition of the previous command.
- If the offset is longer than the whole `Response-ALL-REF-AR-DO` being retrieved, the ARA returns SW '6A 80'.
- The length in the `Block-DO` shall not exceed the length of data which is available for the ARA in the SE outgoing buffer to transfer the response data to the Remote Administration Server.

  Note: When computing the length of the `Block-DO`, the Remote Administration Server shall consider that additional bytes have to fit into the SE outgoing buffer for the [102 225] response packet header and [102 226] response script format, which are required by SCP80.
- If the length exceeds the length of the remaining part of the `Response-All-REF-AR-DO` being retrieved, the ARA returns SW '90 00' and only the remaining data of the `Response-All-REF-AR-DO` is sent.
- If a rule retrieval session has been initiated with a STORE DATA (`Command-Get-All`) or STORE DATA (`Command-Get`) including a `Block-DO`, the following STORE DATA (`Command-Get-Next`) shall include a `Block-DO`; otherwise the ARA shall return SW '69 85'.

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 structure. The ASN.1 object description is based on the implicit ASN.1 encoding (see [X.690]), except when explicitly mentioned (e.g. AID).

There may be two different file structures in the Secure Element:

- DODF(1), a file structure for SHA-1 (deprecated) hash algorithm derived `DeviceAppIDs`. This file structure is identical to the one defined in previous versions of this specification and will potentially be removed in future versions of this specification.
- DODF(2), a file structure introduced in version 1.2 of this specification for SHA-256 hash algorithm derived `DeviceAppIDs`.

Each file structure is in a separate DODF file under the PKCS#15 ODF file. See section 7.1.3 for the registered OIDs of the two file structures.

DODF(1) and DODF(2) have the same file content and file structure; they differ only in the hash algorithm used to derive the `DeviceAppID` in the Condition object of the Access Control Conditions File (ACCF), defined in section 7.1.6.

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 Master File (MF) shall not be used. The path encoding is defined in [PKCS15].

7.1.1 File Padding

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

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

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 EFdir, and then reads the EFdir 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…)

7.1.3 PKCS#15 DODF

PKCS#15 DODFs used as the entry point to access control data have an `oidDO` entry with an OID that must be in the GlobalPlatform scope (under {iso(1) member-body(2) country-USA(840) globalPlatform(114283)}).

There may be two DODF files for this specification:

- DODF(1), for rules where the SHA-1 (deprecated) hash algorithm is used to derive `DeviceAppIDs`:
  The registered OID for this DODF(1) is the same as the OID in previous versions of this specification:

  {iso(1) member-body(2) country-USA(840) globalPlatform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}

  This file exists for backward compatibility with ACEs that support previous versions of this specification.

- DODF(2), for rules where the SHA-256 hash algorithm is used to derive `DeviceAppIDs`:
  The registered OID for this DODF(2) is:

  {iso(1) member-body(2) country-USA(840) globalPlatform(114283) device(200) seAccessControl(1) accessControlMainFileV2(2)}

  This file is used by ACEs that support the current version of this specification. Such ACEs should first attempt to read this SHA-256 based DODF(2), and if not found, should then attempt to read the deprecated SHA-1 based DODF(1).

An ACE complying with v1.2 or later of this specification and using ARF MUST first search for a matching rule in DODF(2). If no matching rule is found in DODF(2), and if DODF(1) is present, it then searches for a matching rule in DODF(1).

For backward compatibility with previous versions of this specification DODF(1) shall come first in the ODF (i.e. before DODF(2), if both DODFs are present).

According to [PKCS15], a 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 a DODF.

The `oidDO` structure of the DODF `DataType oidDO` entry shall contain an id value set to one of the registered OIDs defined above and a value that is a path structure to the respective 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 a 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 by checking whether the id value of the `DataType oidDO` entry is one of the registered OIDs defined above.

7.1.4 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)

<table>
<tr><td colspan="2">Identifier: 'xxxx'</td><td colspan="2">Structure: transparent file</td><td>Mandatory</td></tr>
<tr><td colspan="2">File length: n bytes</td><td colspan="3">Update activity: low</td></tr>
<tr><td colspan="5">Access Conditions:
READ ALW
UPDATE ADM
DEACTIVATE ADM
ACTIVATE ADM</td></tr>
<tr><td>Bytes</td><td colspan="2">Description</td><td>M/O</td><td>Length</td></tr>
<tr><td>1 to n</td><td colspan="2">AccessControlMainFile</td><td>M</td><td>n bytes</td></tr>
</table>

There shall be only one ACMF file per DODF. If a DODF contains several ACMF files, then the security shall be considered compromised and the ACE shall forbid access to all the Applets 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
}
```

7.1.5 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 DODF. If a DODF contains several ACRF files, then the security shall be considered compromised and the ACE shall forbid access to all the Applets for, and only for, this specific Secure Element.

Each Rule object explicitly or implicitly identifies a set of Applets, 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
}
```

The Target object indicates whether the rule applies to one Applet (identified by its AID), or the implicitly selected application, or all other applications (all the Applets 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 Applet,
-- the implicitly selected Applet, or all other Applets
-- When defining an APDUAccessRule the access control target shall
be the AID of the Applet to be selected for APDU exchange.
-- When defining an NFCAccessRule the access control target shall be
the one referenced by the AID parameter in HCI EVT_TRANSACTION as
defined in ETSI TS 102 622 ([102 622]).
Target ::= CHOICE {
  -- the AID of the targeted Applet
  aid [0]EXPLICIT AID,

  -- the (unnamed) default selected Applet
  -- (applies to implicitly selected Applet
  -- on all logical channels)
  default [1]NULL,

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

The AID object uses the following ASN.1 syntax:

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

7.1.6 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 `DeviceAppID` 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 Applets pointed to by the Rule object.

If this file contains a condition without a `DeviceAppID`, then any Rule object pointing to this file is granting any device application access to the Applets 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 DeviceAppID of the authorized device application;
  -- if not indicated, then the Rule pointing to this Condition
  -- applies to all the device applications
  deviceAppID DeviceAppID OPTIONAL,
  accessRules [0]AccessRules OPTIONAL
}

-- DeviceAppID contains one of following values:
--    Hash of the certificate of the device application provider.
--    Unique identifier (this could be a TA UUID)
--    padded with 'FF' in order to provide a length of 20 bytes (in
--    DODF(1), or 32 bytes in DODF(2)) .
DeviceAppID ::= OCTET STRING (SIZE(20|32))

-- 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,
   apduFilters [1]APDUFilters
```

```
            }

            -- 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
            }

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

The size of the `DeviceAppID` in the Condition Object depends on the DODF that references the Condition Object.

- If the Condition Object is referenced from DODF(1), the size of `DeviceAppID` is 20 bytes. In this case the `DeviceAppID` is a SHA-1 (deprecated) hash derived from the application certificate or a unique identifier (e.g. the TA UUID) padded with 'FF'.
- If the Condition Object is referenced from DODF(2), the size of `DeviceAppID` is 32 bytes. In this case the `DeviceAppID` is a SHA-256 hash derived from the application certificate or a unique identifier (e.g. the TA UUID) padded with 'FF'.

Undefined Attributes in Condition Object

The Condition object includes the optional attribute `AccessRules`, which includes optional attributes `APDUAccessRule` and `NFCAccessRule`. If, when retrieving rules from the Access Rule Files (ARF), the ARA-M or the ACE determines that one of these attributes is not defined, it shall interpret the missing attribute as specified in Table 7-4.

Table 7-4: Required Interpretation If Undefined Attribute in Condition Object

	Policy Being Checked, for a Specific AID or for Other AIDs	Attributes in Condition Object Specified in ARF	Required Interpretation
1	APDU permission policy	AccessRules attribute not present or AccessRules attribute present but empty	APDU permission ALWAYS allowed policy
2		AccessRules attribute present • No APDUAccessRule attribute • NFCAccessRule attribute present	• For v1.0 of this specification, APDU permission ALWAYS allowed policy • For v1.1 (or newer) of this specification, APDU permission NEVER allowed policy
3	NFC event policy	AccessRules attribute not present or AccessRules attribute present but empty	NFC event ALWAYS allowed policy
		AccessRules attribute present • No NFCAccessRule attribute • APDUAccessRule attribute present	
4		○ APDUAccessRule set to ALWAYS or contains APDU filter	NFC event ALWAYS allowed policy
5		○ APDUAccessRule set to NEVER	NFC event NEVER allowed policy

Notes:

- The AccessRules attribute is set as optional to ensure backward compatibility with the former implementation of the ARF.
- Management of missing attributes was clarified in this release to ensure consistency between ARA and ARF behaviors. As a result, different versions of this specification specify different behavior when the APDUAccessRule attribute is not present but the NFCAccessRule attribute is present, as shown in row 2 in the table above.

  To avoid any interoperability issue, it is strongly recommended that the attributes for both APDU permission and NFC event are defined explicitly when setting a rule.

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)
globalPlatform(114283) device(200) seAccessControl(1)
accessControlMainFile(1) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

AID ::= OCTET STRING

DeviceAppID ::= OCTET STRING (SIZE(20|32))

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 {
deviceAppID DeviceAppID OPTIONAL,
accessRules [0]AccessRules OPTIONAL
}

