Prepared by	Document id	Version	Document date
WM-data	RKF-0022	1.0	31 May 2001

RKF Travel Card
Implementation Specification Type 1

CONTENTS

1 INTRODUCTION 4
1.1 Scope 4
1.2 Reader’s Guide 4
2 OVERVIEW OF THE RKF TRAVEL CARD 5
2.1 Travel Card Contents 5
2.2 Travel Card Structure 6
2.3 Travel Card Layers 7
3 CARD ISSUER LAYER 8
3.1 CMI: Card Manufacturer Information 8
4 TRAVEL CARD SUPPORT LAYER 11
4.1 TCCI: Travel Card Card Information 11
4.2 TCDI: Travel Card Directory 13
4.3 TCAS: Travel Card Applications Status 15
5 TRAVEL CARD APPLICATIONS LAYER 20
5.1 TCPU: Travel Card Purse 21
5.2 TCEL: Travel Card Event Log 23
5.3 TCTI: Travel Card Ticket 27
5.4 TCCO: Travel Card Contract 32
5.5 TCST-ci/co: Travel Card Special Ticket (ci/co) 38
5.6 TCCP: Travel Card Customer Profile 42
5.7 TCDB: Travel Card Discount Basis 43
5.8 TCRE: Travel Card Reservation 46
6 SECURITY 47
6.1 Smart card security 47
6.2 Information Security Requirements 48
7 IMPLEMENTATION RULES 50
7.1 Transaction Concept 50
7.2 Types of Application Objects 52
7.3 General Rules for Travel Card Data 57
7.4 Rules for Coding Data Elements 60
7.5 Interpretation of the Directory 61
7.6 Ticket/Log Area 62
7.7 Calculation of Checksum and MAC Values 65
7.8 Travel Card Consistency 68
7.9 Special Ticket (TCST) Concept 69
7.10 Management of Currency Unit and Money Values 70
7.11 Management of Places 71

8 CONSTANTS 72
8.1 Identifier Values 72
8.2 AID Values 73
8.3 PIX Values 74
8.4 Event Codes 74

1 INTRODUCTION

1.1 Scope

This document in collaboration with [RKF-0023] specifies the implementation of the RKF travel card on a type 1 card, i.e. a contactless memory card in accordance with [ISO 14443], type A of size 1 kB.

This implementation meets the requirements specified in [RKF-0020] with the exception that some information of the card issuer layer is included in the travel card support layer instead as described in section 3.1.2.

The technical details of the type 1 card are specified in [RKF-0021].

If two or more PTA's base their travel card systems upon this specification and at the same time wishes to obtain interoperability between them, an interoperability agreement should be made in accordance with the description in [RKF-0024].

1.2 Reader's Guide

Chapter 2 is a short overview of the RKF travel card type 1 implementation.

Chapters 3 to 5 describe the implementation of the 3 layers of the RKF travel card. These chapters describe the applications, for example purse, tickets and contracts, in detail.

Chapter 6 describes how the security specifications are implemented.

Chapter 7 describes a set of rules for handling the travel card. It is recommended to read this chapter (especially sections 7.1 and 7.2) before reading chapters 3 to 5.

Finally, chapter 8 lists relevant constants.

The sections named '... Requirements' of chapters 3, 4 and 5 specify in detail which requirements of [RKF-0020] are met, and which are not.

A knowledge of the contents of the requirement specification [RKF-0020] is required.

It is recommended, that this document is read "in parallel" with the implementation specification details in [RKF-0023] and the implementation guide in [RKF-0024].

In this document, names of data elements and data element groups are written in italics using the Arial font, e.g.: *TCPU: Purse*.

