



**IFM**  
**PROJECT**  
INTEROPERABLE FARE MANAGEMENT

# Generic common architecture, interfaces and security principles Strategy Paper

Deliverable 5.2  
June 2010

**Grant Agreement number:** IST-2007-214787  
**Project acronym:** IFM PROJECT  
**Project title:** INTEROPERABLE FARE MANAGEMENT PROJECT

**Funding Scheme:** Support Action  
**Project Coordinator:** John Graham Verity  
Head of Compliance  
ITSO Limited, United Kingdom

**Tel:** +44 121 634 3700  
**Fax:** +44 121 634 3737  
**E-mail:** [compliance@itso.org.uk](mailto:compliance@itso.org.uk)  
**Project website address:** <http://www.ifm-project.eu>

For further information please contact:

**Work package 5 leader**

**Main authors**

VDV-Kernapplikations GmbH & Co. KG

For further information on the IFM Project please contact:

**Coordination**

ITSO Ltd.

Phone +44 121 634 3700

Fax +44 121 634 3737

E-mail: [compliance@itso.org.uk](mailto:compliance@itso.org.uk)

**Secretariat**

TÜV Rheinland Consulting GmbH

Phone +49 221 806 4165

Fax +49 221 806 3496

E-mail: [izaskun.arenaza@de.tuv.com](mailto:izaskun.arenaza@de.tuv.com)

Visit the webpage [www.ifm-project.eu](http://www.ifm-project.eu)

## List of abbreviations

AFC	Automated Fare Collection
CALYPSO	Electronic Ticketing Standard for (microprocessor) contactless Smartcard, designed by a group of European transit operators
CBO	Central Back Office
CRL	Certificate Revocation Lists
EFM	Electronic Fare Management
IFM	Interoperable Fare Management
ISO	International Organization of Standardization
ITSO	Integrated Transport Smartcard Organisation; UK Standard for nationwide Interoperable Electronic Fare Management
MO	mobile operators
PT	public transport
PTO	public transport operator
SD	Security Domain
SE	Secure Element Secure microprocessor chip employed in user media to host applications.
TA	Transport Authority
TO	Transport operator
VDV-KA	VDV Core Application, German Standard for nationwide Interoperable Electronic Fare Management
VDV-KA KG	VDV-Kernapplikations GmbH & Co. KG

## Reference documents

- [R1] ISO/EN24014-1: 2007 - Public transport - Interoperable fare management system - Part 1: Architecture (IFMS)
- [R2] ISO/PrEN 24014-2: [Working draft] -Public Transport - Interoperable Fare Management System - Part 2: Recommended Working draft]
- [R3] IFM Project - Inventory of functions, organisational models and economic issues of existing IFM-Systems, Deliverable 4.1 - March 2009
- [R4] IFM Project – Report on the organisational structures and the differences of the existing IFM systems, Deliverable 4.2 - March 2009
- [R5] IFM Project – Report on the follow-up workshop to explain and disseminate the agreed Common Methodology for preparing a Trust Management Model, Deliverable 1.3 - August 2009
- [R6] IFM Project – Common requirements and recommendations on interoperable media and multi-application management, Deliverable 3.2, Version 2.4 - September 2009

1.	Introduction.....	7
2.	Generic System Model: Components and Interfaces.....	8
2.1.	User Medium.....	8
2.2.	SAM.....	8
2.3.	Generic Terminal.....	9
2.3.1.	Generic product and application retailer terminals.....	9
2.3.2.	Generic service operator terminal.....	9
2.4.	Generic systems.....	10
2.4.1.	Generic product retailer system.....	10
2.4.2.	Generic application retailer system.....	10
2.4.3.	Generic service operator system.....	11
2.4.4.	Generic product owner system.....	11
2.4.5.	Generic application owner system.....	11
2.4.6.	Generic revocation service system.....	12
2.5.	Interfaces between generic systems and terminals.....	12
2.5.1.	Interface SAM=ProductRetailerTerminal.....	12
2.5.2.	Interface SAM=ServiceOperatorTerminal.....	13
2.5.3.	Interface ProductOwner=ProductRetailer.....	13
2.5.4.	Interface ProductOwner=ServiceOperator.....	13
2.5.5.	Interface UserMedium=GenericTerminal.....	13
2.5.6.	Interface RevocationService=ProductRetailer.....	14
2.5.7.	Interface RevocationService=ServiceOperator.....	14
2.5.8.	Interface ApplicationOwner=ProductRetailer and ApplicationOwner=ApplicationRetailer.....	14
2.5.9.	Interface ApplicationOwner=ProductOwner.....	15
2.5.10.	Interface ApplicationOwner=ServiceOperator.....	15
2.5.11.	Interface RevocationService=ApplicationOwner.....	16
2.5.12.	Interface RevocationService=ProductOwner.....	16
2.5.13.	Interface ProductRetailer=ServiceOperator.....	17
2.5.14.	Interface ProductRetailer= ProductRetailer.....	17
3.	Generic system model: Elementary Processes.....	18
3.1.	Elementary processes associated with the sale and distribution of applications and entitlements.....	18
3.1.1.	Elementary processes for issuance of applications.....	18
3.1.2.	Elementary processes for issuance of entitlements.....	21
3.1.3.	Elementary processes for payment.....	22
3.1.4.	Elementary processes for return.....	24
3.1.5.	Elementary processes for reimbursement.....	25
3.1.6.	Elementary processes for modification.....	26
3.2.	Elementary processes for use of entitlements.....	29
3.2.1.	Elementary processes for accessing transport services with an entitlement....	29
3.3.	Elementary processes for revocation.....	30
3.3.1.	Elementary processes for revocation requests.....	31
3.3.2.	Elementary processes for revocation orders.....	34
3.3.3.	Elementary processes for handling of revocation lists.....	37
3.3.4.	Elementary processes for execution of a revocation.....	38
3.3.5.	Elementary processes for requesting cancellation of revocations.....	42
3.3.6.	Elementary processes for cancellation of revocations.....	44
3.3.7.	Elementary processes for unblocking.....	47
3.4.	Elementary processes for configuration.....	48
3.4.1.	EP_Configuration_Revocation_Service.....	48
3.4.2.	EP_Distribution_Keys.....	49
3.4.3.	EP_Distribution_SAMs.....	49

3.4.4.	EP_Distribution_Product_Module.....	49
3.4.5.	EP_Deactivation_Product_Module.....	50
4.	Interactive Processes.....	51
5.	Requirements for assuring Interoperability.....	53

# 1. Introduction

The objective of this paper is to analyse the scenarios identified in the IFM road map with the goal of identifying points that require the interaction of national systems and to establish requirements which will assure effective interaction at these points. In this way we can obtain a rough plan for implementing the different degrees of interoperability described in the road map.

This document has been produced as a generic model of how an EU-IFM Back Office could be structured to service the initial steps within the EU-IFM Roadmap. It does not specify the full requirements required for the use of the EU-IFM Application in a fully multi-application environment. New actors (such as the Trusted Service Manager in a Mobile environment) and their Use Cases are being developed as part of the work of ISO TC204 WG3 / CEN TC278 WG3 SG5 on ISO EN 24014, Interoperable Fare Management. This document is therefore proposed by the IFM Project as an input for future standardisation efforts.

The final conclusions of this standardisation work will need to be incorporated in IFM2 and its contractual relationships for the Back Office.

The approach described in this Deliverable should be developed as a matter of early importance by the new IFM Alliance as it moves towards the adoption of IFM2. This includes identifying the Back Office requirements for each stage of the IFM2 Roadmap and the implications for the contractual relationship that will need to be put in place to support it.

To this end we will propose a generic system model for an IFM resp. EFM system according to ISO 24014-1 at a technical system level which identifies generic components and interfaces as well as generic elementary processes and which will be used as a framework in the analysis.

A set of national IFMs can then be viewed as encapsulated in such a model IFM (having the same types of generic components, interfaces and processes as the national systems) and the analysis will strive to identify the processes at this encapsulated level which will require the interaction of the established national (sub-) systems.

In chapter 2 (“Generic System Model: Components and Interfaces”) we begin by proposing components and interfaces for the generic system model. The proposed elementary processes between these components (actors) are the subject of chapter 3 (“Generic system model: Elementary Processes”).

For the various scenarios of the IFM road map chapter 4 (“Interactive Processes”) will identify the generic processes for which interaction of the national IFMs will be necessary in order to achieve the degree of interoperability represented in the respective scenarios.

Finally, chapter 5 will look at specific aspects (“Requirements for assuring Interoperability”) of the identified “interactive” processes which should be “standardized” in order to assure interoperability.

## 2. Generic System Model: Components and Interfaces

The generic system model includes the basic functionalities of the various logical roles in the role model for EFM which may in fact be found in real systems and their physical components, in particular in the various types of terminals.

In accordance with the refinement of the original role model in CEN TC 278 the role “Collection & Clearing Operator” has been replaced by “Collection & Forwarding Operator”.

Due to the removal of the clearing function the resulting role no longer possesses any functionality specific to the application, which now consists of collecting and forwarding of data only. Consequently this role can be blended out in application oriented views, but must still be considered in network or multi-application views.

In the generic system model the job of clearing is assigned to each of the roles product owner, service operator and product retailer, each of them taking care of their own clearing (possibly outsourcing).

The basic roles of the model

- Customer,
- Product retailer,
- Service operator and
- Product owner

will be complimented by the additional, independent role Application Owner and its sub-roles security management, revocation service, certification and registrar.

In certain contexts in the following it may be useful to further specialise the role product retailer as primary product retailer responsible for issuing applications or entitlements directly to customers.

It is assumed in the following that, independent of the future physical implementation, each role operates a generic system and that the roles of product retailer and service operator, in particular, additionally operate a certain number of generic terminals. By connecting these generic systems and terminals via the identified interfaces we arrive at the generic system model. For each (unordered) pair of generic systems we consider exactly one “logical interface” without indicating which member of the pair actually provides or only uses the interface. These considerations will be further refined in connection with the consideration of the generic elementary processes later on.

### 2.1. *User Medium*

The user medium is the technical representation of the user in the generic system model.

The user medium carries applications and provides interfaces for the interaction with generic terminals. This generic logical interface will be denoted by UserMedium=Terminal.

### 2.2. *SAM*

The SAM is the technical representative of the security management within the systems and terminals of the different roles.

The SAM carries applications and provides interfaces for the interaction with generic terminals.

At the application level the generic logical interfaces to the product retailer and service  
Page 8 of 54

operator terminals will be denoted by  
SAM=ProductRetailerTerminal resp. SAM=ServiceOperatorTerminal.

### **2.3. Generic Terminal**

The generic terminal is the generalization of all types of terminals used in an EFM system and accesses the resources provided by the User Medium and the SAM via their interfaces. Specializations of the generic terminal are the

- generic application retailer terminal, denoted simply as ApplicationRetailerTerminal,
- generic product retailer terminal, denoted simply as ProductRetailerTerminal, and
- generic service operator terminal, denoted simply as ServiceOperatorTerminal.

In actual implementations some terminals may possess multiple functionalities.

#### **2.3.1. Generic product and application retailer terminals**

Specializations of the generic terminal are the generic product retailer terminal operated by the product retailer and generic application retailer terminal operated by the application retailer in order to realize the functions associated with the provision of applications, sales of tickets and information service.