AccessRules ::= SEQUENCE OF AccessRule

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

```
APDUAccessRule ::= CHOICE {
apduPermission [0]APDUPermission,
apduFilters [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 this access control mechanism is specified in a 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 this access control mechanism is specified in a 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) but none of the OIDs of the access control mechanism is 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.

7.4 ARF Parsing

Errors may occur when parsing the access control mechanism structure (ACMF, ACRF, ACCF files) after the reading of the OID in the DODF:

- If the ACMF file cannot be found or cannot be read or its content is invalid, then the device SHALL be denied any access to these Applets.
- If the ACRF file cannot be found or cannot be read or its content is invalid, then the device SHALL be denied any access to these Applets.
- If one ACCF file referenced in the ACRF file cannot be found or cannot be read or its content is invalid, then:
    - Rules applied to Secure Elements application(s) identified by their AIDs and using this ACCF, access SHALL be denied. Other rules referenced in the ACRF and applied to other Applet(s) remain valid and SHALL be used.
    - Rules applied to ALL Secure Elements applications and using this ACCF SHALL be discarded. Other rules referenced in the ACRF remain valid and SHALL be used.

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. (Otherwise, the full nesting is shown elsewhere in this annex.)

Note: The following data objects were renamed in v1.1:

- `Response-ALL-AR-DO => Response-ALL-REF-AR-DO`
- `Response-RefreshTag-DO => Response-Refresh-Tag-DO`
- `Command-Store-AR-DO => Command-Store-REF-AR-DO`
- `Command-Delete-AR-DO => Command-Delete`
- `Command-UpdateRefreshTag-DO => Command-Update-Refresh-Tag-DO`
- `Command-Register-ClientAIDs-DO => Command-Register-Client-AIDs-DO`
- `Command-Get-AR-DO => Command-Get`
- `Command-GetAll-AR-DO => Command-Get-All`
- `Command-Get-ClientAIDs-DO => Command-Get-Client-AIDs-DO`
- `Command-GetNext-AR-DO => Command-Get-Next`

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

Command	P1 P2	Data	Value		
GET DATA	[All]	absent			
	[Specific] (deprecated)	**REF-DO**	AID-REF-DO | DeviceAppID-REF-DO		
				AID-REF-DO	AID
					Empty
				DeviceAppID-REF-DO	DeviceAppID (see section 3.1)
					Empty
	[Refresh tag]	absent			
	[Next]	absent			
	[Config]	**Device-Config-DO**	**Device-Interface-Version-DO**		Version

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

P1 P2 of GET DATA Command	GET DATA Response Includes				
[All]	Response-ALL-REF-AR-DO				
		Empty			
		REF-AR-DO$_1$ | ... | REF-AR-DO$_x$, first bytes			
			REF-DO | AR-DO		
[Specific] (deprecated)	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-Refresh-Tag-DO				
		RefreshTag (8-byte random number)			
[Next]					
		next bytes of REF-AR-DO			
		next bytes of AR-DO			
[Config]	Response-ARAM-Config-DO				
		ARAM-Config-DO			
			Device-Interface-Version-DO		

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

Command	Data	Value
STORE DATA	Command-Store-REF-AR-DO	REF-AR-DO
	Command-Delete	AID-REF-DO
		REF-DO
		REF-AR-DO
		Empty
	Command-Update-Refresh-Tag-DO	Empty
	Command-Register-Client-AIDs-DO	AID-REF-DO$_1$ | … | AID-REF-DO$_x$, first bytes (deprecated)
		AID-REF-DO
	Command-Get	Empty
		AID-REF-DO
	Command-Get-All	Empty
	Command-Get-Client-AIDs-DO	Empty
	Command-Get-Next	Empty
	Command-Get-Device-Config-DO	Empty

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

STORE DATA Command	STORE DATA Response Includes	
(Command-Get)	Response-ALL-REF-AR-DO	
(Command-Get-All)	Response-ALL-REF-AR-DO	
(Command-Get-Client-AIDs-DO)	Response-ARAC-AID-DO	
		Empty
		AID-REF-DO$_1$ | … | AID-REF-DO$_x$, first bytes
(Command-Get-Next)		
		Next bytes of REF-AR-DO
		Next bytes of AID-REF-DO
(Command-Get-Device-Config-DO)	Response-Device-Config-DO	
		Empty
		Device-Config-DO
		Device-Config-DO$_1$ |…| Device-Config-DO$_x$

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 Applets not covered by a specific rule
'C0'	AID-REF-DO	Implicitly selected application (all channels)
'C1'	DeviceAppID-REF-DO	
'D0'	APDU-AR-DO	
'D1'	NFC-AR-DO	
'DF 20'	Response-Refresh-Tag-DO	
'DF 21'	Response-ARAM-Config-DO	
'E1'	REF-DO	
'E2'	REF-AR-DO	
'E3'	AR-DO	
'E4'	Device-Config-DO	
'E5'	ARAM-Config-DO	
'E6'	Device-Interface-Version-DO	
'E7'	Block-DO	
'F0'	Command-Store-REF-AR-DO	
'F1'	Command-Delete	
'F2'	Command-Update-Refresh-Tag-DO	
'F3'	Command-Get	
'F4'	Command-Get-All	
'F5'	Command-Get-Next	
'F6'	Command-Get-Client-AIDs-DO	
'F7'	Command-Register-Client-AIDs-DO	
'F8'	Command-Get-Device-Config-DO	
'FF 40'	Response-ALL-REF-AR-DO	
'FF 50'	Response-AR-DO	
'FF 70'	Response-ARAC-AID-DO	
'FF 7F'	Response-Device-Config-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 `DeviceAppID` of the REE device applications (certificate hashes) are fake values like 111…, 222…, 333…, etc.

Note: All these examples use a `DeviceAppID` based on a hash value. When using a `DeviceAppID` based on a Unique Identifier, the hash values in these examples shall be replaced by the unique identifier padded with 'FF'.

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)
globalPlatform(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
```

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

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

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

```
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 11 11
```

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

```
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)
    globalPlatform(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)
globalPlatform(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 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 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 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
```

C.4 Fourth Example

The setup is in this example is equal to the Third Example but with both DODFs provisioned:

Default AID	→ access allowed for 1 app (hash0, hash2-0)	→ conditions 1
AID1 = A0 00 00 01 51 01	→ access allowed for 1 app (hash1, hash2-1)	→ conditions 2
AID2 = A0 00 00 01 51 02	→ access allowed for 1 app (hash2, hash2-2) + 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(1), DODF(2)
  |-DODF(1) (5207)             --> reference EF ACMain(1)
  |-EF ACMain(1) (4200)        --> reference EF ACRules(1)
  |-EF ACRules(1) (4300)       --> reference EF ACConditions(1)...
  |-EF ACConditions1(1) (4380)
  |-EF ACConditions2(1) (4381)
  |-EF ACConditions3(1) (4382)
  |-EF ACConditions4(1) (4383)
  |-DODF(2) (5217)             --> reference EF ACMain(2)
  |-EF ACMain(2) (4210)        --> reference EF ACRules(2)
  |-EF ACRules(2) (4310)       --> reference EF ACConditions(2)...
  |-EF ACConditions1(2) (4390)
  |-EF ACConditions2(2) (4391)
  |-EF ACConditions3(2) (4392)
  |-EF ACConditions4(2) (4393)

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(1), DODF(2)
  |-DODF(1) (5207)             --> reference EF ACMain(1)
  |-EF ACMain(1) (4200)        --> reference EF ACRules(1)
  |-EF ACRules(1) (4300)       --> reference EF ACConditions(1)...
  |-EF ACConditions1(1) (4380)
  |-EF ACConditions2(1) (4381)
  |-EF ACConditions3(1) (4382)
  |-EF ACConditions4(1) (4383)
  |-DODF(2) (5217)             --> reference EF ACMain(2)
  |-EF ACMain(2) (4210)        --> reference EF ACRules(2)
  |-EF ACRules(2) (4310)       --> reference EF ACConditions(2)...
  |-EF ACConditions1(2) (4390)
  |-EF ACConditions2(2) (4391)
  |-EF ACConditions3(2) (4392)
  |-EF ACConditions4(2) (4393)
```

```
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 files 5207 and 5217.

binary coding:
A7 06 30 04 04 02 52 07
A7 06 30 04 04 02 52 17


DODF(1): 3F00/7F50/5207

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840)
globalPlatform(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(1) = 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(1): 3F00/7F50/4200

Refresh tag value is 01 02 03 04 05 06 07 08
path to EF ACRules(1) = 4300

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

```
EF ACRules(1): 3F00/7F50/4300

Default AID --> EFConditions1(1) 4380 --> access allowed for 1 app (h0)
AID1        --> EFConditions2(1) 4381 --> access allowed for 1 app (h1)
AID2        --> EFConditions3(1) 4382 --> access allowed for 1 app (h2)...
AID3        --> EFConditions4(1) 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 ACConditions1(1): 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 00 00


EF ACConditions2(1): 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 11 11


EF ACConditions3(1): 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 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 ACConditions4(1): 3F00/7F50/4383 (access allowed for all apps)

binary coding:
30 00
```

```
DODF(2): 3F00/7F50/5217

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840)
globalPlatform(114283) device(200) seAccessControl(1)
accessControlMainFileV2 (2)}
http://www.oid-info.com/get/1.2.840.114283
==> HEX encoding = 2A 86 48 86 FC 6B 81 48 01 02

application name = "GP SE Acc Ctl" (example)
path to EF ACMain(2) = 4210

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 02 30 04 04 02 42 10


EF ACMain(2): 3F00/7F50/4210

Refresh tag value is 01 02 03 04 05 06 07 08
path to EF ACRules(2) = 4310

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


EF ACRules(2): 3F00/7F50/4310

Default AID --> EFConditions1(2) 4390 --> acc. allowed for 1 app (h2-0)
AID1        --> EFConditions2(2) 4391 --> acc. allowed for 1 app (h2-1)
AID2        --> EFConditions3(2) 4392 --> acc. allowed for 1 app (h2-2)...
AID3        --> EFConditions4(2) 4393 --> access allowed for all apps

binary coding:
30 08  81 00                          30 04 04 02 43 90
30 10  A0 08 04 06 A0 00 00 01 51 01  30 04 04 02 43 91
30 10  A0 08 04 06 A0 00 00 01 51 02  30 04 04 02 43 92
30 10  A0 08 04 06 A0 00 00 01 51 03  30 04 04 02 43 93


EF ACConditions1(2): 3F00/7F50/4390 (access allowed for 1 app)

Hash2-0 has the value 000…

binary coding:
30 16 04 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00


EF ACConditions2(2): 3F00/7F50/4391 (access allowed for 1 app)

Hash2-1 has the value 111…
```

```
binary coding:
30 16 04 20 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11
```

```
EF ACConditions3(2): 3F00/7F50/4392 (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 20 22 22 22 22 22 22 22 22 22 22 22 22 22 22 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 ACConditions4(2): 3F00/7F50/4393 (access allowed for all apps)

binary coding:
30 00
```

Annex D Rules Conflict Management Examples

Table D-2 shows some examples of how the ACE shall apply combinations of the rules provided by the ARA-M.

Some examples in Table D-2 show multiple rules with the same target (AID, `DeviceAppID`).

There will never be multiple rules for the same target in the same ARA: If a `Command-Store-REF-AR-DO` is submitted with a target that exactly matches the target of a rule that already exists in the current ARA, the `Command-Store-REF-AR-DO` will merge or overwrite the preceding stored rule, as described in section 5.1.1.1.

Similarly, if a `Command-Store-REF-AR-DO` is submitted with a target that exactly matches the target of a rule that already exists in an ARA other than the current ARA, the `Command-Store-REF-AR-DO` will not store the rule, but instead will return error SW '6A 89', as described in section 5.1.1.1.

(A partial target match does not prevent a rule from being stored, as discussed in section 5.1.1.1.)

Nonetheless, rules with the same target can exist in different ARAs (possible scenarios are discussed in section 3.4) and conflict resolution occurs when the rules are fetched.

Table D-1: Terms Used in Table D-2

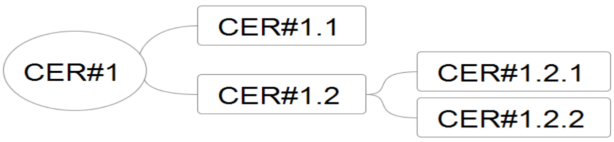
Term	Meaning in Table D-2
Rxx	A certain rule defined in the Secure Element, regardless of where it is stored.
DID#y	`DeviceAppID` contains one of following values: • Hash of the certificate of the Application Provider • Unique identifier (this could be a TA UUID) padded with 'FF' in order to provide a length of 20 (or 32) bytes
DID#x.y	A certain certificate (CER) within a certificate chain of a REE application when hash is used as the `DeviceAppID`. 
DID#y*	A certain certificate or any child certificate thereof; applies only if certificate chains are used.
All	Indicates a generic reference which applies to all device applications which are not covered by a specific reference.
Ref	Reference to an access rule (AR); either one that refers to a specific `DeviceAppID` (`DeviceAppID-Ref`) or the generic 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 Applet are not defined. This means the access rules contain no reference – neither “aid”, “default”, nor “others” – to the corresponding Applet. Thus, no access conditions are assigned to this Applet and all access is denied.