2 OVERVIEW OF THE RKF TRAVEL CARD

2.1 Travel Card Contents

Contents of card issuer layer:

System object	Description
CMI: Card Manufacturer Information	Fixed card serial number and other manufacturer information

Contents of the travel card support layer:

System object	Description
TCCI: Card Information	General information to the travel card, e.g. a travel card version number
TCDI: Directory	Directory describing the positioning of system objects and application objects on the card
TCAS: Applications Status	A system object that manages to ensure transactions to be completed indivisibly

Contents of travel card applications layer:

Application object	Description
TCPU: Purse	Electronic purse (optional)
TCEL: Event Log	The event log has a limited number of log records, holding information on recent transactions (optional)
TCTI: Ticket	Ticket application, that can be composed from a range of small data element groups by the PTA (optional)
TCCO: Travel Card Contract	Contract application, that can be composed from a range of small data element groups by the PTA (optional)
TCST-ci/co: Special Ticket (ci/co)	Special ticket primarily for check-in/check-out validation (optional)
TCCP: Customer Profile	Description of customer that supplies default information etc. to validation transactions (optional)
TCDB: Discount Basis	Registration of acquired level of discount for travels (optional)
TCRE: Reservation	RFU

2.2 Travel Card Structure

The type 1 travel card is structured in 16 sectors numbered 0..15.

Each sector contains 4 blocks numbered 0..3.

Each block contains 16 bytes of 8 bits.

Block 3 of each sector is a sector trailer containing data to control access to blocks 0, 1, and 2 of the sector. Consequently only 3 of 4 blocks of a sector can be used for by the PTAs to store useful data.

Block y of sector x is often denoted:

Sx:By

The following illustration outlines the structure of the type 1 travel card (the sector trailers in block 3 are shaded grey).

	B0	B1	B2	B3
S0	S0:B0	S0:B1	S0:B2	S0:B3
S1	S1:B0	S1:B1	S1:B2	S1:B3
S2	S2:B0	S2:B1	S2:B2	S2:B3
S3	S3:B0	S3:B1	S3:B2	S3:B3
...	...	...	...	...
S14	S14:B0	S14:B1	S14:B2	S14:B3
S15	S15:B0	S15:B1	S15:B2	S15:B3

The size of one block is:

16 B = 128 b

The total size of a sector is:

4 x 16 B = 64 B = 512 b

The size of data available to PTAs (excluding sector trailer) of one sector is:

3 x 16 B = 48 B = 384 b

The total size of an entire card is:

16 x 64 B = 1024 B = 8192 b

The size of data available to PTAs (excluding sector trailers) of an entire card is:

16 x 48 B = 768 B = 6144 b

2.3 Travel Card Layers

In [RKF-0020] the travel card is described as divided into four different layers:

- Card technology layer
- Card issuer layer
- Travel card support layer
- Travel card applications layer

The general description indicates, that these layers communicate in a vertical fashion. However, with the type 1 card technology used by this implementation specification, the different layers are used for conceptual purposes only. It is not possible to identify the 3 top layers physically on the card. Figure 1 below gives a more precise picture of the relation between the layers.

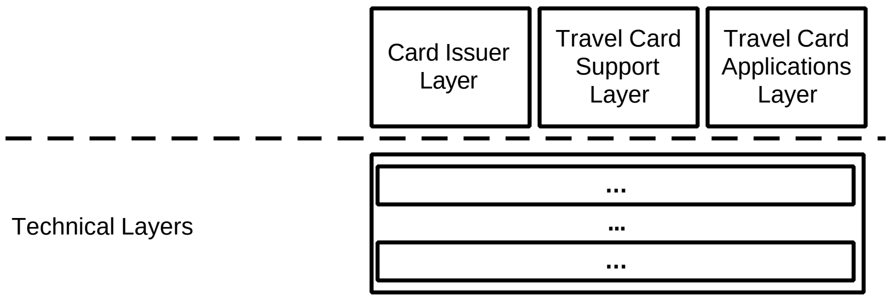


Figure 1: Card layer structure of RKF type 1 travel cards

Other card technologies might support the layers described in [RKF-0021] in a more direct way.

3 CARD ISSUER LAYER

This chapter concerns the contents and structure of the card issuer layer of the type 1 RKF travel card. The main purpose of the card issuer layer is to hold static information about the card which can be used in front systems to determine the proper handling of the card. The information stored in the card issuer layer is written once in the initialisation phase, and after that it shall not be modified. The information in this layer will make it possible for all PTAs to identify cards uniquely.