In the case e.g. of a customer using his PC at home for these functions, the PC will be considered as a connection to the generic retailer terminal. The interfaces for data exchange between the generic retailer terminals and the generic system of the product or application retailer are denoted as

ApplicationRetailer=ApplicationRetailerTerminal  
ProductRetailer=ProductRetailerTerminal

From the view of the EFM / IFM application this is an internal interface and need not be considered further in detail.

#### **2.3.2. Generic service operator terminal**

A specialization of the generic terminal is the generic service operator terminal operated by the service operator in order to realize the functions associated with the validation and inspection of entitlements for travel including on-trip price calculations.

In the following specializations of the generic service operator terminal for validation and inspection may be considered.

The interface for data exchange between the generic service operator terminal and the generic system of the service operator is denoted as ServiceOperator=ServiceOperatorTerminal.

From the view of the EFM / IFM application this is an internal interface and need not be considered further in detail.

## 2.4. Generic systems

### 2.4.1. Generic product retailer system

The generic product retailer system generates and processes data exchanged with the other generic systems. The generic product retailer system can verify the authenticity of all transactions produced by entitlements that it has issued.

The generic product retailer system uses the following interfaces:

Interface	Data exchange with a generic
ApplicationOwner= ProductRetailer	application owner system
ProductOwner= ProductRetailer	product owner system
ProductRetailer= ServiceOperator	Service operator system
RevocationService= ProductRetailer	Revocation service system
ProductRetailer= ProductRetailer	Product retailer system (other)
ProductRetailer= ApplicationRetailer	Application retailer system

### 2.4.2. Generic application retailer system

The generic application retailer system generates and processes data exchanged with the other generic systems. The generic application retailer system registers the applications it issues and manages their life cycles.

The generic application retailer system uses the following interfaces:

Interface	Data exchange with a generic
ApplicationOwner= ApplicationRetailer	application owner system
ProductOwner= ApplicationRetailer	product owner system
ApplicationRetailer= ServiceOperator	Service operator system
RevocationService= ApplicationRetailer	Revocation service system
ProductRetailer= ApplicationRetailer	Product retailer system
ApplicationRetailer= ApplicationRetailer	Application retailer system (other)

### 2.4.3. Generic service operator system

The generic service operator system generates and processes data exchanged with the other generic systems.

The generic service operator system uses the following interfaces:

Interface	Data exchange with a generic
ApplicationOwner=ServiceOperator	application owner system
ProductOwner=ServiceOperator	product owner system
ProductRetailer=ServiceOperator	Product retailer system (other)
ApplicationRetailer=ServiceOperator	Application retailer system
RevocationService=ServiceOperator	Revocation service system

### 2.4.4. Generic product owner system

The generic product owner system administers the product definitions (templates and product modules), registers the business transactions of product retailers and service operators on the basis of products, determines prices for services according to the corresponding product definitions and controls the processing of business transactions with products retailers and service operators. The generic product owner system is responsible for the monitoring the completeness, correctness and authenticity of all transactions related to entitlements issued by retailers under contract with the product owner.

The generic product owner system uses the following interfaces:

Interface	Data exchange with a generic
ApplicationOwner=ProductOwner	application owner system
ProductOwner=ProductRetailer	Product retailer system (other)
ProductOwner=ApplicationRetailer	Application retailer system
RevocationService=ProductOwner	Revocation service system
ProductOwner=ServiceOperator	Service operator system

It may be useful to consider different specializations of the generic product owner system depending on what type of product(s) it is responsible for, e.g. electronic tickets, stored value, post-paid automatic fare entitlement etc.

### 2.4.5. Generic application owner system

The generic application owner system registers the issuance and deployment of SAMs, administers identifiers within the system (for organizations and components), registers the issuance of applications by application retailers, issues revocation orders and cancellations

(e.g. for SAMs, organizations and keys).

The generic application owner system uses the following interfaces:

Interface	Data exchange with a generic
ApplicationOwner= ProductRetailer	Product retailer system (other)
ApplicationOwner= ApplicationRetailer	Application retailer system
ApplicationOwner= ServiceOperator	Service operator system
RevocationService= ApplicationOwner	Revocation service system
ApplicationOwner= ProductOwner	Product owner system

## 2.4.6. Generic revocation service system

The generic revocation service system processes orders for revocations and cancellation of revocations and generates the corresponding revocation lists covering at least applications and entitlements on user media, SAMs, organizations and keys. It seems reasonable to generate three separate lists for these components: one a user media related list (applications and entitlements), a SAM and organization list and a key list.

The revocation service provides these lists to be picked up by the application owner, application retailer, product owner, product retailer and service operator.

A revocation confirmation is returned to the respective system after a revocation has been executed.

The generic revocation service system uses the following interfaces:

Interface	Data exchange with a generic
RevocationService= ProductRetailer	product retailer system
RevocationService= ApplicationRetailer	Application retailer system
RevocationService= ServiceOperator	Service operator system
RevocationService= ApplicationOwner	Application owner system
RevocationService= ProductOwner	Product owner system

## 2.5. Interfaces between generic systems and terminals

### 2.5.1. Interface SAM=ProductRetailerTerminal

The SAM provides an interface for the interaction with generic product retailer terminals.

At the application level the following processes between product retailer terminal and SAM are realized on this interface:

- Sales
- reimbursement
- Revocation
- unblocking
- service
- redemption
- issuance
- modification
- configuration

## 2.5.2. Interface SAM=ServiceOperatorTerminal

The SAM provides an interface for the interaction with generic service operator terminals. At the application level the following processes between service operator terminal and SAM are realized on this interface:

- usage (generation of corresponding transactions)
- validation
- inspection
- revocation
- configuration

## 2.5.3. Interface ProductOwner=ProductRetailer

Specializations of the interface between a generic product owner system and a generic product retailer system will be used for the following elementary processes of the product retailer:

Elementary processes associated with the sale and distribution of applications and entitlements (☒ 3.1)

Elementary processes for revocation requests (☒ 3.3.1)

Elementary processes for handling of revocation lists (☒ 3.3.3)

Elementary processes for execution of a revocation (☒ 3.3.4)

Elementary processes for requesting cancellation of revocations (☒ 3.3.5)

Elementary processes for unblocking (☒ 3.3.7)

Elementary processes for configuration (☒ 3.4)

## 2.5.4. Interface ProductOwner=ServiceOperator

Specializations of the interface between a generic product owner system and a generic service operator system will be used for the following elementary processes of the service operator:

- Elementary processes for use (☒ 3.2)
- Elementary processes for configuration (☒ 3.4)

## 2.5.5. Interface UserMedium=GenericTerminal Interface UserMedium=ProductRetailerTerminal

The user medium provides interfaces for the interaction with generic product retailer terminals.

At the application level the following processes (for provision of entitlements and service) are realized on this interface:

- Elementary processes associated with the sale and distribution of applications and entitlements (☒ 3.1)
- Elementary processes for issuance of entitlements (☒ 3.1.2)
- Elementary processes for execution of a revocation (☒ 3.3.4)
- Elementary processes for unblocking (☒ 3.3.7)

### **Interface UserMedium=ServiceOperatorTerminal**

The user medium provides interfaces for the interaction with generic service operator terminals.

At the application level the following processes are realized on this interface:

- Elementary processes for use (☒ 3.2),
- Elementary processes for execution of a revocation (☒ 3.3.4).

### **2.5.6. Interface RevocationService=ProductRetailer**

Specializations of the interface between the generic revocation service system and a generic product retailer system will be used for the following elementary processes of the product retailer:

- Elementary processes for revocation (☒ 3.3.2)
- Elementary processes for handling of revocation lists (☒ 3.3.3)
- Elementary processes for execution of a revocation (☒ 3.3.4)
- Elementary processes for cancellation of revocations (☒ 3.3.6).

### **2.5.7. Interface RevocationService=ServiceOperator**

Specializations of the interface between the generic revocation service system and a generic service operator system will be used for the following elementary processes of the service operator:

- Elementary processes for handling of revocation lists (☒ 3.3.3),
- Elementary processes for execution of a revocation (☒ 3.3.4)

Interface ApplicationOwner=ProductRetailer and ApplicationOwner=ApplicationRetailer

Specializations of the interface between the generic application owner system and a generic product retailer system or generic application retailer system will be used for the following elementary processes of the product resp. application retailer:

Elementary processes for issuance of applications (☒ 3.1.1)

EP\_revocation\_request\_application (☒3.3.1.1)

EP\_revocation\_order\_application (☒ 3.3.2.1)

EP\_revocation\_application (☒ 3.3.4.1)

EP\_request\_revocation\_cancellation\_Application (☒ 3.3.5.1)

EP\_order\_revocation\_cancellation\_application (☒ 3.3.6.1)

EP\_revocation\_request\_SAM (☒ 3.3.1.3)

EP\_revocation\_request\_organisation (☒ 3.3.1.4)  
EP\_revocation\_request\_key (☒ 3.3.1.5)  
EP\_revocation\_order\_SAM (☒ 3.3.2.2)  
EP\_revocation\_order\_organisation (☒ 3.3.2.3)  
EP\_revocation\_order\_key (☒ 3.3.2.4)  
EP\_revocation\_SAM (☒ 3.3.4.2)  
EP\_revocation\_Organisation (☒ 3.3.4.3)  
EP\_request\_revocation\_cancellation\_SAM (☒ 3.3.5.3)  
EP\_request\_revocation\_cancellation\_organisation (☒ 3.3.5.4)  
EP\_request\_revocation\_cancellation\_key (☒ 3.3.5.5)  
EP\_order\_revocation\_cancellation\_SAM (☒ 3.3.6.2)  
EP\_order\_revocation\_cancellation\_organisation (☒ 3.3.6.3)  
EP\_order\_revocation\_cancellation\_key (☒ 3.3.6.4)  
EP\_Unlock\_Application (☒ 3.3.7.1)  
EP\_Distribution\_SAMs (☒ 3.4.3)  
EP\_Distribution\_Keys (☒ 3.4.2)

## 2.5.8. Interface ApplicationOwner=ProductOwner

Specializations of the interface between the generic application owner system and a generic product owner system will be used for the following elementary processes of the product owner:

- EP\_Distribution\_Keys (☒ 3.4.2)
- EP\_revocation\_request\_organisation (☒ 3.3.1.4)
- EP\_revocation\_order\_organisation (☒ 3.3.2.3)
- EP\_revocation\_request\_key (☒ 3.3.1.5)
- EP\_revocation\_order\_key (☒ 3.3.2.4)
- EP\_request\_revocation\_cancellation\_organisation (☒ 3.3.5.4)
- EP\_order\_revocation\_cancellation\_organisation (☒ 3.3.6.3)
- EP\_request\_revocation\_cancellation\_key (☒ 3.3.5.5)
- EP\_order\_revocation\_cancellation\_key (☒ 3.3.6.4)

## 2.5.9. Interface ApplicationOwner=ServiceOperator

Specializations of the interface between the generic application owner system and a generic service operator system will be used for the following elementary processes of the service operator:

EP\_Distribution\_Keys (☒ 3.4.2)  
EP\_Distribution\_SAMs (☒ 3.4.3)  
EP\_revocation\_request\_SAM (☒ 3.3.1.3)  
EP\_revocation\_order\_SAM (☒ 3.3.2.2)  
EP\_request\_revocation\_cancellation\_SAM (☒ 3.3.5.3)  
EP\_order\_revocation\_cancellation\_SAM (☒ 3.3.6.2)