Table D-2: Rules Conflict Management

	Applet #1	Applet #2	Other AIDs (generic rule)	Access Result for the Applets
1	R0: (DID#1-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* are granted access to Applet #1. • Access to any other Applet is denied.
2	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (DID#2-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* or with DID#2* are granted access to Applet #1. • Access to any other Applet is denied.
3	R0: (DID#1-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Access to any Applet is denied.
4	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (DID#1-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Access to Applet #1 is denied because NEVER rule has higher priority. • Access to any other Applet is denied.
5	R0: (DID#1-Ref)-AR-APDU-Filter	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* are granted access to Applet #1, but only with an APDU matching the APDU filter. • Access to any other Applet is denied.
6	R0: (DID#2-Ref)-AR-APDU-NEVER	R0: (DID#2-Ref)-AR-APDU-Filter	R0: (DID#2-Ref)-AR-APDU-ALWAYS	• All device applications are denied access to SE App #1. • Device applications signed with DID#2* are granted access to Applet #2, but only with an APDU matching the APDU filter. • Device applications signed with DID#2* are granted access to any other Applet.
7	R0: (DID#1-Ref)-AR-APDU-NEVER R1: (DID#2-Ref)-AR-APDU-Filter	NOT EXIST	NOT EXIST	• Device applications signed with DID#2* are granted access to Applet #1, but only with an APDU matching the APDU filter. • Access to any other Applet is denied.

	Applet #1	Applet #2	Other AIDs (generic rule)	Access Result for the Applets
8	R0: (DID#1.2-Ref)-AR-APDU-Filter R1: (DID#1.2.1-Ref)-AR-APDU-Filter	NOT EXIST	NOT EXIST	• Device applications signed with DID#1.2.1* are granted access to Applet #1, but only with an APDU matching the R1 APDU filter. • Device applications signed with DID#1.2* are granted access to Applet #1, but only with an APDU matching the R0 APDU filter. • Access to any other Applet is denied.
9	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (ALL-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* have an access to Applet #1. • Access to any other Applet is denied
10	R0: (DID#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* have no access to Applet #1. • All other device applications don't have access to Applet #1 because there is a specific rule protecting access to Applet #1.
11	R0: (DID#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-NEVER	NOT EXIST	NOT EXIST	• Access to all Applets is denied.
12	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (ALL-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* have access to Applet #1. • All other device applications don't have access to any Applet (including Applet #1).
13	NOT EXIST	NOT EXIST	R0: (ALL-Ref)-AR-APDU-ALWAYS	• All device applications have access to all Applets.
14	R0: (ALL-Ref)-AR-APDU-NEVER	R0: (ALL-Ref)-AR-APDU-NEVER	R0: (ALL-Ref)-AR-APDU-ALWAYS	• Access to Applet #1 and Applet #2 is denied for all device applications. • All other Applets can be accessed by all device applications.

	Applet #1	Applet #2	Other AIDs (generic rule)	Access Result for the Applets
15	R0: (ALL-Ref)-AR-APDU-ALWAYS	R0: (ALL-Ref)-AR-APDU-ALWAYS	R0: (ALL-Ref)-AR-APDU-NEVER	• Access to Applet #1 and Applet #2 is granted for all device applications. • All other Applets cannot be accessed by any device application.
16	R0: (DID#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-ALWAYS	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (ALL-Ref)-AR-APDU-NEVER	R0: (DID#1-Ref)-AR-APDU-NEVER R1: (ALL-Ref)-AR-APDU-ALWAYS	• A device application signed with DID#1 or any other certificate has no access to Applet #1. • A device application signed with DID#1 is allowed to access Applet #2. • All other device applications have no access to Applet #2. • Access to all other Applets is denied for all device applications because there is a rule associating all other Applets with a specific device application.
17	R0: (DID#1-Ref)-AR-APDU-NEVER R1: (DID#2-Ref)-AR-APDU-NEVER R2: (DID#3-Ref)-AR-APDU-NEVER R3: (ALL-Ref)-AR-APDU-ALWAYS	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (DID#2-Ref)-AR-APDU-NEVER R2: (DID#3-Ref)-AR-APDU-NEVER R3: (ALL-Ref)-AR-APDU-ALWAYS	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (DID#2-Ref)-AR-APDU- ALWAYS R2: (DID#3-Ref)-AR-APDU- ALWAYS R3: (ALL-Ref)-AR-APDU-NEVER	• Applet #1 can never be accessed. • Applet #2 can only be accessed by device applications signed with DID#1*. • All other Applets can be accessed by all device applications that are signed with DID#1, DID#2, DID#3.

	Applet #1	Applet #2	Other AIDs (generic rule)	Access Result for the Applets
18	R0: (DID#1-Ref)-AR-APDU-ALWAYS R1: (DID#2-Ref)-AR-APDU-ALWAYS R2: (DID#3-Ref)-AR-APDU-ALWAYS	R0: (DID#4-Ref)-AR-APDU-ALWAYS R1: (DID#5-Ref)-AR-APDU-ALWAYS R2: (DID#6-Ref)-AR-APDU-ALWAYS R3: (ALL-Ref)-AR-APDU-NEVER	R0: (DID#7-Ref)-AR-APDU-ALWAYS R1: (DID#8-Ref)-AR-APDU-ALWAYS R2: (DID#9-Ref)-AR-APDU-ALWAYS R3: (ALL-Ref)-AR-APDU-NEVER	• Applet #1 can be accessed by all device applications that are signed with DID#1, DID#2, DID#3. • Applet #2 can be accessed by all device applications that are signed with DID#4, DID#5, DID#6. • All other Applets can only be accessed by device applications signed with DID#7, DID#8, DID#9.
19	NOT EXIST	NOT EXIST	NOT EXIST	• Access to any Applet is denied.
NFC Events Management				
20	R0: (DID#1-Ref)-AR-NFC-ALWAYS	R0: (DID#1-Ref)-AR-APDU-Filter	NOT EXIST	• Device applications signed with DID#1* can receive NFC events from Applet #1. • Device applications signed with DID#1* are granted access to Applet #2, but only with an APDU matching the APDU filter. • Device applications signed with DID#1 can receive NFC events from Applet #2. • Access to any other Applet is denied.
21	R0: (DID#1-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* can receive NFC events from Applet #1, because this application is allowed APDU access to Applet #1. • Other device applications cannot receive NFC events.
22	R0: (DID#1-Ref)-AR-NFC-ALWAYS R1: (DID#1-Ref)-AR-NFC-NEVER	NOT EXIST	NOT EXIST	• NFC events cannot be received by any device application.

	Applet #1	Applet #2	Other AIDs (generic rule)	Access Result for the Applets
23	R0: (DID#1-Ref)-AR-NFC-NEVER R1: (DID#1-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• NFC events cannot be received by any device application. • Device applications signed with DID#1* are granted APDU access to Applet #1. • Access to any other Applet is denied.
24	R0: (DID#1-Ref)-AR-NFC-ALWAYS R1: (DID#1-Ref)-AR-APDU-ALWAYS	NOT EXIST	NOT EXIST	• Device applications signed with DID#1* can receive NFC events. • Other device applications cannot receive NFC events. • Device applications signed with DID#1* are granted APDU access to Applet #1. • Access to any other Applet is denied.
25	NOT EXIST	NOT EXIST	NOT EXIST	• Access to any Applet 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 ACE (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

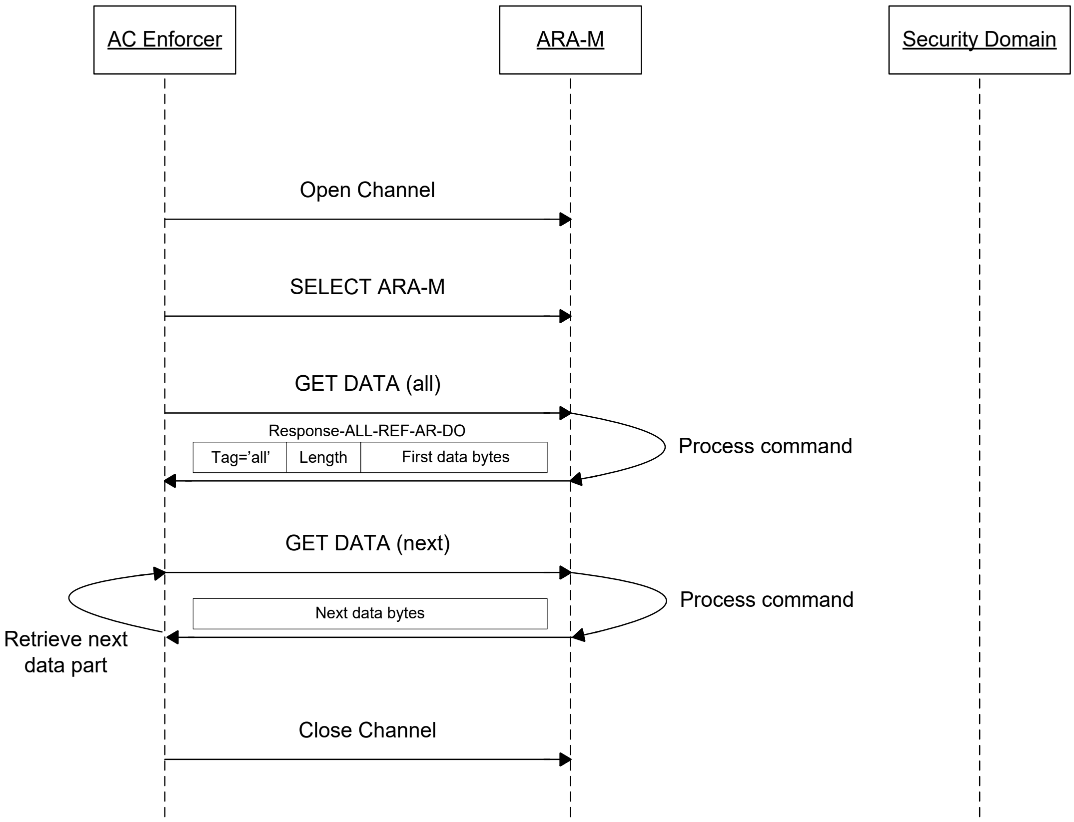


Figure E-2 shows how the ACE (Off-card Entity) shall retrieve specific access rules from the ARA-M using GET DATA [Specific] (deprecated), as described in section 4.2.2. (See Figure E-3 for a more complete version of the same process.)