3.1 CMI: Card Manufacturer Information

This section describes the implementation of the travel Card Manufacturer Information.

3.1.1 CMI: Overview

The Card Manufacturer Information is the only system object in the Card Issuer Layer. The CMI holds the globally unique and unchangeable serial number of the card.

3.1.2 CMI: Requirements

The CMI meets the requirements specified in [RKF-0020] (section 'Card Issuer Layer') with the exception that the required unique identification of the card issuer and the card expiration date is not present. Instead this information is included in the *TCCI: Card Information* system object of the travel card support layer (data elements *CardProvider* and *CardValidityEndDate*).

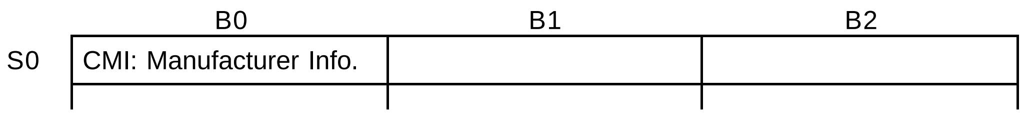
3.1.3 CMI: Contents

The CMI consists of one static data element group which is written during initialisation of the card, and not changed again throughout the lifetime of the card.

Id.	Data element group	Size	Main contents
None	CMI	1 block	Card serial number
			Manufacturer data

The detailed contents of the data element group is described in [RKF-0023].

The CMI occupies 1 block of the travel card, and it is always positioned in the first block of sector 0 (S0:B0). The following illustration outlines the positioning of the CMI system object on the travel card:



The CMI system object contain only one data element group, which is extraordinary in the sense that it contains neither identifier nor version number data elements. The reason for this is the static nature of the CMI as an object that will not be updated throughout the lifetime of the card.

The CMI contains a *CardSerialNumber* which uniquely identifies the card, a *CardSerialNumberCheckByte* used for security reasons and 11 one-byte Manufacturer Data data elements (*ManufacturerData1, ManufacturerData2* etc) which can be used for any information purposes deemed relevant by the manufacturer of the card.

3.1.4 CMI: Functions

3.1.4.1 CMI: Initiation Function

The initiation function writes the unique *CardSerialNumber*, the *CardSerialNumberCheckByte* and the *ManufacturerData* onto the travel card once and for all.

The initiation function consists of:

	Operation	Data element group	Size
1	Write	CMI: Manufacturer Information	1 block

3.1.4.2 CMI: Read Function

The read function provides information on the ID of the travel card and any manufacturer information of relevance.

The read function consists of:

	Operation	Data element group	Size
1	Read	CMI: Manufacturer Information	1 block

3.1.4.3 CMI: Update Function

The CMI can never be updated and therefore this function is not relevant.

3.1.5 CMI: Implementation considerations

It is permitted for the manufacturer to decide which information to place in the 11 bytes of *ManufacturerData*.

The *CardSerialNumber* must be unique and chosen from an interval decided by Resekortsföreningen.

4 TRAVEL CARD SUPPORT LAYER

This chapter describes the implementation of the travel card support layer on type 1 RKF travel cards.

This support layer contains system objects supporting the management of the application objects of the travel card applications layer.

4.1 TCCI: Travel Card Card Information

This section describes the implementation of the TCCI (Travel Card Card Information) system object, which is a part of the Travel Card Support Layer.

4.1.1 TCCI: Overview

The TCCI is a system object designed to hold information of the Travel Card that are of importance for all system objects and application objects. The elements in the TCCI contains information about how certain data elements in other system objects should be interpreted and what general rules should be used when reading information from the card. The TCCI holds the overall status and version number of the travel card.

4.1.2 TCCI: Requirements

The TCCI meets the requirements specified in [RKF-0020] (section ‘Travel Card Support Layer’). As explained in section 3.1.2, the TCCI contains information that according to [RKF-0020] should belong to the card issuer layer.

4.1.3 TCCI: Contents

The TCCI consists of one static element which is written during initialisation of the card. The contents of TCCI is not updated as a part of normal validation transactions.