EP\_revocation\_request\_organisation (3.3.1.4)

EP\_revocation\_order\_organisation (3.3.2.3)

EP\_request\_revocation\_cancellation\_organisation (3.3.5.4)

EP\_order\_revocation\_cancellation\_organisation (3.3.6.3)

EP\_revocation\_request\_key (3.3.1.5)

EP\_revocation\_order\_key (3.3.2.4)

EP\_request\_revocation\_cancellation\_key (3.3.5.5)

EP\_order\_revocation\_cancellation\_key (3.3.6.4)

## 2.5.10. Interface RevocationService=ApplicationOwner

Specializations of the interface between the generic revocation service system and the generic application owner system will be used for the following elementary processes of the application owner:

- EP\_revocation\_order\_organisation (3.3.2.3)
- EP\_order\_revocation\_cancellation\_organisation (3.3.6.3)
- EP\_revocation\_order\_SAM (3.3.2.2)
- EP\_order\_revocation\_cancellation\_SAM (3.3.6.2)
- EP\_revocation\_order\_key (3.3.2.4)
- EP\_order\_revocation\_cancellation\_key (3.3.6.4)
- EP\_revocation\_Organisation (3.3.4.3)
- EP\_revocation\_SAM (3.3.4.2)
- Elementary processes for handling of revocation lists (3.3.3)

## 2.5.11. Interface RevocationService=ProductOwner

Specializations of the interface between the generic revocation service system and the generic product owner system will be used for the following elementary processes of the product owner:

Elementary processes for handling of revocation lists (3.3.3)

Elementary processes for execution of a revocation (3.3.4)

EP\_Configuration\_Revocation\_Service (3.4.1)

## 2.5.12. Interface ProductRetailer=ServiceOperator

Specializations of the interface between a generic product retailer system and a generic service owner system will be used for the following elementary processes:

- Elementary processes for revocation requests (☒ 3.3.1)
- Elementary processes for revocation orders (☒ 3.3.2)
- Elementary processes for execution of a revocation (☒ 3.3.4)
- Elementary processes for requesting cancellation of revocations (☒ 3.3.5)
- Elementary processes for cancellation of revocations (☒ 3.3.6)
- Elementary processes for unblocking (☒ 3.3.7)

## 2.5.13. Interface ProductRetailer= ProductRetailer

Specializations of the interface between one generic product retailer system and another will be used for the following elementary processes:

- Elementary processes for revocation requests (☒ 3.3.1)
- Elementary processes for revocation orders (☒ 3.3.2)
- Elementary processes for execution of a revocation (☒ 3.3.4)
- Elementary processes for requesting cancellation of revocations (☒ 3.3.5)
- Elementary processes for cancellation of revocations (☒ 3.3.6)
- Elementary processes for unblocking (☒ 3.3.7)

### 3. Generic system model: Elementary Processes

The objective of this chapter is to collect and briefly describe the relevant technical “elementary processes” (EP) between the generic components and systems which require their own interfaces.

The EPs defined shall be independent in the sense that none of the EPs described should be able to be constructed from some number of the remaining EPs.

Necessary links to other processes will be described by indicating predecessor and successor processes.

For each EP a brief description, a list of relevant interfaces as well as a list of events triggering the EP and events triggered by the EP will be given.

#### 3.1. *Elementary processes associated with the sale and distribution of applications and entitlements*

This section covers the following elementary processes related to the sale and distribution of applications and entitlements:

- ☒ Elementary processes for issuance of applications
- ☒ Elementary processes for issuance of entitlements
- ☒ Elementary processes for payment
- ☒ Elementary processes for
- ☒ Elementary processes for reimbursement
- ☒ Elementary processes for modification

Here payment is only considered in connection with the transport application and transport service used and, depending on the transport entitlement, may be based on any of a number of means of payment including transport specific stored value or legal tender.

##### 3.1.1. Elementary processes for issuance of applications

This section covers the elementary processes for the issuance of applications to user media which have been appropriately configured.

In case payment is necessary for the performance of any one of the issuance elementary processes an elementary process for payment will follow.

###### 3.1.1.1. EP\_Load\_Application

###### Description:

This elementary process realizes the loading of application code in the form of an executable load file to an appropriately configured user media by the application retailer or application owner. The authenticity and confidentiality of the code must be protected end-to-end between the application retailer resp. application owner and the user media.

It may be required that the User Medium return a receipt to the application retailer as proof of the successful performance of the loading process which is forwarded by the application retailer to the application owner and SE owner.

The executable load file is present and registered in the persistent memory (mutable or immutable) of the SE. The executable load file is initially associated with the SD used to load it. If it is present in immutable persistent memory it is initially associated with the SD of the SE Owner (issuer security domain). See (1).

Loading to immutable persistent memory occurs during the production of the SE. This process is subject to agreement between the SE owner and the application owner and will be considered out of scope for this document.

Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
ApplicationOwner=ApplicationRetailer  
SE Owner=ApplicationRetailer

Triggered by:

Although there is no other elementary process which makes application loading mandatory as successor, there are the following restraints on possible predecessor processes:

- Any of the elementary processes in section can be an immediate predecessor in case the application code is being loaded under the SD of the SE owner (issuer SD).
- EP\_Personalize\_SD: In case the application code is being loaded under an appropriate supplementary SD.
- 

Triggers:

The loading process only makes sense if it will be followed by EP\_Install\_Application possibly with an indefinite delay.

### 3.1.1.2. EP\_Install\_Application

Description:

In this elementary process the linking of executable application code, memory allocation and corresponding registration of the application in the SE is performed. The process is performed within the context of the security policy of the SD associated with the application code from the load process or of another SD equipped with appropriate “authorized management” or “delegated management” privileges. The application becomes accessible to off-card entities authenticated by the associated Security Domain, but is not yet directly selectable from the outside.

This process will only be carried out by the SE if the appropriate authorization of the SE owner is given. See (1).

It may be required that the SE return a receipt to the application retailer as proof of the successful performance of the install process which is forwarded by the application retailer to the application owner and SE owner.

Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
ApplicationOwner=ApplicationRetailer  
SE Owner=ApplicationRetailer

Triggered by:

The issuance of applications only makes sense if the process EP\_Install\_Application is carried out. So while there is no elementary process which makes install application mandatory as successor, the necessary predecessor is EP\_Load\_Application.

Triggers:

The install process only makes sense if it will be followed by EP\_Make Selectable\_Application possibly with an indefinite delay.

### 3.1.1.3. EP\_Make Selectable\_Application

#### Description:

This elementary process makes the installed application able to receive commands from off-card entities. It may be combined with the install process. See (1).

The process is performed within the context of the security policy of the SD associated with the application code from the load process or of another SD equipped with appropriate “authorized management” or “delegated management” privileges.

It may be required that the SE returns a receipt to the application retailer as proof of the successful performance of this process which is forwarded by the application retailer to the application owner and SE owner.

#### Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
ApplicationOwner=ApplicationRetailer  
SE Owner=ApplicationRetailer

#### Triggered by:

The issuance of applications only makes sense if the process EP\_Make Selectable\_Application is carried out. So while there is no elementary process which makes this one mandatory as successor, the necessary predecessor is EP\_Install\_Application.

#### Triggers:

The issuance of applications only makes sense if the generic applications which are issued are actually equipped with specific data for the purpose of ticketing so this elementary process will be followed by EP\_Personalize\_Application possibly with an indefinite delay.

### 3.1.1.4. EP\_Personalize\_Application

#### Description:

This elementary process supplies the generic transport application on the user medium with data specific to the instance of the application including e.g. identifiers, keys and certificates, customer data etc. These data are necessary in order that the application can be used for transport.

In the generic model considered here it will be assumed that entitlements (including where applicable their specific keys) will be loaded to the transport application according to the processes in section 3.1.2 although some or all of these data could alternatively be supplied in the personalization process.

Data elements can be supplied to the application in the personalization process using one or more of the following three general alternatives:

1. Data is delivered in Store Data commands to the SD associated with the application using the security mechanisms of the SD and these commands are forwarded internally to the application;
2. Data is delivered to the application using the command set of the application with secure messaging based on the security mechanisms of the associated SD. The application calls the resources of the SD to process and check the secure messaging.
3. Data is delivered to the application using the command set of the application with

security mechanisms contained in the application itself.

The application retailer can retain a profile of the application for application management purposes which may also need to be forwarded to the application owner.

It may be required that the application return a receipt to the application retailer as proof of the successful performance of this process which is forwarded by the application retailer to the application owner.

Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
ApplicationOwner=ApplicationRetailer

Triggered by:

The issuance of applications only makes sense if the generic applications which are issued are actually equipped with specific data for the purpose of ticketing, i.e. if the process EP\_Personalize\_Application is carried out. So while there is no elementary process which makes this one mandatory as successor, the necessary predecessor for alternative 1 is EP\_Install\_Application and for alternatives 2 and 3 is EP\_Make Selectable\_Application.

Triggers:

If payment is required for the application, then a payment process based on legal tender will be triggered which is considered out of scope for this paper.

### **3.1.2. Elementary processes for issuance of entitlements**

This section covers the elementary processes for the issuance of entitlements to user media which have been appropriately equipped with the corresponding application (see section 3.1.1).

In case payment is necessary for the performance of any one of the elementary issuance processes an elementary process for payment will follow.

#### **3.1.2.1. EP\_Issue\_Entitlement**

Description:

This elementary process realizes the issuance of an entitlement, including if applicable associated keys, to a User Medium by a Product Retailer terminal and the generation of a receipt (issuance transaction) as proof of the successful issuance of the specific entitlement.

The Product Retailer terminal sends the receipt to the Product Retailer system from where it is also forwarded to the relevant Product Owner system.

This process applies equally well to different types of entitlements. In the model presented here the following types are considered:

- Electronic ticket
- Stored Value Entitlement
- Post-Paid Entitlement
- 

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
ProductOwner=ProductRetailer

Triggered by:

Page 21 of 54

None

Triggers:

If payment is required for the entitlement, then a payment process based on legal tender or on an available stored value or post-paid entitlement will be triggered. See Elementary processes for payment (3.1.3).

### 3.1.2.2. EP\_Load\_Stored Value

Description:

This elementary process realizes the loading of value to an existing stored value entitlement on a User Medium by a Product Retailer terminal and the generation of a receipt (load transaction) as proof of the successful loading of the stored value. The Product Retailer terminal sends the receipt to the Product Retailer system from where it is forwarded to the relevant Product Owner system and the system of the Product Retailer who originally issued the stored value entitlement.

The Product Owner is responsible for monitoring all transactions (load or debit) to the stored value entitlement, in particular with regard to authenticity, correctness and completeness.

The issuing Product Retailer is responsible for clearing the stored value account.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
ProductOwner=ProductRetailer  
ProductRetailer=ProductRetailer

Triggered by:

None

Triggers:

If payment is required for the transaction, then a payment process based on legal tender will be triggered.

### 3.1.3. Elementary processes for payment

#### 3.1.3.1. EP\_Debit\_Stored Value

Description:

This elementary process realizes the reduction of the current value in a stored value entitlement on a User Medium by an appropriate amount in exchange for acquired goods and services by a

- Product Retailer terminal or
- Service Operator terminal (in case transport services are acquired on the basis of in-out transactions)

and the generation of a receipt (debit transaction) as proof of the reduction of the stored value. The Product Retailer terminal resp. Service Operator terminal sends the receipt to the Product Retailer system resp. Service Operator system from where it is forwarded to the relevant Product Owner system and to the system of the Product Retailer who originally issued the stored value entitlement.