Note: Figure E-2, Figure E-3, and Figure E-4 demonstrate access rule retrieval based on GET DATA [Specific], which is deprecated.

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

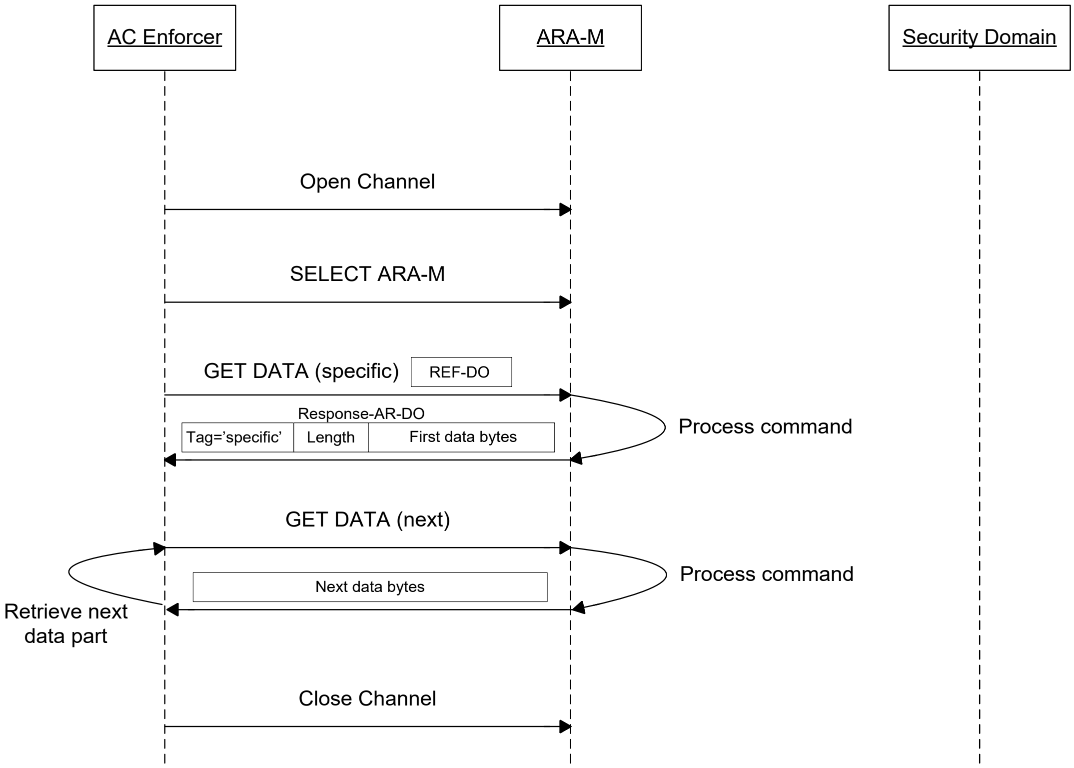
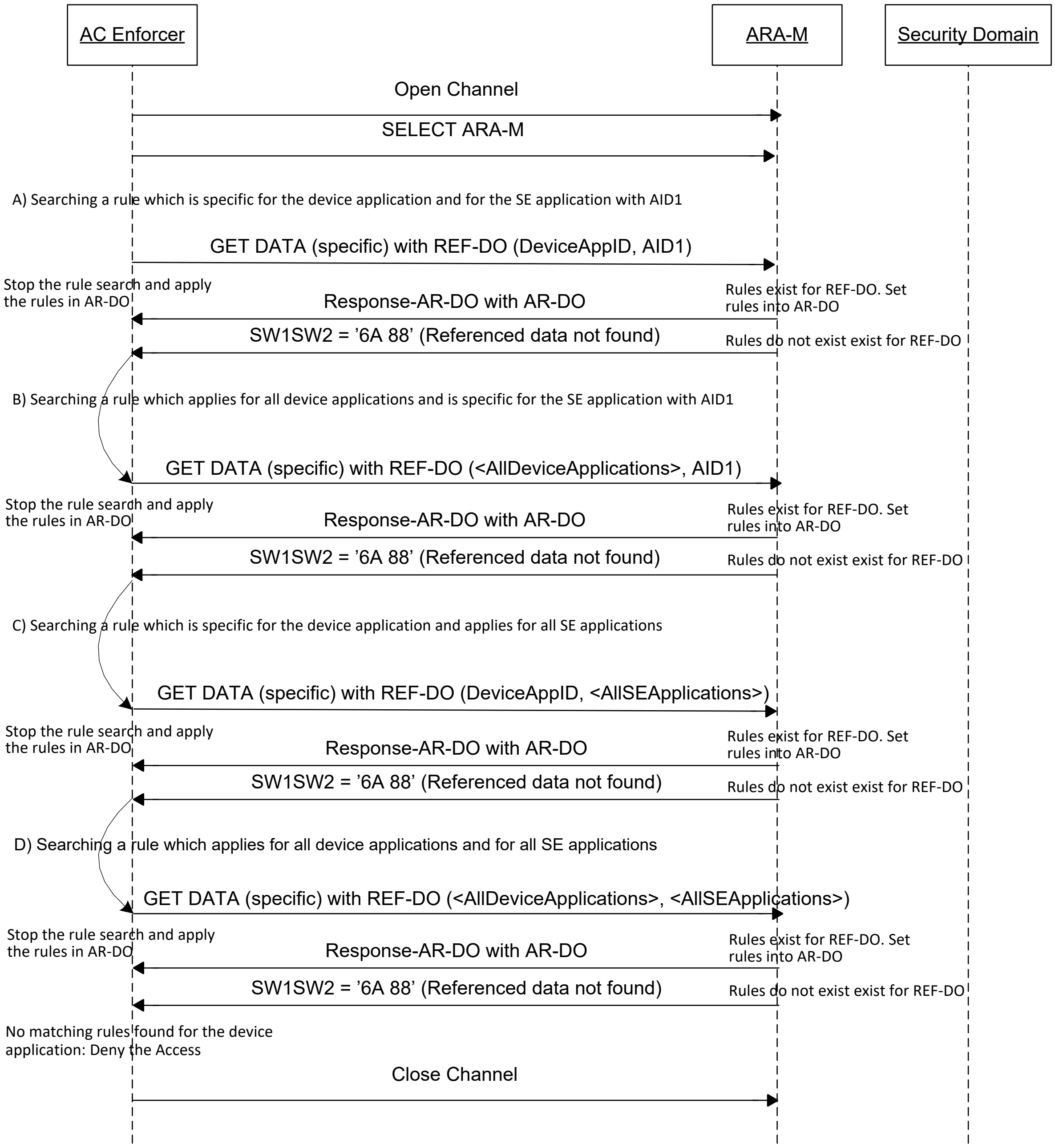


Figure E-3 shows how the ACE (Off-card Entity) shall retrieve specific access rules from the ARA-M with the whole retrieval sequence, as described in section 4.2.2.

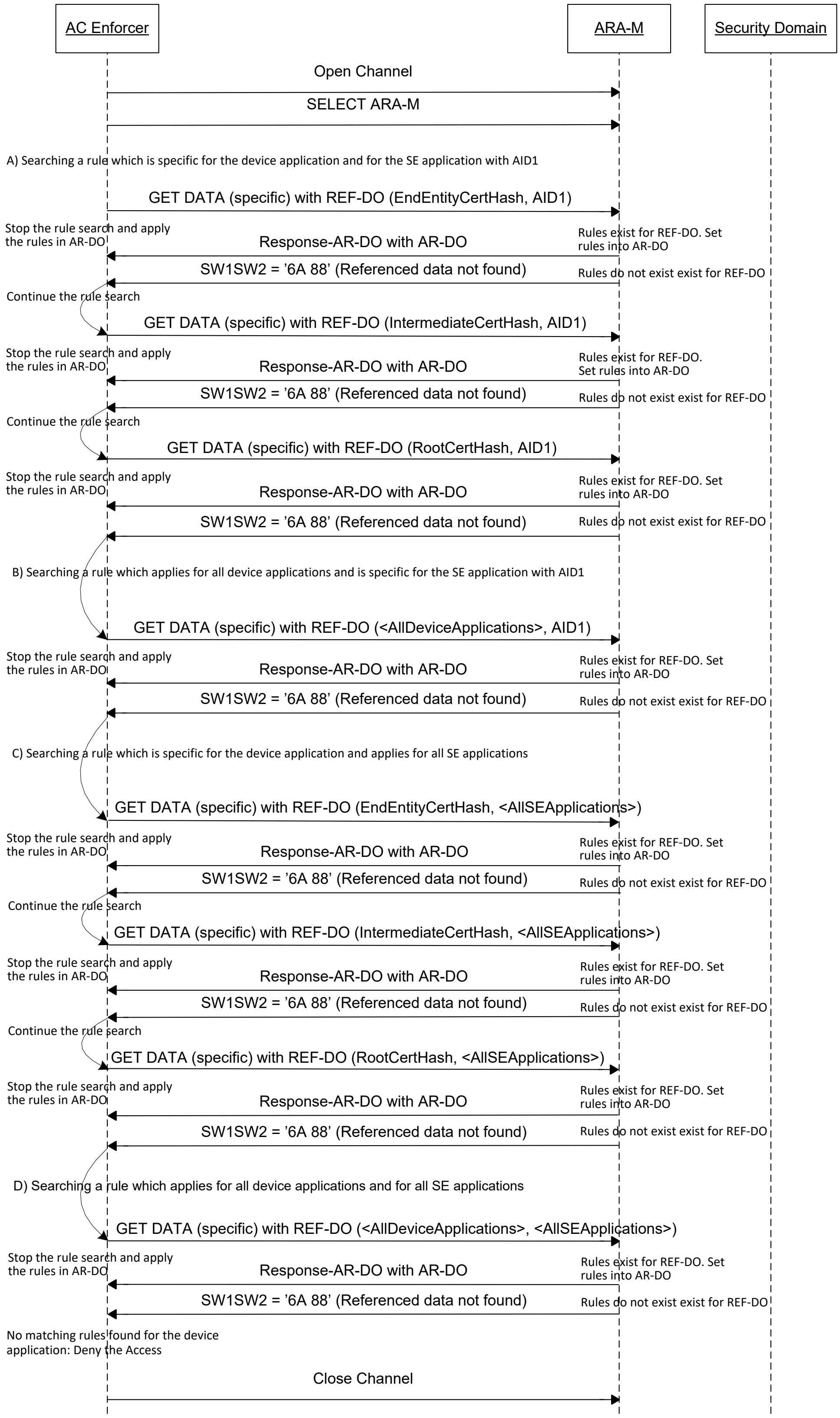
Figure E-3: Access Rule Retrieval Sequence (Deprecated)



Note for Figure E-3 and Figure E-4: If a rule is not found, then the ARA-M returns either an SW '6A 88' (Referenced data not found) or an SW '90 00' with response data (`Response-AR-DO` or a `Response-ALL-REF-AR-DO` of null length ('FF5000' or 'FF4000')).