Id.	Data element group	Size	Main contents
None	TCCI: Card Information	1 block	Card version number
			Card provider
			Card status
			Currency unit

The detailed contents of the system object is described in [RKF-0023].

The TCCI is always positioned in the second block of sector 0 (S0:B1) and it always uses 1 block of space. The following illustration shows the positioning of the TCCI system object on the travel card:

	B0	B1	B2
S0		TCCI: Card Information	

The TCCI system object contains only one data element group. It does not contain identifier or version number data elements due to the static nature of the TCCI.

MADInfoByte is 0 to indicate that MAD (Mifare Application Directory) is not available.

CardVersion indicates the Travel Card version of the actual card. This information is important in order for the front system to interpret the information on the card correctly.

CardProvider is the issuer of the travel card.

The date of expiry of the travel card is stated in the element named *CardValidityEndDate*.

During the use of the card the current status of the card is indicated by the element *CardStatus*. This element is used to indicate whether the card has been temporarily blocked or not.

The *CardCurrencyUnit* contains information of what currency and what unit of currency is used in all application objects of the card (cf. section 7.10).

4.1.4 TCCI: Functions

4.1.4.1 TCCI: Initiation Function

The initiation function creates the card information system object on the travel card and it consists of:

	Operation	Data element group	Size
1	Write	TCCI: Card Information	1 block

4.1.4.2 TCCI: Read Function

The read function provides relevant data to interpret system objects and application objects of the travel card.

The read function consists of:

	Operation	Data element group	Size
1	Read	TCCI: Card Information	1 block

4.1.4.3 TCCI: Update Function

The update function writes updated information about the travel card onto the card. Any change in the card information should be reflected in relevant system objects and application objects.

Updating the *Status* data element could be performed as a part of a normal validation transaction. Updating other data elements will involve updating most of the system objects and application objects of the card.

The update function consists of only one writing:

	Operation	Data element group	Size
1	Write	TCCI: Card Information	1 block

4.1.5 TCCI: Implementation considerations

When initiating a new travel card it must be decided what currency and unit to use. It must also be determined which version of RKF travel card to use.An expiry date should be decided.

4.2 TCDI: Travel Card Directory

This section describes the implementation of the travel card Directory.

4.2.1 TCDI: Overview

The travel Card Directory is a part of the mandatory part of the RKF travel card structure. It is needed to indicate the positioning of system objects and application objects on the card. Since each system object and application object starts at a sector boundary, the directory only contains information on a sector level.

As the layout of the first sector of the RKF Travel Card (sector 0) is fixed, it is only necessary for the directory to contain information about sectors 1 through 15. Each sector is described by an AID value and a PIX value.

4.2.2 TCDI: Requirements

The TCCI meets the requirements specified in [RKF-0020] (section 'TCDI: Travel Card Directory').

4.2.3 TCDI: Contents

The TCDI consists of a single data 3-block element group. Each block of the data element group holds the directory of 5 sectors, 1..5, 6..10, and 11..15 respectively. The TCDI is written during initialisation of the card, and when application objects are to or removed from the card.

Id.	**Data element group**	**Size**	**Main contents**
None	TCDI: Directory	3 blocks	Directory information for each sector

The detailed contents of the system object is described in [RKF-0023].

The TCDI is always positioned in the three blocks of sector 1 (S1:B0, S1:B1, S1:B2). The following illustration shows the positioning of the TCCI system object on the travel card:

	B0	B1	B2
S1	TCDI: Directory (1/3)	TCDI: Directory (2/3)	TCDI: Directory (3/3)

The TCDI does not contain an identifier or a version number data elements. If the interpretation of TCDI has to be changed, the *CardVersion* of TCCI must be increased.

The TCDI describes each sector x by a pair of *AID (sector x)* and *PIX (sector x)* data elements.

The general rule is that the directory entry of the first sector of an application object describes the application object. The AID value identifies the PTA issuing the application object. Within the scope of each AID value, PIX values identifies the specific type of application.