The Product Owner is responsible for monitoring all transactions (load or debit) to the stored value entitlement, in particular with regard to authenticity, correctness and completeness.

The issuing Product Retailer is responsible for clearing the stored value account.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
resp.  
UserMedium=ServiceOperatorTerminal  
and  
ProductOwner=ProductRetailer  
ProductRetailer=ProductRetailer  
resp.  
ProductOwner=ServiceOperator  
ProductRetailer=ServiceOperator

Triggered by:

EP\_Issue\_Entitlement

EP\_Capture\_Entitlement

Triggers:

None

### 3.1.3.2. EP\_Debit\_Post-Paid

Description:

This elementary process realizes the reduction of the post-paid account associated with the post-paid entitlement on a User Medium by an appropriate amount in exchange for acquired goods and services based on a transaction performed by a

- Product Retailer terminal or
- Service Operator terminal (in case transport services are acquired on the basis of in-out transactions)
- 

and the generation of a receipt (debit transaction) as proof of the reduction of the stored value. The Product Retailer terminal resp. Service Operator terminal sends the receipt (debit transaction) to the Product Retailer system resp. Service Operator system from where it is forwarded to the relevant Product Owner system and to the system of the Product Retailer who originally issued the post-paid entitlement.

The Product Owner is responsible for monitoring all debit transactions to the post-paid entitlement, in particular with regard to authenticity, correctness and completeness.

The issuing Product Retailer is responsible for settling the customers post-paid account.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
resp.  
UserMedium=ServiceOperatorTerminal  
and  
ProductOwner=ProductRetailer  
ProductRetailer=ProductRetailer  
resp.  
ProductOwner=ServiceOperator  
ProductRetailer=ServiceOperator

Triggered by:

EP\_Issue\_Entitlement

EP\_Capture\_Entitlement

Triggers:

None

### **3.1.4. Elementary processes for return**

Here the processes for the return of entitlements and applications before their expiration are described.

If applicable the value of the returned objects will also be refunded, usually to the method of payment that was used to acquire the object in the first place. For the purposes of “reimbursing” the refunded value there are in turn additional elementary processes which will be described below in section 3.1.5.

#### **3.1.4.1. EP\_Return\_Application**

Description:

This elementary process realizes the termination of the application on a User Medium by an Application Retailer terminal upon request of the holder of the User Medium. The process can only be carried out by the Application Retailer who originally issued the specific instance of the application. The User Medium / application must generate a receipt as proof of the successful termination which will be sent to the Application Retailer system and forwarded from there to the Application Owner and the SE Owner.

In case the Application Issuer is also the issuer of the User Medium the termination may be realized by retaining the User Medium and either destroying it or deleting its data.

If the User Medium can not be retained, then after the termination neither the application nor any of the data contained in the application may be accessible on the User Medium. If the application code was stored in mutable persistent memory, then the allocated storage will be freed up.

Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
ApplicationOwner=ApplicationRetailer  
SE Owner=ApplicationRetailer

Triggered by:  
None

**3.1.4.2. If necessary the entitlements contained in the application must first be returned and their value reimbursed before executing the return of the application: EP\_Return\_Entitlement**

**3.1.4.3. EP\_Return\_Entitlement**

Description:

This elementary process realizes the deletion of an entitlement in the application on a User Medium by a Product Retailer terminal upon request of the holder of the User Medium and may only be carried out by the Product Retailer who originally issued the entitlement. The User Medium / application must generate a receipt as proof of the successful deletion which will be sent to the Product Retailer system and forwarded from there to the Product Owner system.

Before an entitlement is deleted any applicable reimbursement of the value of the entitlement to the customer will be carried out. The value of the reimbursement is included in the receipt sent to the Product Owner. In the case of a post-paid entitlement the final billing can only be carried out when all transactions executed before the return of the entitlement have been received by the (issuing) Product Retailer.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
ProductOwner=ProductRetailer

Triggered by:  
None

Triggers:

If applicable, a corresponding elementary process for reimbursement (☒ 3.1.5).

### **3.1.5. Elementary processes for reimbursement**

In connection with the return of an entitlement a reimbursement of the remaining value of the entitlement to the customer may be possible. The reimbursement of the value may be carried out by loading the value to a stored value entitlement, crediting a post paid entitlement or repaying on the basis of legal tender.

**3.1.5.1. EP\_Reimburse\_Post Paid**

Description:

In this elementary process the value of a returned entitlement is credited to a post paid account. The Product Retailer terminal performs a transaction with the User Medium which serves as proof of the authenticity and contains a reference to the post paid entitlement and the amount to be credited. The Product Retailer terminal sends the transaction to the Product Retailer system which forwards it to the Product Owner system. If the Product Retailer

performing the reimbursement is not the same as the Product Retailer who issued the post paid entitlement to be credited, then the Product Owner system forwards the transaction to the system of the issuing Product Retailer.

The issuing Product Retailer is responsible the settlement and clearing.

Relevant interfaces:

UserMedium=ProductRetailerTerminal (executer of reimbursement transaction)  
ProductOwner=ProductRetailer (executer of reimbursement transaction)  
ProductOwner=ProductRetailer (issuer of the credited post paid entitlement)

Triggered by:

EP\_Return\_Entitlement

Triggers:

None

### 3.1.5.2. EP\_Reimburse\_Stored Value

Description:

This elementary process realizes the reimbursement of the value of a returned entitlement to an existing stored value entitlement on a User Medium by a Product Retailer terminal and the generation of a receipt (reimbursement transaction) as proof of the successful reimbursement. The Product Retailer terminal sends the receipt to the Product Retailer system from where it is forwarded to the relevant Product Owner system and the system of the Product Retailer who originally issued the stored value entitlement.

The Product Owner is responsible for monitoring all transactions (in particular reimbursements) to the stored value entitlement, in particular with regard to authenticity, correctness and completeness.

The issuing Product Retailer is responsible for clearing the stored value account.

Relevant interfaces:

UserMedium=ProductRetailerTerminal (executer of reimbursement transaction)  
ProductOwner=ProductRetailer (executer of reimbursement transaction)  
ProductOwner=ProductRetailer (issuer of the credited post paid entitlement)

Triggered by:

EP\_Return\_Entitlement

Triggers:

None

### 3.1.6. Elementary processes for modification

In the following processes for modifying entitlements that have already been issued to User Media will be described.

It may be appropriate for example to modify tariff relevant parameters connected with stored value and post paid entitlements, but without altering other critical data associated with the entitlement e.g. keys, identifiers counters.

Electronic tickets in the usual sense on the other hand are fixed service offers with a predefined price that is payed up front so that it generally does not make sense to modify such entitlements. If necessary, they will instead normally be returned and reissued.

Modifications to other data structures (application data, customer profile data, PIN) may also be appropriate.

#### 3.1.6.1. EP\_Modify\_User Tariff Parameters

##### Description:

This elementary process realizes the modification of personal tariff parameters by the user (e.g. number of passengers travelling, first or second class travel etc.) at a Product Retailer or Service Operator terminal. The modifications are relevant for stored value and post paid entitlements for which the price is calculated on trip or later. The parameters can be modified by the customer at any time, but will only become effective after the next check-out transaction.

A receipt will be generated as proof of the successful modification of the parameters which will be sent by the Product Retailer resp. Service Operator terminal to the Product Retailer resp. Service Operator system from where it is forwarded to the relevant Product Owner system and the system of the Product Retailer who originally issued the entitlement.

##### Relevant interfaces:

UserMedium=ProductRetailerTerminal  
resp.

UserMedium=ServiceOperatorTerminal  
and

ProductOwner=ProductRetailer (executing the modification)  
resp.

ProductOwner=ServiceOperator (executing the modification)  
and

ProductOwner=ProductRetailer (original issuer of the modified entitlement)

##### Triggered by:

None

##### Triggers:

None

#### 3.1.6.2. EP\_Modify\_Tariff Parameter

##### Description:

This elementary process realizes the modification of tariff parameters of an entitlement corresponding to a change in the terms and conditions of the agreement between the

customer and the issuing Product Retailer at a Product Retailer terminal. The modifications are relevant for stored value and post paid entitlements for which the price is calculated on trip or later. The process can only be carried out by the Product Retailer that originally issued the entitlement.

A receipt will be generated as proof of the successful modification of the parameters which will be sent by the Product Retailer terminal to the Product Retailer system from where it is forwarded to the relevant Product Owner system.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
ProductOwner=ProductRetailer

Triggered by:

None

Triggers:

None

### 3.1.6.3. EP\_Modify\_Application

Description:

This elementary process realizes the modification of application data of an application that has already been issued to a User Medium.

The process can only be carried out by the Product Retailer that originally issued the entitlement.

A receipt will be generated as proof of the successful modification of the data on the User Medium which will be sent by the Product Retailer terminal to the Product Retailer system from where it is forwarded to the Application Owner.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
ApplicationOwner=ProductRetailer

Triggered by:

None

Triggers:

None

### 3.1.6.4. EP\_Modify\_Customer Profil

Description:

This elementary process realizes the modification of customer profile data of an application that has already been issued to a User Medium.

The process can only be carried out by the Product Retailer that originally issued the entitlement.

A receipt will be generated as proof of the successful modification of the data on the User Medium which will be sent by the Product Retailer terminal to the Product Retailer system from where it is forwarded to the Application Owner.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
ApplicationOwner=ProductRetailer

Triggered by:

None

Triggers:

None

### 3.1.6.5. EP\_Modify\_PIN

Description:

This elementary process realizes the modification of the PIN for access to certain data in an application by the holder of the User Medium at a Product Retailer or Service Operator terminal. The modification requires knowledge of the currently valid PIN or of a valid unblocking code.

Relevant interfaces:

UserMedium=ProductRetailerTerminal  
UserMedium=ServiceOperatorTerminal

Triggered by:

None

Triggers:

None

## 3.2. *Elementary processes for use of entitlements*

This section describes the processes of inspecting and capturing entitlements that are performed between a Service Operator and an entitlement on the User Medium of a customer when the customer uses the transport service.

Here inspection means that the authenticity of an entitlement and its validity for access to a specific transport service are checked. It may be required to include the generation of corresponding receipts (transactions) as proof of the successful checking of the entitlement and for the Service Operator terminal to generate logs of aborted processes. This is relevant for all types of entitlement.

Capture is specific to stored value or post paid entitlements only. In the case of a stored value entitlement it refers to the process of reducing the stored value by an appropriate amount for a transport service (which in a tap-in tap-out system may involve two stages) and generating an authentic receipt (transaction). In the case of a post paid entitlement appropriate authentic transactions regarding the usage of transport services are generated and used ex post for settlement.

### 3.2.1. **Elementary processes for accessing transport services with an entitlement**

#### 3.2.1.1. EP\_Capture\_Entitlement

Description:

Page 29 of 54

This process realizes the authentic capture of unique identifiers of a stored value or post paid entitlement (including identifiers of the issuing Product Retailer and Product Owner), of the Service Operator as well as the SAM and Terminal used in the process and possible further information relevant for the tariff at the point where a transport service is used by a customer and the generation of a unique receipt (transaction) containing this information, whose authenticity can be checked by the Product Owner and issuing Product Retailer.