Figure E-4 shows how the ACE (Off-card Entity) shall retrieve specific access rules from the ARA-M when the device application has a certificate chain.

2262 **Figure E-4: Chained Certificate Querying (Deprecated)**



2263

E.2 Remote Interface Based on RAM APDU Flow

This section describes different scenarios for remote management of access rules depending on the available OTA channels to access the Secure Element:

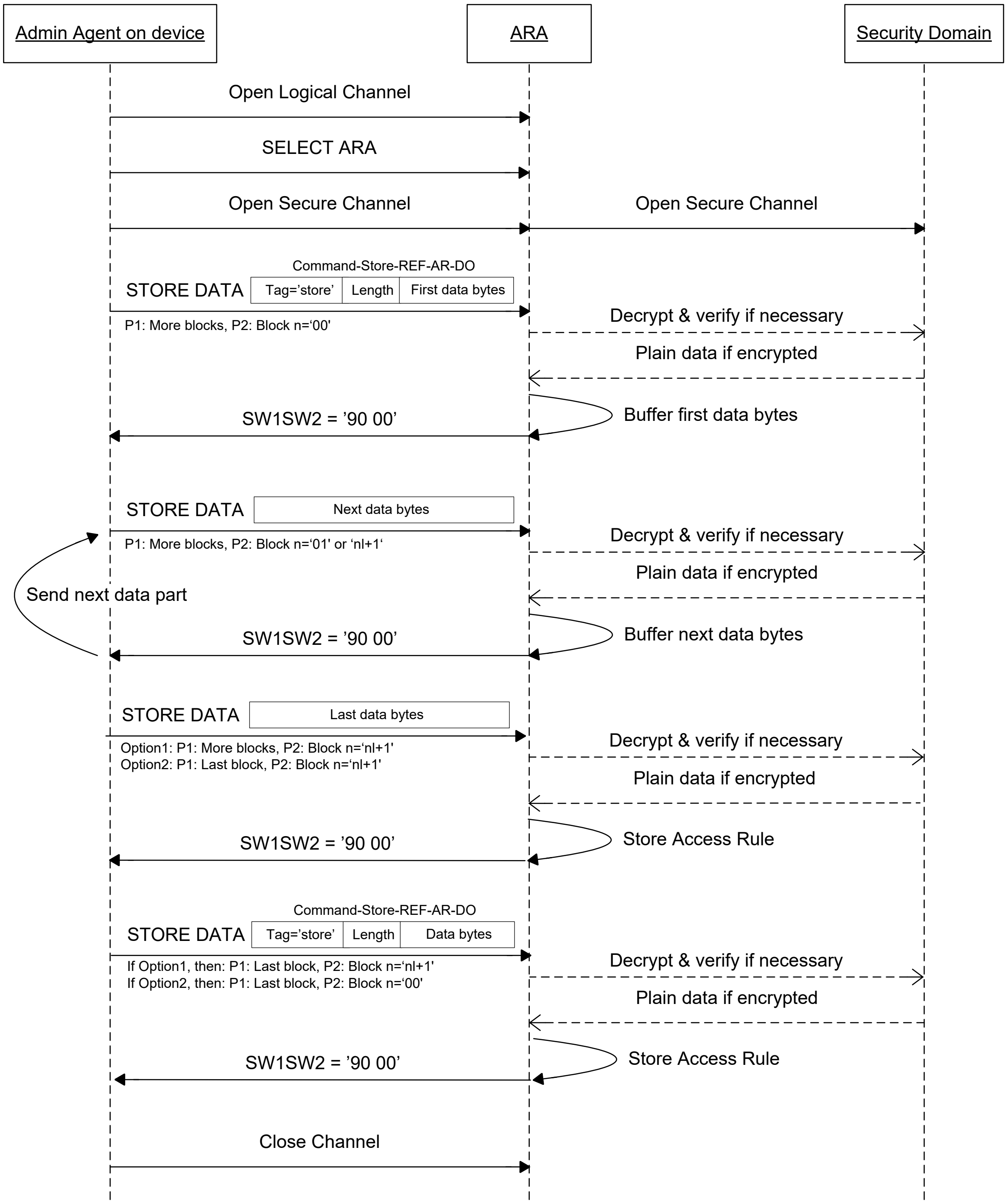
- Remote management using an Admin Agent on device
- Remote management using a direct OTA connection to the Secure Element
- Remote management using a direct OTA connection to the Secure Element, with limited buffers

E.2.1 Remote Management with Admin Agent on Device

This section illustrates remote management scenarios using an Admin Agent on device as described in [GP SE OTA]: Between the Admin Agent and the Secure Element the communication is managed by SCP02 or SCP03 by sending the commands either directly to the ARA or to its SD as defined in [GP Card Spec].

Figure E-5 shows how a remote administration server can store access rule data with an Admin Agent on device to the ARA (ARA-M or ARA-C) by sending a STORE DATA (`Command-Store-REF-AR-DO`) command directly to the ARA. This figure shows the storage of two access rules. The first rule is stored by using several STORE DATA commands and the second access rule is stored by using only one STORE DATA command.

Figure E-5: Provisioning Directly to ARA with Admin Agent on Device



Note: 'nl' = block number of the last command

The deletion of access rule data can be performed in the same way with a STORE DATA (Command-Delete). Since the value of Command-Delete is always short, this command can always be transferred with one APDU. This figure shows the deletion of two access rules.

Figure E-6: Deletion Directly to ARA with Admin Agent on Device

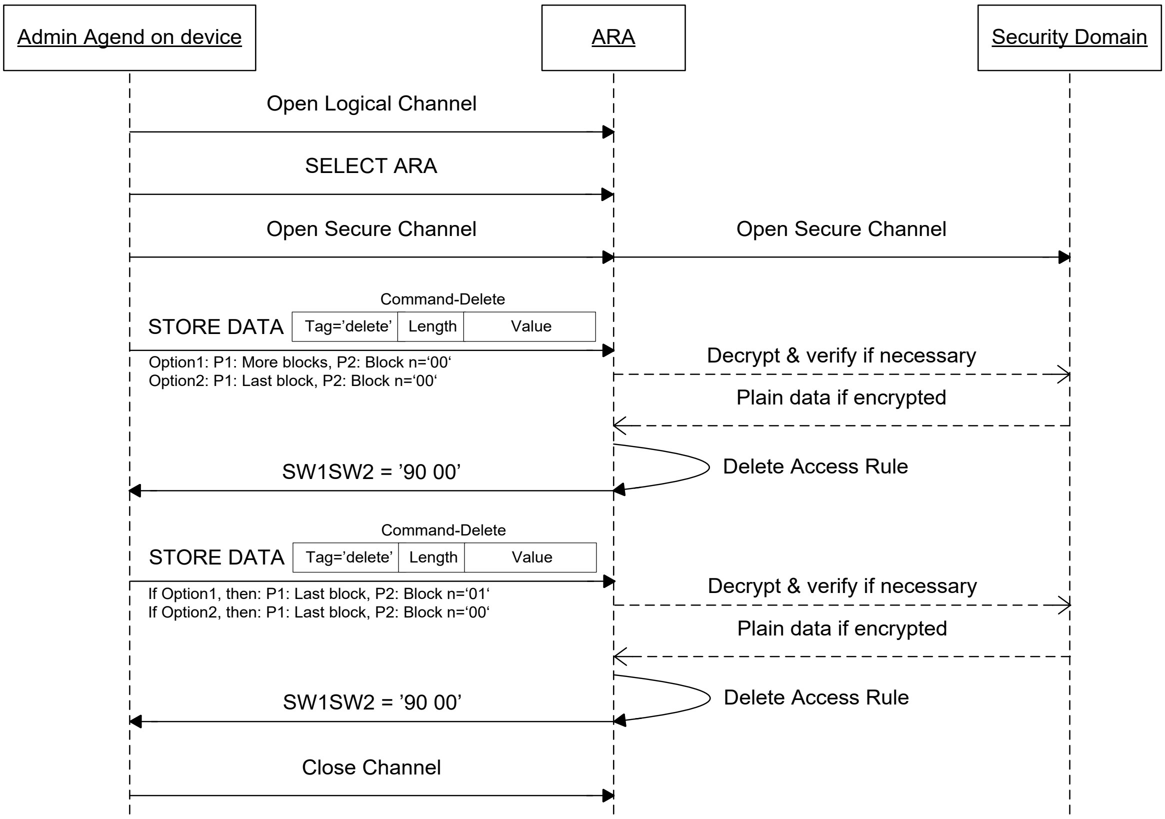
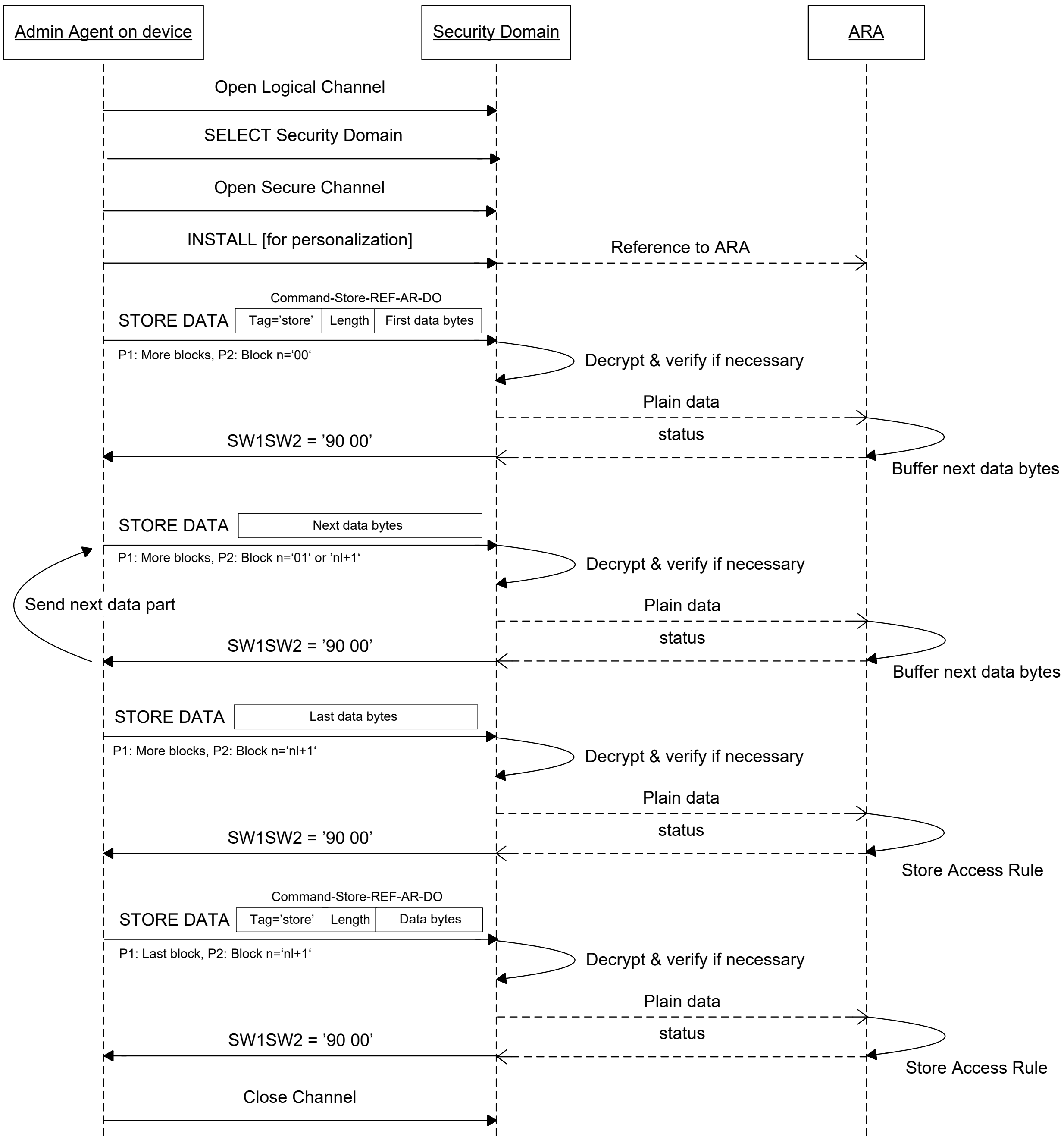