Some AID values and PIX values have special meaning. They are used to designate system objects etc.

The complete set of rules for interpreting the TCDI is described in section 7.5.

4.2.4 TCDI: Functions

4.2.4.1 TCDI: Initiation Function

The initiation function creates the Directory on the travel card:

	Operation	Data element group	Size
1	Write	TCDI: Directory	3 blocks

4.2.4.2 TCDI: Read Function

The TCDI should be read as a part of all transaction to determine which applications objects are present on the card, and the position of these objects. This information is used to determine which type of transaction to perform.

The read function consists of:

	Operation	Data element group	Size
1	Read	TCDI: Directory	3 blocks

The Directory is always read in its full extent.

4.2.4.3 TCDI: Update Function

The Directory is never updated during normal use. Whenever new applications are added, or applications are removed from the card, a new initiation function on the TCDI is performed.

4.2.5 TCDI: Implementation considerations

The Directory system object has no issues to be dealt with.

4.3 TCAS: Travel Card Applications Status

This section describes the implementation of the travel card Applications Status.

4.3.1 TCAS: Overview

The Applications Status system object is the primary means for achieving transaction to be completed indivisibly. In other words it is used for ensuring transactions not to be just partly completed. Furthermore, the Applications

Status is used for optimising the handling of the Event Log and a possible ticket/log area of the card.

4.3.2 TCAS: Requirements

The TCAS meets the requirements specified in [RKF-0020] (section 'TCAS: Travel Card Applications Status').

4.3.3 TCAS: Contents

The TCAS consists of two dynamic system objects having identical structure (*TCAS: Applications Status*). The two instances of the system object are called *Applications Status (1)* and *Applications Status (2)* or *TCAS (1)* and *TCAS (2)*.

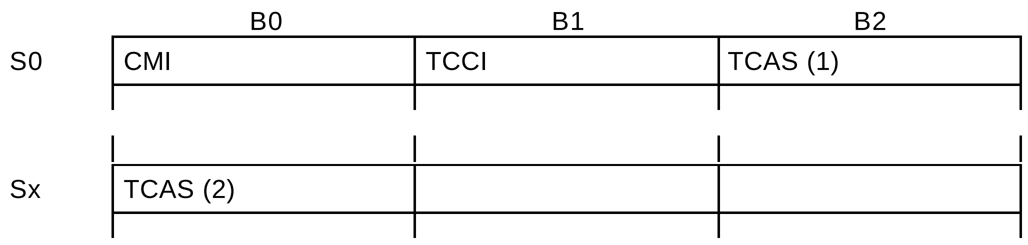
Id.	Data element group	Size	Main contents
A0 H	TCAS: Applications Status	1 block	General sector status
			Latest transaction number
			Latest event log record number
			Management of ticket/log area

The detailed contents of the system object is described in [RKF-0023].

The first instance *TCAS (1)* is always positioned in block S0:B2.

The second instance *TCAS (2)* is always positioned in the first block in a separate sector. In *TCDI: Directory* this sector must have AID value OD H. Blocks 1 and 2 of the sector will remain unused.

The following illustration outlines the positioning of the TCAS blocks:



The TCAS system object is divided into 5 groups of elements:

A. General Sector Status

B. Transaction Number

C. Event Log

D. Ticket/log Area

E. MAC

A. General Sector Status

A 2 bit *SectorStatus* for each sector of the card describes the status of the sector:

0: Undefined value. There are several possibilities:

- The sector is unused
- The sector is part of an application type AT3 (*TCEL: Event Log*)
- The sector is the first sector of an application type AT4
- The sector is the second or subsequent sector of an application

1, 2: The sector is the first sector of an application type AT1 or AT5. The value indicates, if the 1st or the 2nd dynamic element contains current data.

3: The sector is the first sector of an application type AT2

B. Transaction Number

TransactionNumber is the most recently used transaction number for sending data to the back office system describing card changes of this card. The number is increased cyclically every time the card is changed. See also section 7.1.

C. Event Log

EventLogRecordNumber is the number of the most recently used log record of *TCEL: Event Log*. See also section 5.2.