In the case of a stored value entitlement the current value (in the User Medium) has to be reduced by the corresponding amount for the service and this amount must also be contained in the receipt. Since the value is administered in the User Medium the terminal must perform an on-trip price calculation. In a tap-in tap-out system this may require increasing the stored value during the capture of a tap-out by an amount less than the reduction performed during the capture of the last tap-in.

The receipt is sent by the Service Operator terminal to the Service Operator system from where it is forwarded to the relevant Product Owner system and from there to the system of the issuing Product Retailer.

Relevant interfaces:

ProductOwner=ServiceOperator  
ProductOwner=ProductRetailer  
UserMedium=ServiceOperatorTerminal

Triggered by:

None

Triggers:

EP\_Debit\_Post-Paid

EP\_Debit\_Stored Value

### **3.2.1.2. EP\_Inspect\_Entitlement**

Description:

This process realizes the verification that appropriate transactions have been performed on a valid stored value or post paid entitlement or that a valid electronic ticket (acquired in advance of the journey) is present for access to the transport service.

It may be required to generate a receipt (transaction) containing information on the result of the inspection, whose authenticity can be checked by the Product Owner and issuing Product Retailer.

If a receipt is generated, it is sent by the Service Operator terminal to the Service Operator system from where it is forwarded to the relevant Product Owner system and from there to the system of the issuing Product Retailer.

Relevant interfaces:

ProductOwner=ServiceOperator  
ProductOwner=ProductRetailer  
UserMedium=ServiceOperatorTerminal

Triggered by:  
None

Triggers:

If a corresponding entry in the appropriate revocation list is found during the inspection process, then the one of the following elementary processes will follow:

### **3.3. Elementary processes for revocation**

Within the framework of a fare management application revocations of the following entities or objects are possible:

- Entitlements,
- User Media Applications,
- SAMs,
- Keys and certificates as well as
- Organisations

and we need elementary processes for

- revocation request,
- revocation order,
- handling of revocation lists,
- execution of a revocation,
- requesting cancellation of revocations,
- unblocking and
- cancellation of revocations

Any authorized entity participating in the fare management system can request the revocation of a certain object or entity within the system. The request is sent to the entity responsible for the issuance and administration of the object (owner).

In the case of revocation of an Entitlements or a User Media Application the request is sent to the issuing Product Retailer resp. Application Retailer. In the case of revocation of a key the request is sent to the owner of the key within the system. In the case of revocation of a SAM or an Organisation the request is sent to the Application Owner.

If the request is approved by the responsible entity a so called request order is sent by the entity to the Revocation Service, which will generate a corresponding entry in the revocation lists it provides to other entities within the system to the extent that the object is relevant for them.

In all cases the entity responsible for the object informs the requesting entity about the approval or rejection of the request.

When an Entitlement or User Media Application is encountered for which an entry in the current revocation list exists, the object must be deleted, locked or otherwise marked as invalid on the media. The successful revocation of an object must be reported so that the corresponding entry in the revocation list can be removed in order to limit the size of the revocation list. It may be necessary to implement permanent as well as reversible revocations. If a key appears in a revocation list, then it is not to be used in any operation with the various objects of the system. The key should be deleted from the components of the system, so that the entry in the revocation list can be eliminated.

No objects that were issued with a revoked SAM after the time of revocation are to be accepted. Revoked SAMs are to be removed from the system (either by physically removing and destroying them or by deleting or otherwise permanently locking the applications with all

their data).

The application owner has special rights with regard to the revocation of User Media Applications. Requests for such revocations by the Application Owner must be implemented as Revocations Orders by the responsible Application Retailer.

The cancellation of a revocation may also be requested. This can be performed by unblocking the object after a revocation on it has been executed. In this case a report of the successful unblocking must be sent.

In case the revocation has not yet been executed on the corresponding object, a cancellation request can be sent to the responsible entity, which it can implement as an order for cancellation of the revocation. After the entry in the corresponding revocation list has been removed a report of the cancellation is returned.

### **3.3.1. Elementary processes for revocation requests**

#### **3.3.1.1. EP\_revocation\_request\_application**

##### Description:

A request for revocation of a User Media Application can be sent by the Application Owner, an authorized Product or Application Retailer or an authorized Service Operator to the Application Retailer who originally issued the application.

Generally the issuing Application Retailer can deny or approve such requests except in the case of requests from the Application Owner, which are to be considered as final.

The issuing Application Retailer informs the requesting entity of the status of the request (denied or approved) in a revocation report.

Upon approving a request the issuing Application Retailer sends a corresponding revocation order to the Revocation Service, which, if successful, leads to the addition of a corresponding entry in the revocation list by the Revocation Service.

##### Relevant interfaces:

ApplicationOwner=ApplicationRetailer (issuer of the application)  
ProductRetailer=ApplicationRetailer (issuer of the application)  
ApplicationRetailer=ApplicationRetailer (issuer of the application)  
ServiceOperator=ApplicationRetailer (issuer of the application)

##### Triggered by:

None

##### Triggers:

EP\_revocation\_order\_application (☒ 3.3.2.1)

#### **3.3.1.2. EP\_revocation\_request\_entitlement**

##### Description:

A request for revocation of an Entitlement can be sent by the respective Product Owner, an authorized Product or Application Retailer or an authorized Service Operator to the Product Retailer who originally issued the Entitlement.

Generally the issuing Product Retailer can deny or approve such requests except in the case of requests from the Product Owner, which are to be considered as final.

The issuing Product Retailer informs the requesting entity of the status of the request (denied or approved) in a revocation report.

Upon approving a request the issuing Product Retailer sends a corresponding revocation order to the Revocation Service, which, if successful, leads to the addition of a corresponding entry in the revocation list by the Revocation Service.

Relevant interfaces:

ProductOwner=ProductRetailer (issuer of the entitlement)  
ProductRetailer=ProductRetailer (issuer of the entitlement)  
ApplicationRetailer=ProductRetailer (issuer of the entitlement)  
ServiceOperator=ProductRetailer (issuer of the entitlement)

Triggered by:

None

Triggers:

EP\_revocation\_order\_entitlement (☒ 3.3.2.5)

### **3.3.1.3. EP\_revocation\_request\_SAM**

Description:

A request for revocation of a SAM can be sent by an authorized Product Owner, Application Retailer, Product Retailer or Service Operator to the Application Owner.

Generally the Application Owner can deny or approve such requests.

The Application Owner informs the requesting entity of the status of the request (denied or approved) in a revocation report.

Upon approving a request the Application Owner sends a corresponding revocation order to the Revocation Service, which, if successful, leads to the addition of a corresponding entry in the revocation list by the Revocation Service.

Relevant interfaces:

ApplicationOwner=ProductOwner  
ApplicationOwner=ApplicationRetailer  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ServiceOperator

Triggered by:

None

Triggers:

EP\_revocation\_order\_SAM (☒ 3.3.2.2)

### **3.3.1.4. EP\_revocation\_request\_organisation**

Description:

A request for revocation of an Organisation can be sent by an authorized Product Owner, Application Retailer, Product Retailer or Service Operator to the Application Owner.

Generally the Application Owner can deny or approve such requests.

The Application Owner informs the requesting entity of the status of the request (denied or

approved) in a revocation report.

Upon approving a request the Application Owner sends a corresponding revocation order to the Revocation Service, which, if successful, leads to the addition of a corresponding entry in the revocation list by the Revocation Service.

Relevant interfaces:

ApplicationOwner=ProductOwner  
ApplicationOwner=ApplicationRetailer  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ServiceOperator

Triggered by:

None

Triggers:

EP\_revocation\_order\_organisation(☒ 3.3.2.3)

### 3.3.1.5. EP\_revocation\_request\_key

Description:

A request for revocation of specific key can be sent by the Application Owner, an authorized Product Owner, Application Retailer, Product Retailer or Service Operator to the owner of the key, which may be an instance of any one of the roles defined in the role model.

Generally the key owner can deny or approve such requests. The Application Owner can make final requests that cannot be rejected.

The key owner informs the requesting entity of the status of the request (denied or approved) in a revocation report.

Upon approving a request the key owner sends a corresponding revocation order to the Revocation Service, which, if successful, leads to the addition of a corresponding entry in the revocation list by the Revocation Service.

Relevant interfaces:

Alle combinations of interfaces between pairs of instances of the roles are possible.

Triggered by:

None

Triggers:

EP\_revocation\_order\_key (☒ 3.3.2.4)

## 3.3.2. Elementary processes for revocation orders

Order by an authorized entity is sent to the Revocation Service, which then which then includes a corresponding entry in the relevant, subsequently distributed revocation lists.

If the order was triggered by a corresponding request of another entity, then the entity authorized to send the order to the Revocation Service also sends a report on the status of the request (denied or approved) to the requesting entity.

### 3.3.2.1. EP\_revocation\_order\_application

Description:

In this elementary process the revocation of a User Medium Application (a specific instance)

is ordered by the issuing Application Retailer. The order is sent by the Application Retailer to the Revocation Service and may indicate one of the following different types of revocations:

- reversible revocation of the application,
- irreversible revocation of the application and
- irreversible revocation of the application with confiscation of the User Medium.

The Application Retailer sends a report to the requesting entity (Application Owner, other Application Retailer, Product Retailer or Service Operator) on the status of the request (denied or approved).

The Revocation Service must provide the revocation entry to all Application and Product Retailers as well as Service Operators within the entire fare management system.

Relevant interfaces:

ApplicationRetailer=RevocationService  
ApplicationRetailer=ApplicationOwner  
ApplicationRetailer= ApplicationRetailer  
ApplicationRetailer=ProductRetailer  
ApplicationRetailer=ServiceOperator

Triggered by:

EP\_revocation\_request\_application (☒ 3.3.1.1)

Triggers:

None

### **3.3.2.2. EP\_revocation\_order\_SAM**

Description:

In this elementary process the revocation of a SAM is ordered by the Application Owner. The order is sent by the Application Owner to the Revocation Service and may indicate one of the following different types of revocations:

- reversible revocation of the application,
- irreversible revocation of the application and
- irreversible revocation of the application with physical removal of the SAM.

The Application Owner sends a report to the requesting entity (Application Retailer, Product Retailer or Service Operator) on the status of the request (denied or approved).

The Revocation Service must provide the revocation entry to all Application Retailers, Product Retailers and Service Operators within the entire fare management system.

Relevante interfaces:

ApplicationOwner=RevocationService  
ApplicationOwner=Application Retailer  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ServiceOperator

Triggered by:

EP\_revocation\_request\_SAM (☒ 3.3.1.3)

Triggers:

None

### **3.3.2.3. EP\_revocation\_order\_organisation**

#### Description:

In this elementary process the revocation of an organization participating in the fare management system is ordered by the Application Owner. The order is sent by the Application Owner to the Revocation Service and may indicate one of the following different types of revocations:

- reversible revocation of the organization or
- irreversible revocation of the organization.

The Application Owner sends a report to the requesting entity (authorized Product Owner, Application Retailer, Product Retailer or Service Operator) on the status of the request (denied or approved).

The Revocation Service must provide the revocation entry to all entities (in all roles) within the entire fare management system.

The revocation of an organization must lead in the fare management system to the nonacceptance / invalidity of all SAMs, Keys, User Media, Entitlements and Transactions of this organization subsequent to the time of revocation.