Figure E-7 shows how a remote administration server can store access rule data with an Admin Agent on device to the ARA (ARA-M or ARA-C) by sending to the associated Security Domain an INSTALL [for personalization] command and then a STORE DATA (`Command-Store-REF-AR-DO`) command. This figure shows the storage of two access rules. The first access rule is stored by using several STORE DATA commands and the second access rule is stored by using only one STORE DATA command.

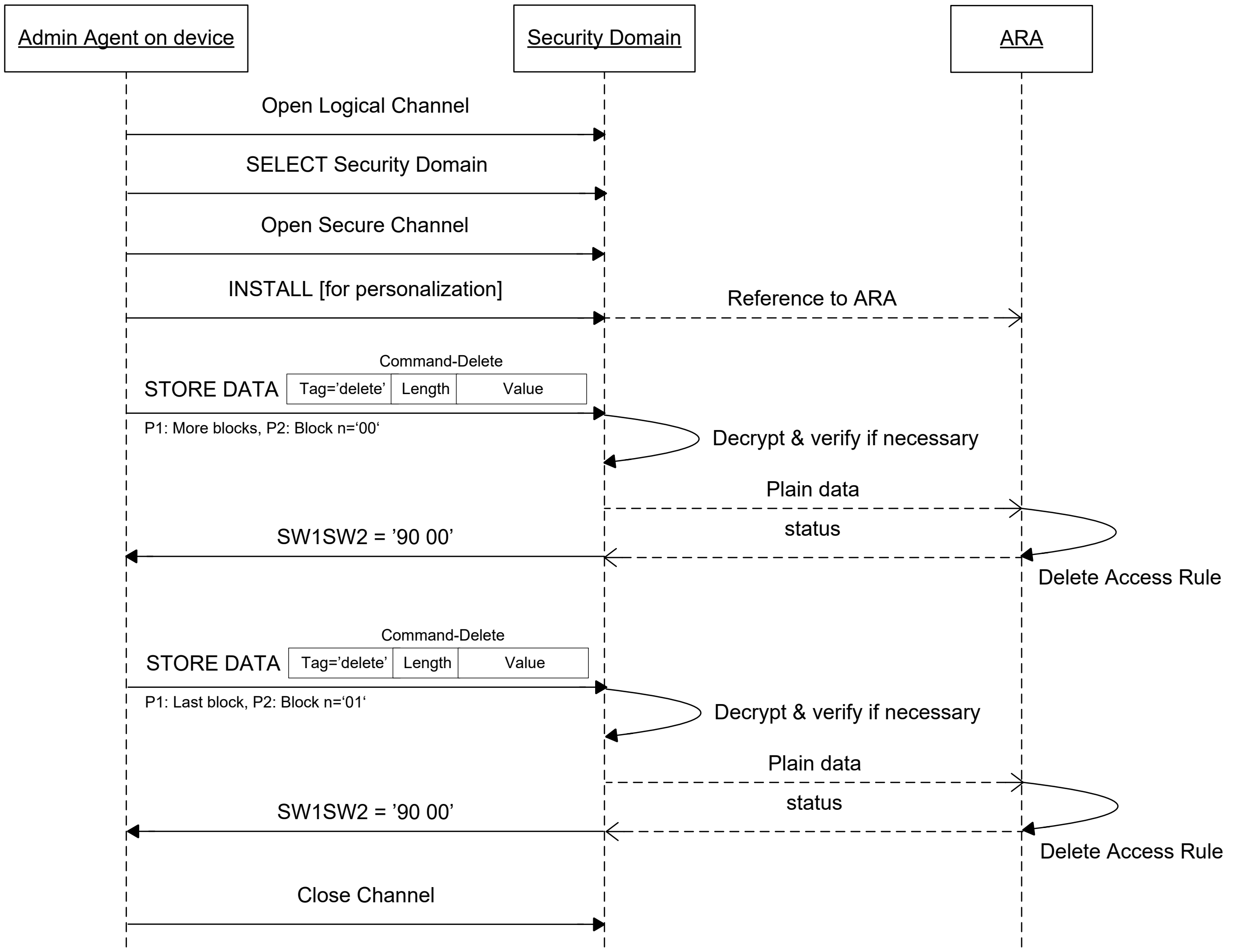
Figure E-7: Provisioning through Security Domain with Admin Agent on Device



Note: 'nl' = block number of the last command

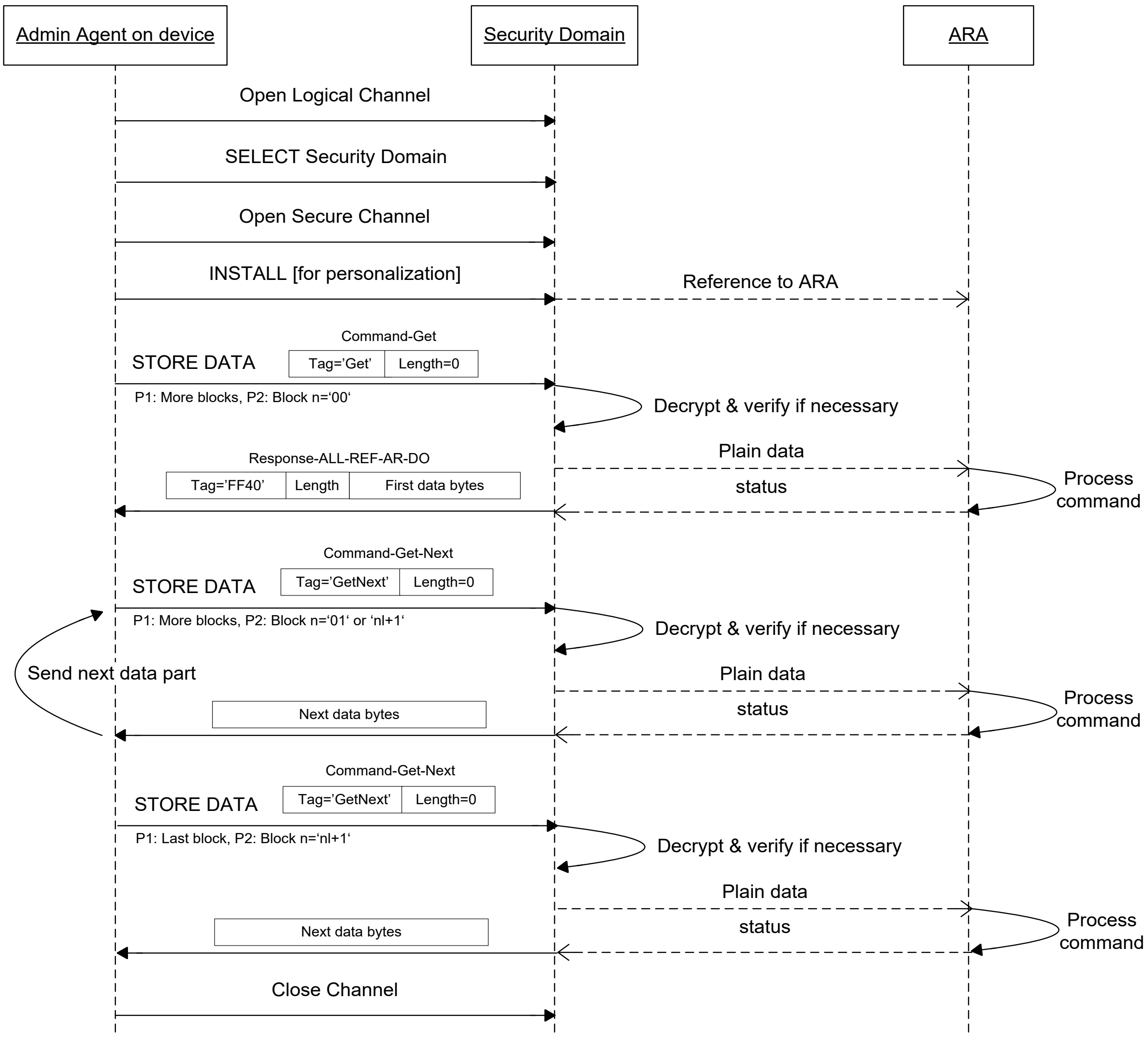
The deletion of access rule data can be performed in the same way with a STORE DATA (Command-Delete). Since the value of Command-Delete is always short, this command can always be transferred with one APDU. This figure shows the deletion of two access rules.

Figure E-8: Deletion through Security Domain with Admin Agent on Device



The retrieval of access rule data can be performed in the same way with a STORE DATA (`Command-Get`) or a STORE DATA (`Command-Get-All`). When the requested rules couldn't be fetched in response to this first command, a command STORE DATA (`Command-Get-Next`) can be performed to retrieve the remaining data.