D. Ticket/Log Area

If the card contains an application of type AT2, these data elements manage the ticket/log area of sectors holding the application. Section 7.6 describes how this management is performed.

E. MAC

The MAC authenticator protects the contents of the system object.

4.3.4 TCAS: Functions

4.3.4.1 TCAS: Initiation Function

The initiation function writes a new set of identical applications status system objects onto the travel card.

The initiation function consists of:

	Operation	Data element group	Size
1	Write	TCAS: Travel Card Applications Status (1)	1 block
2	Write	TCAS: Travel Card Applications Status (2)	1 block

Related card changes:

- The contents of TCAS must be in accordance with all applications of the card

4.3.4.2 TCAS: Read Function

The read function provides information on the status of the applications of the travel card, e.g. which of the dynamic elements of TCPU holds the current value.

The read function consists of:

	Operation	Data element group	Size
1	Read	TCAS: Travel Card Applications Status (1)	1 block
2	Read	TCAS: Travel Card Applications Status (2)	1 block

The two instances TCAS (1) and TCAS (2) will be identical, if the previous transaction was completed normally. If, however, the previous transaction broke down while writing TCAS (1) or TCAS (2), the instances will be different. The interpretation must be done as specified:

```
if TCAS (1) is consistent:
    TCAS (1) is used for interpreting the card
else
    if TCAS (2) is consistent:
        TCAS (2) is used for interpreting the card
    else
        Error, the card is inconsistent
```

TCAS (x) is considered consistent, if the MAC value is correct, and all data elements have legal values (cf. section 7.8).

4.3.4.3 TCAS: Update Function

The update function writes the status of the applications of the travel card onto the card as the very last actions of a complete transaction. This update “commits” the transaction.

The update function consists of two writings, which must always be done in this sequence:

	Operation	Data element group	Size
1	Write	TCAS: Travel Card Applications Status (1)	1 block
2	Write	TCAS: Travel Card Applications Status (2)	1 block

Related card changes:

- The contents of TCAS must be in accordance with all application objects of the card

4.3.5 TCAS: Implementation considerations

When designing the travel card contents and structure it must be decided, which sector the second TCAS will be positioned in.

The part of TCAS managing the event log is only used, if the card has an event log.

The part of TCAS managing a ticket/log area is only used, if the card contains an application type AT2.

5 TRAVEL CARD APPLICATIONS LAYER

This chapter describes the implementation of the travel card applications layer on type 1 RKF travel cards. The applications layer is the layer in which PTAs can store their application objects to support the customers use of the card.

All information stored in this layer is arranged as one of the application types described in section 7.2, and it is either one of the below mentioned application objects or an application object designed by a PTA and approved by the RKF.

5.1 TCPU: Travel Card Purse

This section describes the implementation of the travel card Purse application object.

5.1.1 TCPU: Overview

The TCPU is an electronic purse holding a purse balance value and deposit value. Although the TCPU is issued by a specific PTA, it is intended to be used by many PTAs based on the interoperability agreements.

5.1.2 TCPU: Requirements

The TCPU meets the requirements specified in [RKF-0020] (section 'TCPU: Travel Card Purse').½

5.1.3 TCPU: Contents

The TCPU is implemented as a type AT1 application (cf. section 7.2), i.e. it has a static element (*TCPU: Static Data*) and 2 dynamic elements (*TCPU: Dynamic Data*), which alternately holds the current value of the purse.

Id.	Data element group	Size	Main contents
85 H	TCPU: Static Data	1 block	Issuing PTA
–	TCPU: Dynamic Data	1 block	Current purse value
			Current deposit value

The detailed contents of the system objects is described in [RKF-0023].

The TCPU occupies 1 sector of the travel card, and it can be positioned in any of the sectors 2..15. The following illustration outlines the positioning of the TCPU application object on the travel card:

	B0	B1	B2
Sx	TCPU: Static Data	TCPU: Dynamic Data (1)	TCPU: Dynamic Data (2)

5.1.3.1 TCPU: Static Data

The purse is identified by a serial number contained in the *PurseSerialNumber* data element. The static part of the TCPU also contains the data element *StartDate* which is the date from which the purse can be used.