#### Relevante Interfaces:

ApplicationOwner=RevocationService  
ApplicationOwner=ProductOwner  
ApplicationOwner=ApplicationRetailer  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ServiceOperator

#### Triggered by:

EP\_revocation\_request\_organisation (3.3.1.4)

#### Triggers:

None

### **3.3.2.4. EP\_revocation\_order\_key**

#### Description:

In this elementary process the revocation of a key is ordered by the Key Owner (in any of the roles Application Owner, Product Owner, Application Retailer, Product Retailer or Service Operator). The order is sent by the Key Owner to the Revocation Service and may indicate one of the following different types of revocations:

- reversible revocation of the key or
- irreversible revocation of key.

The Key Owner sends a report to the requesting entity (Application Owner, authorized Product Owner, Application Retailer, Product Retailer or Service Operator) on the status of the request (denied or approved).

The Revocation Service must provide the revocation entry in the revocation lists to all entities (in all roles) employing the key.

The revocation of a key must prevent the use of the key in all parts of the fare management system subsequent to the time of revocation. As final consequence of an irreversible revocation the key should be deleted from or otherwise permanently blocked from usage in all parts of the fare management system.

#### Relevante interfaces:

Page 36 of 54

All interfaces between one of the roles Application Owner, Product Owner, Application Retailer, Product Retailer or Service Operator as Key Owner and the Revocation Service. All interfaces between two of the roles Application Owner, Product Owner, Application Retailer, Product Retailer or Service Operator, one as Key Owner the other as organization requesting the revocation.

Triggered by:

EP\_revocation\_request\_key (☒ 3.3.1.5)

Triggers:

None

### 3.3.2.5. EP\_revocation\_order\_entitlement

Description: In this elementary process the revocation of an Entitlement is ordered by the issuing Product Retailer. The order is sent by the Product Retailer to the Revocation Service and may indicate a reversible or irreversible revocation of the Entitlement.

The Product Retailer sends a report to the requesting entity (respective Product Owner, Product or Application Retailer or Service Operator) on the status of the request (denied or approved).

The Revocation Service must provide the revocation entry to all Product Retailers and Service Operators that accept or otherwise use the Entitlement within the entire fare management system.

Relevante interfaces:

ProductRetailer=RevocationService  
ProductRetailer=ProductOwner  
ProductRetailer=ProductRetailer  
ProductRetailer=ApplicationRetailer  
ProductRetailer=ServiceOperator

Triggered by:

EP\_revocation\_request\_entitlement (☒ 3.3.1.2)

Triggers:

None

### 3.3.3. Elementary processes for handling of revocation lists

The Revocation Service generates revocation lists for the organisations participating in the fare management system. In order to restrict the size of these lists it will generally be necessary to prepare specific lists for the individual organisations containing only entries relevant to each of them. It may also make sense to prepare different types of lists each dedicated to certain kinds of revocations. Some revocations need to be received by all participants in the entire fare management system, e.g. revocations of User Media Applications, SAMs, Organisations and possibly certain Keys.

Instances of the roles Product Owner, Application Retailer, Product Retailer and Service Operator, who will need to check for various types of revocations during operations, must request revocation lists from the Revocation Service in regular, defined intervals. The Application Owner must also have access to the appropriate revocation lists. The relevant elementary processes are

EP\_revocation\_list\_request (☒ 3.3.3.1)

EP\_revocation\_list\_distribution (IFM 3.3.3.2)

### 3.3.3.1. EP\_revocation\_list\_request

Description:

The Revocation Service must provide an interface over which the Application Retailer, Product Retailer, Product Owner, Application Owner and Service Operator Systems can send requests for appropriate, current revocation lists.

Relevant interfaces:

RevocationService=ProductOwner  
RevocationService=ApplicationRetailer  
RevocationService=ProductRetailer  
RevocationService=ServiceOperator  
RevocationService=ApplicationOwner

Triggered by:

None

Triggers:

EP\_revocation\_list\_distribution (IFM 3.3.3.2)

### 3.3.3.2. EP\_revocation\_list\_distribution

Description:

Following a request for revocation lists by an authorized organisation participating in the fare management system, the Revocation Service generates the appropriate list(s) based on the most current information and returns the list(s) to the requesting entity via an interface provided by the Revocation Service for this purpose.

Relevant interfaces:

RevocationService=ProductOwner  
RevocationService=ApplicationRetailer  
RevocationService=ProductRetailer  
RevocationService=ServiceOperator  
RevocationService=ApplicationOwner

Triggered by:

EP\_revocation\_list\_request (IFM 3.3.3.1)

Triggers:

None

## 3.3.4. Elementary processes for execution of a revocation

With regard to revocation there are two categories of elements:

- Individual entitlements or User Medium Applications and
- SAMs, Keys or Organisations which may be associated with the issuance of many entitlements and User Medium Applications.

The execution of a revocation for an entitlement or User Medium Application means that the element is deleted or otherwise marked or put out of service, in which case a proof of a revocation is sent by the system of the revoking instance (Service Operator, Application

Retailer or Product Retailer) to the Revocation Service.

The Revocation Service reduces the revocation list accordingly by not supplying the corresponding entry in future lists.

In the case of the revocation of a SAM, Key or Organisation any related entitlements or User Medium Application must also be revoked (implicitly or explicitly), leading also to the corresponding proof of revocation messages to the Revocation Service.

After a revoked SAM, Key or Organisation has been effectively removed from the fare management system the elementary process for the cancellation of a revocation enables the removal of the corresponding entries in revocation lists.

The proof of revocation messages are forwarded by the Revocation Service to the organizations ordering the revocations. In the case of a User Media Application this is the Application Retailer, in the case of an Entitlement it is the Product Retailer. The organization that ordered the revocation then forwards the proof of revocation to the organization that requested the revocation.

### **3.3.4.1. EP\_revocation\_application**

#### Description:

Upon encountering a User Medium Application, a Service Operator, Product Retailer or Application Retailer Terminal will check whether there is a corresponding entry in the current revocation list. If this is so, the terminal executes the revocation as specified by the revocation list.

The revocation of a User Medium Application may be reversible (locked) or irreversible (deleted) and maybe realized by functionality within the application itself or by an underlying application life cycle management utility. This may additionally enable the reversible (locked) or irreversible (terminated) revocation of the secure element of the User Medium itself. Whether a revocation is to be reversible or irreversible revocation and whether it is to be executed at the level of the User Medium or just the Application must be explicitly included or implicitly clear from the information in the revocation list.

An authentic proof of revocation message is generated with the User Medium by the Terminal and sent via the System of the Service Operator, Product Retailer or Application Retailer to the Revocation Service from where it is forwarded to the Application Retailer who originally ordered the revocation. The Application Retailer in turn forwards the message to the organization originally requesting the revocation (Application Owner, Service Operator, Application Retailer or Product Retailer).

#### Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
UserMedium=ProductRetailerTerminal  
UserMedium=ServiceOperatorTerminal  
ApplicationRetailerTerminal=RevocationService  
ProductRetailerTerminal=RevocationService  
ServiceOperatorTerminal=RevocationService  
RevocationService=ApplicationRetailer  
ApplicationRetailer=ApplicationOwner  
ApplicationRetailer=ProductRetailer  
ApplicationRetailer=Application Retailer  
ApplicationRetailer=Service Operator

#### Triggered by:

Any elementary process in which a User Medium Application interacts with a terminal.

Triggers:

None

### **3.3.4.2. EP\_revocation\_SAM**

Description:

During operation a Service Operator, Product Retailer or Application Retailer Terminal must check whether there is an entry for its SAM in the current revocation list. If this is so, the terminal is not allowed to execute any operations with the SAM.

Furthermore, upon encountering a User Medium Application or Entitlement, a Service Operator, Product Retailer or Application Retailer Terminal must check whether the object was issued with a SAM appearing in the revocation list. If this is the case, then the revocation of the User Medium Application resp. Entitlement must be executed according to EP\_revocation\_application (☒ 3.3.4.1) resp. EP\_revocation\_Entitlement (☒ 3.3.4.5).

As long as objects issued by a specific revoked SAM are still valid and in circulation the corresponding entry must remain in the list. The execution of the elementary process EP\_order\_revocation\_cancellation\_SAM (☒ 3.3.6.2) leads to the removal of the entry.

Relevante interfaces:

UserMedium=ApplicationRetailerTerminal

UserMedium=ProductRetailerTerminal

UserMedium=ServiceOperatorTerminal

Triggered by:

All processes that require a Terminal to use a SAM or to communicate with an object issued by a SAM.

Triggers:

EP\_revocation\_application (☒ 3.3.4.1)

EP\_revocation\_Entitlement (☒ 3.3.4.5)

### **3.3.4.3. EP\_revocation\_Organisation**

Description:

During operation a Service Operator, Product Retailer or Application Retailer Terminal must check whether there is an entry for its own Organisation in the current revocation list. If this is so, the terminal is not allowed to execute any operations with User Media or its SAM. Before using a Key the Terminal must check whether it belongs to an Organisation appearing in the revocation list.

Furthermore, upon encountering a User Medium Application or Entitlement, a Service Operator, Product Retailer or Application Retailer Terminal must check whether the object was issued by or requires the use of a key belonging to an Organisation appearing in the revocation list. If this is the case, then the revocation of the User Medium Application resp. Entitlement must be executed according to EP\_revocation\_application (☒ 3.3.4.1) resp. EP\_revocation\_Entitlement (☒ 3.3.4.5).

As long as objects issued by a specific revoked Organisation are still valid and in circulation the corresponding entry must remain in the list. The execution of the elementary process EP\_order\_revocation\_cancellation\_organisation (☒ 3.3.6.3) leads to the removal of the entry.

Relevante interfaces:

UserMedium=ApplicationRetailerTerminal  
UserMedium=ProductRetailerTerminal  
UserMedium=ServiceOperatorTerminal

Triggered by:

Any process that requires a Terminal to use a SAM or to communicate with an object issued by a SAM.

Triggers:

EP\_revocation\_application (☒ 3.3.4.1)  
EP\_revocation\_Entitlement (☒ 3.3.4.5)

### 3.3.4.4. EP\_revocation\_Key

Description:

Before using a key a Service Operator, Product Retailer or Application Retailer Terminal must check whether there is an entry for it in the current revocation list. If this is so, the terminal is not allowed to execute any operations with this key.

Furthermore, upon encountering a User Medium Application or Entitlement, a Service Operator, Product Retailer or Application Retailer Terminal must check whether the object was issued by or requires the use of a key appearing in the revocation list. If this is the case, then the revocation of the User Medium Application resp. Entitlement must be executed according to EP\_revocation\_application (☒ 3.3.4.1) resp. EP\_revocation\_Entitlement (☒ 3.3.4.5).

As long as objects associated with a specific revoked Key are still valid and in circulation the corresponding entry must remain in the list. The execution of the elementary process EP\_order\_revocation\_cancellation\_key (☒ 3.3.6.4) leads to the removal of the entry.

Relevante interfaces:

UserMedium=ApplicationRetailerTerminal  
UserMedium=ProductRetailerTerminal  
UserMedium=ServiceOperatorTerminal

Triggered by:

Any process that requires a Terminal to use a SAM or to communicate with an object issued by a SAM.

Triggers:

EP\_revocation\_application (☒ 3.3.4.1)  
EP\_revocation\_Entitlement (☒ 3.3.4.5)

### 3.3.4.5. EP\_revocation\_Entitlement

Description:

Upon encountering an Entitlement, a Service Operator, Product Retailer or Application Retailer Terminal will check whether there is a corresponding entry in the current revocation list. If this is so, the terminal executes the revocation as specified by the revocation list.