Figure E-9: Rules Retrieval through Security Domain with Admin Agent on Device



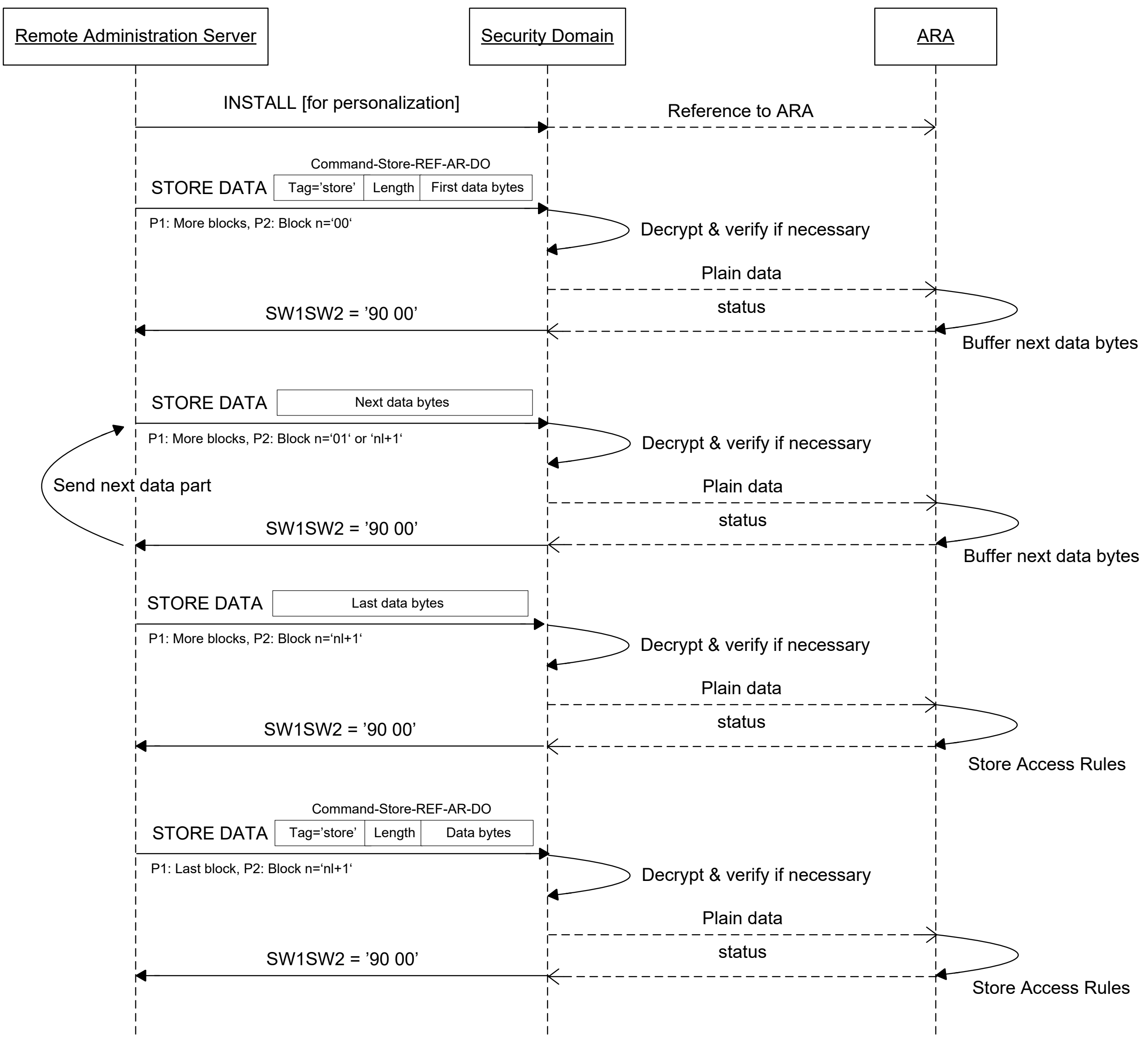
Note: 'nl' = block number of the last command

E.2.2 Remote Management with Direct OTA Connection

This section illustrates remote management scenarios using a direct OTA connection between the remote administration server and the Secure Element based on SCP80 or SCP81.

Figure E-10 shows how a 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-REF-AR-DO`) command to the associated SD. This figure shows the storage of two access rules. The first access rule is stored by using several STORE DATA commands and the second access rule is stored by using only one STORE DATA command.

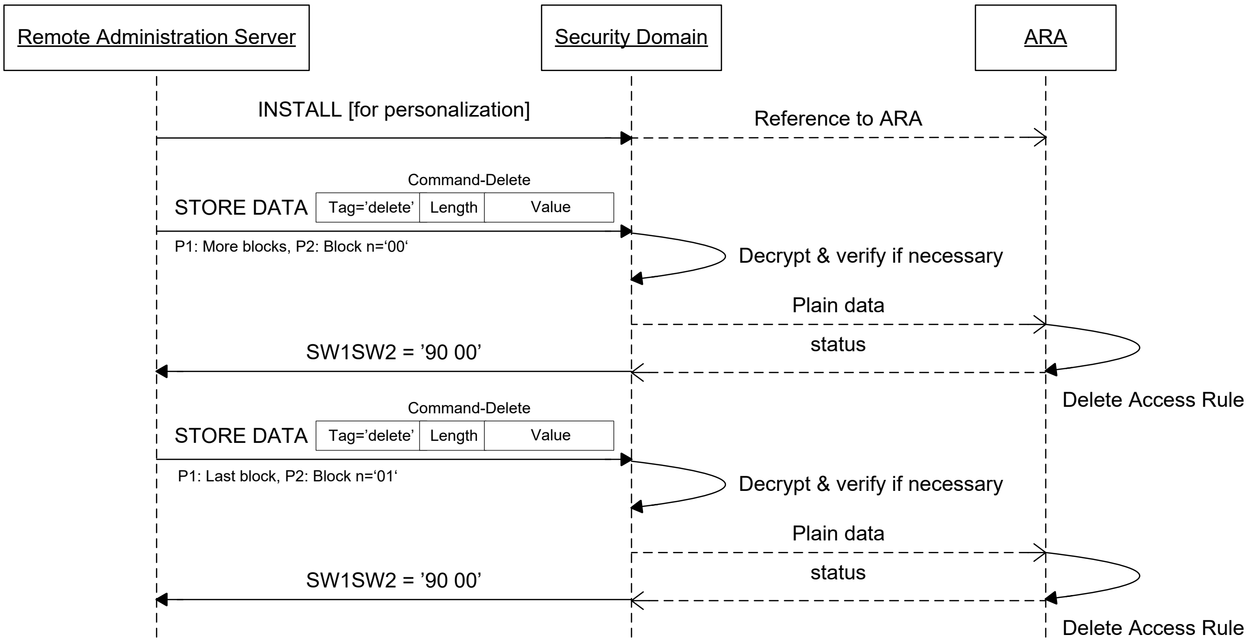
Figure E-10: Provisioning through Security Domain Using Direct OTA Connection



Note: 'nl' = block number of the last command

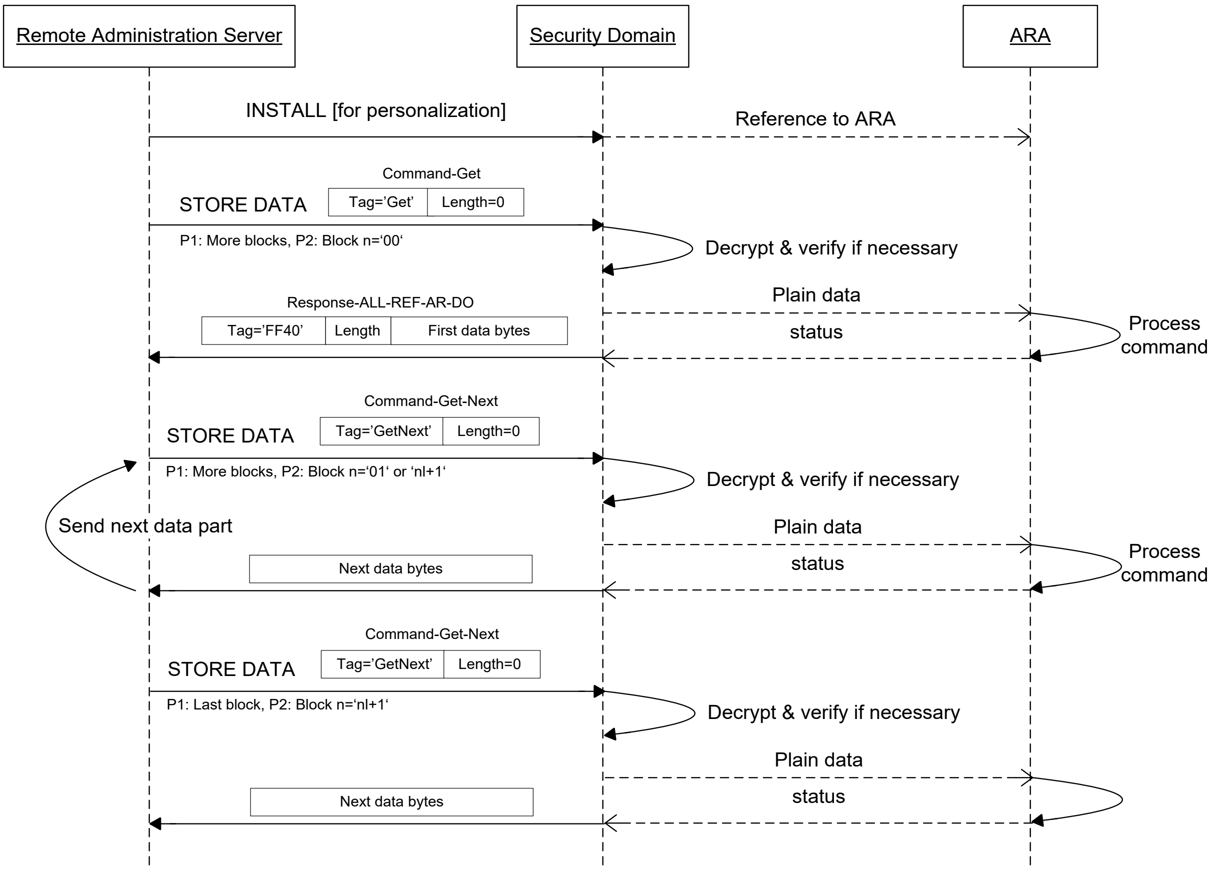
The deletion of access rule data can be performed in the same way with a STORE DATA (Command-Delete). Since the value of Command-Delete is always short this command can always be transferred with one APDU. This figure shows the deletion of two access rules.

Figure E-11: Deletion through Security Domain Using Direct OTA Connection



The retrieval of access rule data can be performed in the same way with a STORE DATA (`Command-Get`) or a STORE DATA (`Command-Get-All`). When the requested rules couldn't be fetched in response to this first command, a command STORE DATA (`Command-Get-Next`) can be performed to retrieve the remaining data.