The *DataPointer* data element can point to an optional PTA specific system object with complementary information.

5.1.3.2 TCPU: Dynamic Data

The current dynamic data element is determined by the relevant *SectorStatus* of *TCAS: Applications Status*.

PurseTransactionNumber contains a counter that is incremented each time a transaction is made involving the purse.

The purse is only valid until its expiry date which is written in the *EndDate* data element.

Value is the balance value of the purse. The value must be interpreted using the currency unit *CardCurrencyUnit* of *TCCI: Card Information* (cf. section 7.10). Note, that the purse is a 24 b signed money value as opposed to the default 20 b unsigned values used elsewhere on the travel card.

The status of the purse is expressed in the *Status* data element.

Deposit is the amount, the customer may have paid when obtaining the travel card. The value must be interpreted using the currency unit *CardCurrencyUnit* of *TCCI: Card Information* (cf. section 7.10).

5.1.4 TCPU: Functions

5.1.4.1 TCPU: Initiation Function

The initiation function writes a new purse onto the travel card.

The initiation function consists of:

	Operation	Application element	Size
1	Write	TCPU: Static Data	1 block
2	Write	TCPU: Dynamic Data	1 block

Related card changes:

- The entry of *TCDI: Directory* describing the sector of TCPU must contain:

  AID: 0B H

  PIX: 0

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of the TCPU must be set depending on which dynamic data element is written.

5.1.4.2 TCPU: Read Function

The read function provides the static values and the current dynamic values of the purse.

The read function consists of:

	Operation	Application element	Size
1	Read	TCPU: Static Data	1 block
2	Read	TCPU: Dynamic Data	1 block

5.1.4.3 TCPU: Update Function

The update function changes the value of the purse (or other dynamic values).

The update function consists of:

	Operation	Application element	Size
1	Write	TCPU: Dynamic Data	1 block

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of the TCPU must be set depending on which dynamic data element is written.

5.1.5 TCPU: Implementation considerations

When implementing the TCPU, the following issues must be considered:

- The sector number for the TCPU must be decided. Different sector numbers can be used for different types of cards within one PTA.
- The use of the PTA specific *DataPointer* of *TCPU: Static Data* must be decided. This decision must be co-ordinated with all PTAs using the TCPU.

5.2 TCEL: Travel Card Event Log

This section describes the implementation of the travel card Event Log application object.

5.2.1 TCEL: Overview

The Travel Card Event Log is an optional log application that can be used to store information about transactions performed on the card. This will enable a certain level of history function. The Event Log can be used to assist performing a roll-back of travel card transactions.

The log records are used cyclically.

Please note, that even if a travel card contains an event log, transactions are not forced to generate log records in the event log. I.e. the event log will not necessarily give a complete picture of the activities of a travel card.

5.2.2 TCEL: Requirements

The TCEL meets the requirements specified in [RKF-0020] (section ‘TCEL: Travel Card Event Log’).

5.2.3 TCEL: Contents

The TCEL is an AT3 application type consisting of a number of identically formatted event log records - each of a size of one block. If the TCEL occupies S sectors on the travel card, it will contain 3 x S log records. Each log record of the application object is describes as a date element group.

Id.	Data element group	Size	Main contents
84 H	TCEL: Event Log Record	1 block	Identification of PTA and device
			Event Code
			Data specific to event code

The detailed contents of the data element group is described in [RKF-0023].

The TCEL must be positioned as 1 to 5 consecutive sectors within sectors 2 to 15 of the travel card.

The following illustration outlines a possible positioning of TCEL, assuming that 3 sectors (i.e. 9 log records) are allocated:

	B0	B1	B2
Sx	TCEL Log Record (1)	TCEL Log Record (2)	TCEL Log Record (3)
S(x+1)	TCEL Log Record (4)	TCEL Log Record (5)	TCEL Log Record (6)
S(x+2)	TCEL Log Record (7)	TCEL Log Record (8)	TCEL Log Record (9)

The most recently used log record of TCEL is designated by *EventLogRecordNumber* of *TCAS: Applications Status*

The *EventDateTimeStamp* of each log record contains the date and time of the event described by the record.