The revocation of an Entitlement may be reversible or irreversible. Whether a revocation is to be reversible or irreversible revocation must be explicitly included or implicitly clear from the information in the revocation list.

An authentic proof of revocation message is generated with the User Medium by the Terminal and sent via the System of the Service Operator, Product Retailer or Application Retailer to the Revocation Service from where it is forwarded to the Product Retailer who originally ordered the revocation and Product Owner. The Product Retailer in turn forwards the message to the organization originally requesting the revocation (Service Operator, Application Retailer or Product Retailer).

#### Relevant interfaces:

UserMedium=ApplicationRetailerTerminal  
UserMedium=ProductRetailerTerminal  
UserMedium=ServiceOperatorTerminal  
ApplicationRetailerTerminal=RevocationService  
ProductRetailerTerminal=RevocationService  
ServiceOperatorTerminal=RevocationService  
RevocationService=ProductRetailer  
RevocationService=ProductOwner  
ProductRetailer=ApplicationOwner  
ProductRetailer=ProductRetailer  
ProductRetailer=Application Retailer  
ProductRetailer=Service Operator

#### Triggered by:

Any elementary process in which a Terminal interacts with an Entitlement.

#### Triggers:

None

### **3.3.5. Elementary processes for requesting cancellation of revocations**

Any instance that has sent a revocation request is responsible to decide if the reason for the revocation still exists. If no proof of revocation message has been received and the reason no longer exists, then the requesting instance must send a cancellation request to the

- issuing Application Retailer in case of the revocation of a User Medium Application,
- issuing Product Retailer in case of the revocation of an Entitlement or
- Application Owner in case of the revocation of a SAM, Key or Organisation.

If all existing requests for revocation of the object are cancelled, the instance responsible for the object, i.e. for the revocation order, can send a cancellation order to the Revocation Service.

In case a reversible revocation of a User Medium Application or Entitlement has already been executed, a cancellation request leads to an unlock process.

The instance responsible for the object, i.e. for the revocation order, informs the entities requesting the revocation and its cancellation of the status of the cancellation request (denied or approved) in a notification of cancellation of a revocation.

### 3.3.5.1. EP\_request\_revocation\_cancellation\_Application

Description:

In this elementary process the Application Owner, a Service Operator, Product Retailer or Application Retailer sends a request for the cancellation of the revocation of a User Medium Application to the issuing Application Retailer. The issuing Application Retailer can decide on his own to cancel the revocation of a User Medium Application.

The issuing Application Retailer decides whether a cancellation order for or an unblocking of the User Medium Application is to be executed.

Relevant interfaces:

ApplicationOwner=ApplicationRetailer (issuer of the User Medium Application)  
ApplicationRetailer=ApplicationRetailer (issuer of the User Medium Application)  
ProductRetailer=ApplicationRetailer (issuer of the User Medium Application)  
ServiceOperator=ApplicationRetailer (issuer of the User Medium Application)

Triggered by:

None

Triggers:

EP\_order\_revocation\_cancellation\_application (☒ 3.3.6.1)

or

EP\_Unlock\_Application (☒ 3.3.7.1)

### 3.3.5.2. EP\_request\_revocation\_cancellation\_entitlement

Description:

In this elementary process the Product Owner, a Service Operator, Product Retailer or Application Retailer sends a request for the cancellation of the revocation of an Entitlement to the issuing Product Retailer. The issuing Product Retailer can decide on his own to cancel the revocation of an Entitlement.

The issuing Product Retailer decides whether a cancellation order for or an unblocking of the Entitlement is to be executed.

Relevant interfaces:

ProductOwner=ProductRetailer (issuer of the Entitlement)  
ApplicationRetailer=ProductRetailer (issuer of the Entitlement)  
ProductRetailer=ProductRetailer (issuer of the Entitlement)  
ServiceOperator=ProductRetailer (issuer of the Entitlement)

Triggered by:

None

Triggers:

EP\_order\_revocation\_cancellation(☒ 3.3.6.5)

or

EP\_Unblock\_Entitlement (☒ 3.3.7.2)

### 3.3.5.3. EP\_request\_revocation\_cancellation\_SAM

#### Description:

In this elementary process a Service Operator, Product Retailer or Application Retailer sends a request for the cancellation of the revocation of SAM to the Application Owner. The Application Owner can decide on his own to cancel the revocation of a SAM.

The Application Owner decides whether a cancellation order for or an unblocking of the SAM is to be executed.

#### Relevant interfaces:

ApplicationOwner=ApplicatioRetailer  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ServiceOperator

#### Triggered by:

None

#### Triggers:

EP\_order\_revocation\_cancellation\_SAM (☒ 3.3.6.2)  
or  
EP\_Unlock\_Application (☒ 3.3.7.1)

### 3.3.5.4. EP\_request\_revocation\_cancellation\_organisation

#### Description:

In this elementary process a Service Operator, Product Retailer, Application Retailer or Product Owner sends a request for the cancellation of the revocation of an Organisation to the Application Owner. The Application Owner can decide on his own to cancel the revocation of an Organisation.

#### Relevant interfaces:

ApplicationOwner=ProductOwner  
ApplicationOwner=ApplicationRetailer  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ServiceOperator

#### Triggered by:

None

#### Triggers:

EP\_order\_revocation\_cancellation\_organisation (☒ 3.3.6.3)

### 3.3.5.5. EP\_request\_revocation\_cancellation\_key

#### Description:

In this elementary process the Application Owner, a Product Owner, Service Operator, Product Retailer or Application Retailer sends a request for the cancellation of the revocation of a Key to the Key Owner (which may take on any of the roles). A Key Owner can decide on his own to cancel the revocation of one of his own keys.

The Key Owner decides whether a cancellation order for or an unblocking of the key is to be executed.

Relevant interfaces:

Alle combinations of interfaces between pairs of instances of the roles are possible.

Triggered by:

None

Triggers:

EP\_order\_revocation\_cancellation\_key (☒ 3.3.6.4)

### **3.3.6. Elementary processes for cancellation of revocations**

If the reason for a revocation of a User Medium Application, Entitlement, SAM, Organisation or Key no longer exists the revocation must be cancelled and the corresponding entry should no longer appear in any new revocation lists.

The instance requesting the cancellation will be informed about the status of the request in a notification of revocation cancellation by the organization responsible for the decision with regard to revocation.

In case revocation requests were received from more than one organization, then the order for cancellation of the revocation may only be given when each of these organizations has also sent a corresponding request for cancellation of the revocation.

#### **3.3.6.1. EP\_order\_revocation\_cancellation\_application**

Description:

In this elementary process the issuing Application Retailer sends an order to cancel the revocation of a User Medium Application to the Revocation Service and a corresponding notification of the cancellation to the instance requesting the cancellation of the revocation (as well as the original revocation). The Revocation Service updates the revocations lists.

Relevant interfaces:

RevocationService=ApplicationRetailer (original issuer of the User Medium Application)

ApplicationOwner=ApplicationRetailer (original issuer of the User Medium Application)

ApplicationRetailer=ApplicationRetailer (original issuer of the User Medium Application)

ProductRetailer=ApplicationRetailer (original issuer of the User Medium Application)

ServiceOperator=ApplicationRetailer (original issuer of the User Medium Application)

Triggered by:

EP\_revocation\_request\_application (☒ 3.3.1.1)

Triggers:

None

#### **3.3.6.2. EP\_order\_revocation\_cancellation\_SAM**

#### Description:

In this elementary process the Application Owner sends an order to cancel the revocation of a SAM to the Revocation Service and a corresponding notification of the cancellation to the instance requesting the cancellation of the revocation as well as to the instance requesting the original revocation. The Revocation Service updates the revocation lists.

Instead of separately revoking objects issued by a revoked SAM, e.g. applications and entitlements, these may be implicitly recognized as invalid by comparing information about the issuing SAM contained in the application and entitlement data with the revocation list for SAMs. With the removal of a SAM from the revocation list, any of these objects associated with it will no longer be considered invalid. For this reason it may be necessary to leave a SAM in the revocation list even after it has been found and physically removed from the system.

#### Relevant interfaces:

RevocationService=ApplicationOwner  
ApplicationOwner=ServiceOperator  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ApplicationRetailer

#### Triggered by:

EP\_request\_revocation\_cancellation\_SAM (📄 3.3.5.3)

#### Triggers:

None

### **3.3.6.3. EP\_order\_revocation\_cancellation\_organisation**

#### Description:

In this elementary process the Application Owner sends an order to cancel the revocation of an Organisation to the Revocation Service and a corresponding notification of the cancellation to the instance requesting the cancellation of the revocation as well as to the instance requesting the original revocation. The Revocation Service updates the revocation lists.

Instead of separately revoking objects and components issued and operated by a revoked Organisation, e.g. Terminals, Keys, SAMs, Applications and Entitlements, these may be implicitly recognized as invalid by comparing the ID of the issuing Organisation associated with the object or component with the revocation list for Organisations. With the removal of an Organisation from the revocation list, any of these objects and components associated with it will no longer be considered invalid. For this reason it may be necessary to leave an Organisation in the revocation list even after certain components belonging to the Organisation (e.g. Terminals, Keys, SAMs) have been removed from the system.

#### Relevant interfaces:

RevocationService=ApplicationOwner  
ApplicationOwner=ProductOwner  
ApplicationOwner=ServiceOperator  
ApplicationOwner=ProductRetailer  
ApplicationOwner=ApplicationRetailer

#### Triggered by:

EP\_request\_revocation\_cancellation\_organisation (📄 3.3.5.4)

#### Triggers:

None

#### 3.3.6.4. EP\_order\_revocation\_cancellation\_key

##### Description:

In this elementary process a KeyOwner (in any one of the roles) sends an order to cancel the revocation of one of his Keys to the Revocation Service and a corresponding notification of the cancellation to the instance requesting the cancellation as well as to the instance requesting the original revocation. The Revocation Service updates the revocation lists. Instead of separately revoking objects associated with a revoked Key, e.g. Applications and Entitlements, these may be implicitly recognized as invalid by comparing the Keys associated with such objects with those on the revocation list. With the removal of a key reference from the revocation list, any objects associated with it will no longer be considered invalid. For this reason it may be necessary to leave a key reference in the revocation list even after the Key has been removed or blocked in system components (until the associated objects have either all expired or been blocked).

##### Relevant interfaces:

Between RevocationService and each of the roles as KeyOwner  
Between the KeyOwner and each of the roles as sender of the cancellation request  
Between the KeyOwner and each of the roles as sender of the revocation request

##### Triggered by:

EP\_revocation\_request\_key (☒ 3.3.1.5)

##### Triggers:

None

#### 3.3.6.5. EP\_order\_revocation\_cancellation\_entitlement

##### Description:

In this elementary process a Product Retailer responsible for the issuance of an Entitlement sends an order to cancel the revocation to the Revocation Service and a corresponding notification of the cancellation to the instance requesting the cancellation as well as to the instance requesting the original revocation. The Revocation Service updates the revocation lists.

The Entitlement should remain in the revocation list until it is either blocked or no longer valid.