Figure E-12: Rules Retrieval through Security Domain Using Direct OTA Connection



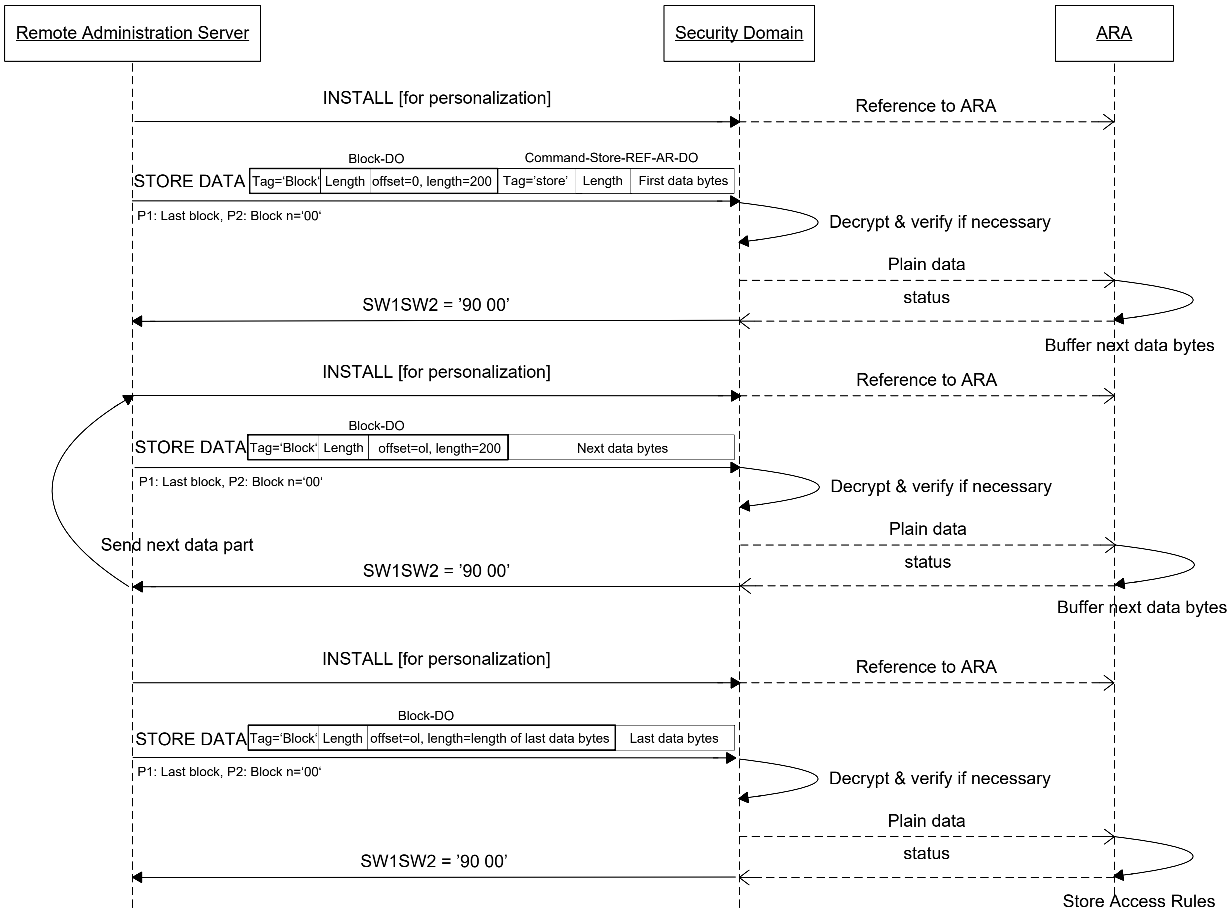
Note: 'nl' = block number of the last command

E.2.3 Remote Management with Direct OTA Connection and Limited Buffers

This section illustrates remote management scenarios using a direct OTA connection between the remote administration server and the Secure Element based on SCP80 with limited incoming and outgoing buffers.

The incoming buffer might be too small to store an access rule to an ARA within an INSTALL [for personalization] session. In this case the `Block-DO` may be used to store access rule data block by block over several INSTALL [for personalization] sessions.

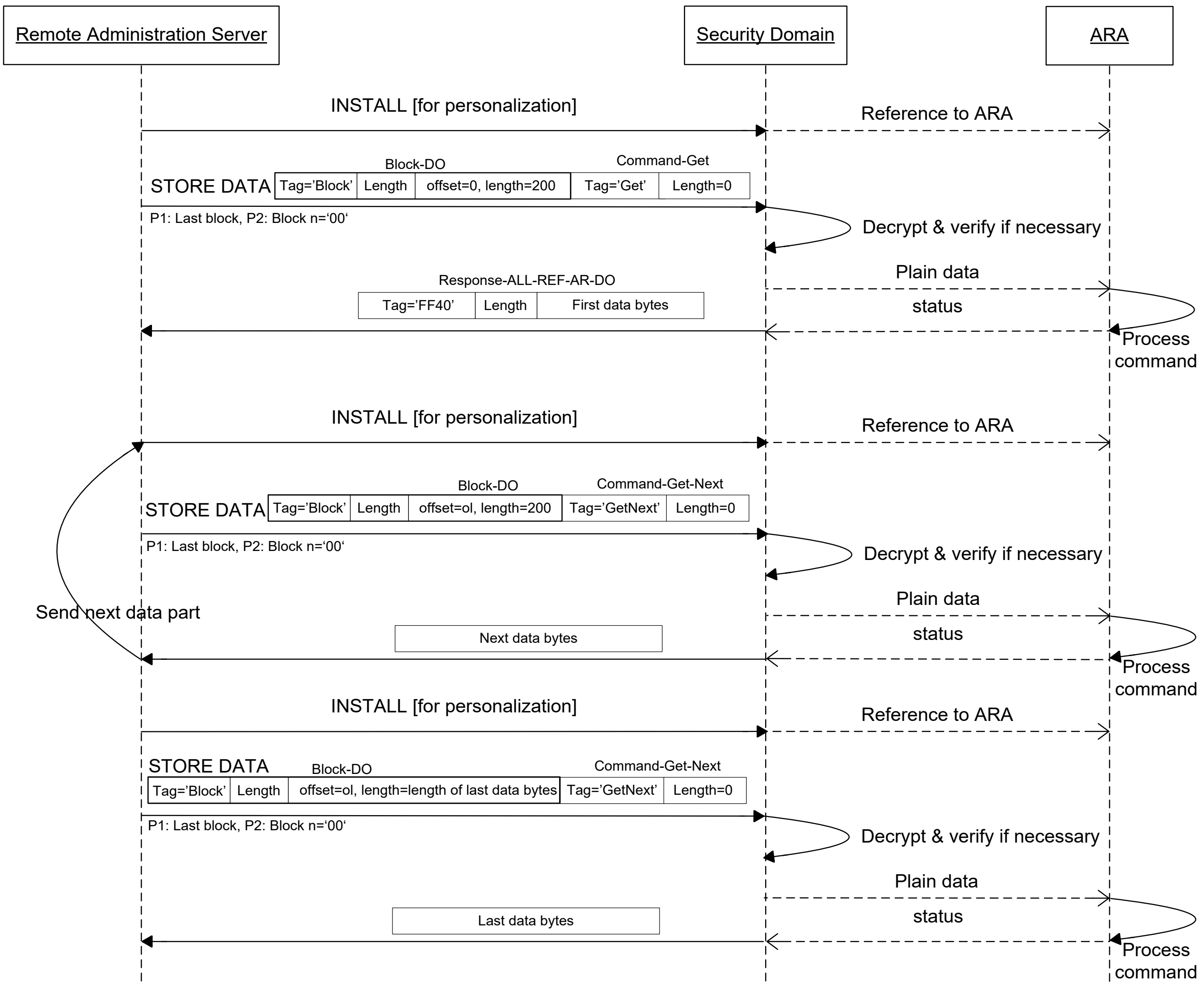
Figure E-13: Provisioning Using Direct OTA Connection with Limited Buffer



Note: 'ol' = offset + length of last command

The outgoing buffer might be too small to retrieve all access rules from an ARA within an INSTALL [for personalization] session. In this case the `Block-DO` may be used to retrieve access rule data block by block over several INSTALL [for personalization] sessions.

Figure E-14: Rules Retrieval Using Direct OTA Connection with Limited Buffer



Note: 'ol' = offset + length of last command

Annex F Migration Scenarios for the UICC

Prior to the publication of this specification, GSMA issued a document ([GSMA]) where access to applications on a UICC is defined by a set of elementary files. 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 ACE must support fallback to ARF for a UICC.

The following transition scenarios for UICCs can be expected:

- A UICC that is deployed with ARF 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.

Annex G ACCESS RULE INTERPRETATION

Table G-1 defines how ARA access rule policies shall be interpreted by the ACE, when using GET DATA [All] to retrieve the rules.

Table G-1: Interpretation of Access Rules Stored in the ARA

	Access Rule Policies	Granted Access
1	no APDU / no NFC policy	N/A
2	APDU (ALWAYS/APDUFilter) / no NFC	APDU (ALWAYS/APDUFilter) / NFC (ALWAYS)
3	no APDU / NFC (ALWAYS)	APDU (NEVER) / NFC (ALWAYS)
4	APDU (NEVER) / no NFC	APDU (NEVER) / NFC (NEVER)
5	no APDU / NFC (NEVER)	APDU (NEVER) / NFC (NEVER)
6	APDU (ALWAYS/APDUFilter) / NFC (ALWAYS)	APDU (ALWAYS/APDUFilter) / NFC (ALWAYS)
7	APDU (NEVER) / NFC (NEVER)	APDU (NEVER) / NFC (NEVER)
8	APDU (ALWAYS/APDUFilter) / NFC (NEVER)	APDU (ALWAYS/APDUFilter) / NFC (NEVER)
9	APDU (NEVER) / NFC (ALWAYS)	APDU (NEVER) / NFC (ALWAYS)

Table G-2 defines how ARF access rule policies shall be interpreted by the ACE when retrieving rules from the ARF or by the ARA-M having an ARF reading capability.

Table G-2: Interpretation of Access Rules Stored in ARF

	Access Rule Policies	Granted Access
1	no APDU / no NFC policy	APDU (ALWAYS) / NFC (ALWAYS)
2	APDU (ALWAYS/APDUFilter) / no NFC	APDU (ALWAYS/APDUFilter) / NFC (ALWAYS)
3	no APDU / NFC (ALWAYS)	APDU (NEVER) / NFC (ALWAYS)
4	APDU (NEVER) / no NFC	APDU (NEVER) / NFC (NEVER)
5	no APDU / NFC (NEVER)	APDU (NEVER) / NFC (NEVER)
6	APDU (ALWAYS/APDUFilter) / NFC (ALWAYS)	APDU (ALWAYS/APDUFilter) / NFC (ALWAYS)
7	APDU (NEVER) / NFC (NEVER)	APDU (NEVER) / NFC (NEVER)
8	APDU (ALWAYS/APDUFilter) / NFC (NEVER)	APDU (ALWAYS/APDUFilter) / NFC (NEVER)
9	APDU (NEVER) / NFC (ALWAYS)	APDU (NEVER) / NFC (ALWAYS)

Note: When several rules apply to the same access request, aggregation and conflict resolution shall be performed by the ACE, using the algorithm defined in section 3.4.1. However, that algorithm doesn't consider missing attributes. Rules shall be interpreted as defined in Table G-1 (rules read from ARA-M) or Table G-2 (rules read from ARF) only if attributes are missing from the result of the rule conflict resolution or combination.