##### Relevant interfaces:

RevocationService=ProductRetailer  
ProductRetailer=ProductOwner  
ProductRetailer=ServiceOperator  
ProductRetailer= ProductRetailer

##### Triggered by:

EP\_request\_revocation\_cancellation\_entitlement (☒ 3.3.5.2)

##### Triggers:

None

### 3.3.7. Elementary processes for unblocking

The execution of a revocation on a User Medium Application or Entitlement means that it is blocked for further use resp. will be treated as invalid by Terminals. The Application or Entitlement can be unblocked by the Application Retailer resp. the Product Retailer if the reason for the revocation no longer applies. The instances requesting the revocation resp. the cancellation of the revocation, if applicable, are notified of the unblocking.

If the blocking occurred as a result of the revocation of a SAM, Key or Organisation, it will in general be necessary to issue a new User Medium Application resp. Entitlement.

Although an Application or Entitlement has been unblocked, it may happen that some Terminals are still working with a Revocation List containing the original, but no longer applicable entry. It should be avoided that the Application resp. Entitlement be consequently blocked again on the basis of such an entry. To achieve this, the Application resp. Entitlement must contain e.g. a counter, whose original value is included in the revocation list entry and whose stored value is incremented during the blocking and unblocking operations.

### **3.3.7.1. EP\_Unlock\_Application**

#### Description:

In this elementary process the Application Retailer unblocks the User Medium Application. For this purpose the Application Retailer Terminal must provide a corresponding interface to the User Medium. If not already carried out, the Application Retailer must send an order to cancel the original revocation to the Revocation Service.

The Application Retailer System sends notifications of the unblocking to the Application Owner System, the System of the Service Operator resp. Product Retailer or Application Retailer originally requesting the revocation and if applicable to the System of the instance requesting the cancellation of the revocation.

#### Relevant interfaces:

ApplicationRetailerTerminal=UserMedium  
ApplicationRetailer=ApplicationOwner  
ApplicationRetailer=ServiceOperator  
ApplicationRetailer=ProductRetailer  
ApplicationRetailer=ApplicationRetailer

#### Triggered by:

None

#### Triggers:

EP\_order\_revocation\_cancellation\_application (■ 3.3.6.1)

### **3.3.7.2. EP\_Unblock\_Entitlement**

#### Description:

In this elementary process the Product Retailer unblocks an Entitlement in a User Medium Application. For this purpose the Product Retailer Terminal must provide a corresponding interface to the User Medium. If not already carried out, the Product Retailer must send an order to cancel the original revocation to the Revocation Service.

The Product Retailer System sends notifications of the unblocking to the Product Owner System, the System of the Service Operator resp. Product Retailer or Application Retailer originally requesting the revocation and if applicable to the System of the instance requesting the cancellation of the revocation.

Relevant interfaces:

ProductRetailerTerminal=UserMedium  
ProductRetailer=ProductOwner  
ProductRetailer=ServiceOperator  
ProductRetailer=ProductRetailer  
ProductRetailer=ApplicationRetailer

Triggered by:

None

Triggers:

EP\_order\_revocation\_cancellation\_entitlement (3.3.6.5)

### **3.4. Elementary processes for configuration**

#### **3.4.1. EP\_Configuration\_Revocation\_Service**

Description:

The Revocation Service sends revocation lists tailored to the needs of individual organizations or terminals. To this end the Revocation Service requires information about which products are issued or accepted by the organizations or terminals. In this elementary process a Product Owner informs the Revocation Service which organizations or possibly also which terminals are allowed to issue resp. accept its product(s).

Relevante interfaces:

RevocationService=ProductOwner

Triggered by:

None

Triggers:

None

#### **3.4.2. EP\_Distribution\_Keys**

This elementary process is concerned with the secure distribution of keys needed for the loading and administration of applications (User Medium and SAM) as well as keys needed in SAMs within the transport application itself.

In particular keys belonging to the Application Owner for the loading and initialization of User Medium or SAM Applications to trusted hardware must be initially distributed to the hardware in a secure way.

Further keys specific to the transport application and belonging to instances in the various roles are distributed in the next step to the SAMs.

#### **3.4.3. EP\_Distribution\_SAMs**

Description:

In this elementary process SAMs are distributed to Application and Product Retailers as well as Service Operators for use in their Terminals. This may be achieved either by opening a secure channel to the Application Owner to load the SAM Application directly to a trusted hardware component in the terminal or by sending the SAM in secure hardware form to authorized organizations responsible for terminals.

Relevant interfaces:

ApplicationOwner=ApplicationRetailer resp. ApplicationRetailerTerminal (trusted hardware)

ApplicationOwner=ProductRetailer resp. ProductRetailerTerminal (trusted hardware)  
ApplicationOwner=ServiceOperator resp. ServiceOperatorTerminal (trusted hardware)

Triggered by:

None

Triggers:

None

### **3.4.4. EP\_Distribution\_Product\_Module**

Description:

In this elementary process a Product Owner authorizes Product Retailers to sell and Service Operators to accept a specific product and supplies them with product module for their terminals which contains the tariff parameters, rules and algorithms for calculating the price and for automatic validation of the product as well as a template describing the structure and necessary parameters of the product on the User Media Application.

Relevant interfaces:

ProductOwner=ProductRetailer

ProductOwner=ServiceOperator

Triggered by:

None

Triggers:

None

### **3.4.5. EP\_Deactivation\_Product\_Module**

Description:

In this elementary process a Product Owner informs authorized Product Retailers and Service Operators that a specific product module is no longer valid and specifies a point in time after which the module may not be used for issuing entitlements and another later point in time after which it may not be used for validation of entitlements.

Relevant interfaces:

ProductOwner=ProductRetailer

ProductOwner=ServiceOperator

Triggered by:

None

Triggers:

None

## 4. Interactive Processes

In the IFM Roadmap the following series of scenarios of increasing interoperability between national systems has been identified:

Step 0: Status Quo, separate IFMs

Step 1: Multi-application

1a) common multi-application processes

1b) common European portal

Step 2: Common EU Application

2a) European Application Owner

2b) Common Statutes accepted

2c) Common EU User Medium Application established and used for local products

2d) Some products agreed and accepted as common in EU Application

2e) EU networks using EU IFM Application and EU products directly

The next step is to consider at what levels two or more such systems will have to interact with each other at each of these steps in the roadmap. The approach to this problem here is to identify for the generic system model and for each of these steps those generic elementary processes which will have to operate (at least partially) in the wider context of more than one system resp. for which a common approach will be necessary. This should include considerations not only of technical interfaces and data, but also logistical, contractual and commercial aspects.

For each of the scenarios the following table identifies the elementary processes (marked with an X) where interaction in this sense appears necessary. These items will then need to be examined further to identify the need for standardization. This question is the subject of the next chapter.

Elementary Process	Scenarios						
	1a	1b	2a	2b	2c	2d	2e
EP_Load_Application	X						
EP_Install_Application	X						
EP_Make_Selectable_Application	X						
EP_Personalize_Application	X						
EP_Issue_Entitlement			X				
EP_Load_Stored Value						X	
EP_Debit_Stored Value						X	
EP_Debit_Post-Paid						X	
EP_Return_Application		X					
EP_Return_Entitlement		X					
EP_Reimburse_Post Paid						X	
EP_Reimburse_Stored Value						X	
EP_Modify_User Tariff Parameters			X				
EP_Modify_Tariff Parameter			X				
EP_Modify_Application		X					
EP_Modify_Customer Profil			X				
EP_Modify_PIN			X				
EP_Capture_Entitlement			X				
EP_Inspect_Entitlement			X				
EP_revocation_request_application			X				
EP_revocation_request_entitlement					X		
EP_revocation_request_SAM						X	

Elementary Process	Scenarios						
	1a	1b	2a	2b	2c	2d	2e
EP_revocation_request_organisation							
EP_revocation_request_key							
EP_revocation_order_application							
EP_revocation_order_SAM							
EP_revocation_order_organisation							
EP_revocation_order_key							
EP_revocation_order_entitlement							
EP_revocation_list_request							
EP_revocation_list_distribution							
EP_revocation_application							
EP_revocation_SAM							
EP_revocation_Organisation							
EP_revocation_Key							
EP_revocation_Entitlement							
EP_request_revocation_cancellation_Application							
EP_request_revocation_cancellation_entitlement							
EP_request_revocation_cancellation_SAM							
EP_request_revocation_cancellation_organisation							
EP_request_revocation_cancellation_key							
EP_order_revocation_cancellation_application							
EP_order_revocation_cancellation_SAM							
EP_order_revocation_cancellation_organisation							
EP_order_revocation_cancellation_key							
EP_order_revocation_cancellation_entitlement							
EP_Unlock_Application							
EP_Unblock_Entitlement							
EP_Configuration_Revocation_Service							
EP_Distribution_Keys							
EP_Distribution_SAMs							
EP_Distribution_Product_Module							
EP_Deactivation_Product_Module							

## 5. Requirements for assuring Interoperability

In the next steps we should examine the identified “interactive” processes and find out which specific aspects should be “standardized” in order to assure interoperability. It should be considered for which aspects common ground already exists, what alternatives there are for a common standard and which alternative is to be recommended.

A detailed treatment at the level of the individual processes is beyond the scope of this paper. However, we can give a general indication of the most important aspects in each of the scenarios of the road map.

For the examination we can categorize standardization into the following aspects:

- data structures and communication,
- security measures,
- assignment of residual risk,
- cooperation agreements (rules),
- commercial agreements and
- service level agreements.

In scenario 1a the essential problem is finding a common framework for the cross loading of User Medium Applications, i.e. the loading of the application of one EFM on to the User Media of another EFM which of course must be done in a secure manner.

This requires a common technical basis in the User Media for application loading, the organization of key management, in particular, distribution procedures appropriate to the individual EFMs so they can manage their applications as well as cooperation and commercial agreements with regard to the mutual usage of User Media platforms, e.g. with regard to the cost of storage on the platform or graphical layouts on smart cards or shared advertisement.

It has been proposed in WP3 to use the Global Platform standard as the technical basis for the application download. While implementing the first scenario the details concerning the application of this standard still need to be worked out and the questions concerning key management and agreements have to be addressed.

In scenario 1b a common EU portal will be added which will serve as a “one-stop” portal for customers who require applications from more than one of the cooperating IFM systems. The portal needs to reroute the requests to the various EFM systems appropriately. Consequently the portal is an additional coordinating system outside the EFM systems which however does not imply a fundamental structural change or opening for the EFM systems. Following the procedure for loading the various applications of scenario 1a Since the after the loading of the appropriate applications the further steps take place essentially within the separate EFM systems, the common portal acting as a proxy connecting to interfaces to the various EFM systems that are already available

Scenarios 2a) and 2b) prepare the ground with regard to common rules for establishment of a central organisation that will play the role of Application Owner for a central EU Application. The daily operational processes of national EFMs will however for the most part not be influenced.

In scenario 2c there will be migrations in the individual EFM systems to support the use of the common EU Application for their local products. This does not include a further opening of the systems to more interaction, but can be considered as preparing the way for this as

the next step.

In scenarios 2d and 2e full opening of the national EFM systems to issue and use common products based on the common EU Application. In fact all of the contractual relations and technical processes found between organisations within the national EFMs can now take place between entites previously operating in separate national EFMs.

As these last scenarios have a high level of complexity it appears necessary to approach them with general models applicable to the various national EFMs, e.g. the elementary processes as above, and to work out in detail common procedures within the framework of a later implementation project dedicated to these stages.