Device identifies the front system performing the transaction and *DeviceTransactionNumber* is the transaction number of the device.

EventCode identifies the type of event, that is logged. The *EventData* is interpreted depending on the value of *EventCode*.

Possible event codes and matching event data is described in the following table:

Event code	Event	Event data
00 H	*None*	*Undefined event code*
01 H .. 04 H	Purchase of a ticket or contract	A pointer that references the issued ticket or contract and a price of the ticket
05 H	Ticket issued by a contract	A pointer to the ticket and a pointer to the contract that issued the ticket.
06 H	Validation of a ticket	A pointer to the ticket and PTA specific data
07 H	Extension of a ticket	A pointer to the ticket and PTA specific data
08 H	Charge of the TCPU	The amount that was charged to the TCPU
09 H .. 0F H	Removal of TCTI, TCCO, TCPU, TCST-ci/co, TCCP, TCDB, TCRE	Pointer to the sector (TicketPointer) that has been removed and PTA specific data
16 H	Initialisation of card	The amount that was paid as deposit for the travel card
17 H	An application object has been created.	Pointer to the sector of the new application object and PTA specific data
18 H	An application object has been repurchased or reimbursed	A pointer to the sector of the application object and the price
19 H	TCCO application activated	Pointer to the sector of the TCCO and PTA specific data
1A H	Paper ticket bought with TCPU	Pointer to the sector of the TCPU and the ticket price
1B H	TCST-ci/co check out validation	Pointer to the validated ticket and PTA specific data
1C H	Check out validation of TCTI / TCCO / TCST-ci/co	Pointer to the validated ticket and PTA specific data
1D H	Control validation of TCTI / TCCO / TCST-ci/co	Pointer to the validated ticket and PTA specific data
1E H	Excess fare validation of TCTI / TCCO / TCST-ci/co	Pointer to the validated ticket and PTA specific data

EventCode values are described in greater detail in section 8.4 and [RKF-0023].

5.2.4 TCEL: Functions

5.2.4.1 TCEL: Initiation Function

The initiation function writes all the empty log records on the card (i.e. *EventCode* 0).

The initiation function consists of, assuming that TCEL occupies N sectors on the travel card:

	Operation	Application element	Size
1	Write	TCEL: Event Log Records	N sectors

Related card changes:

- The entry of *TCDI: Directory* describing the sector of TCEL must contain:

  AID: 0A H

  PIX: 00 H (the first sector)

  PIX: 01 H (subsequent sectors)

- The *EventLogRecordNumber* of *TCAS: Applications Status* is set to 0.

5.2.4.2 TCEL: Read Function

The read function provides the contents of one or more already generated log records. Reading of log records is not expected to be a part of normal validation transactions. Instead the log records will be read for service purposes.

EventLogRecordNumber of *TCAS: Applications Status* is used to determine which log records to read.

To read one log record, the read function consists of:

	Operation	Application element	Size
1	Read	TCEL: Event Log Record	1 block

To read all log records, the read function consists of, assuming that TCEL occupies N sectors on the travel card:

	Operation	Application element	Size
1	Read	TCEL: Event Log Record	N sectors

5.2.4.3 TCEL: Update Function

The update function generates a new log record of the TCEL, thereby overwriting the oldest log record.

EventLogRecordNumber of *TCAS: Applications Status* is used to determine which log records to write.

The update function consists of:

	Operation	Application element	Size
1	Write	TCEL: Event Log Record	1 block

Related card changes:

- The *EventLogRecordNumber* of *TCAS: Applications Status* indicating the most recently used log record must be increased by 1.

5.2.5 TCEL: Implementation considerations

When implementing the TCEL, the following issues must be considered:

- The position and number of allocated sectors for the TCEL must be decided.
- The use of the PTA specific data element of *EventData* must be decided.

5.3 TCTI: Travel Card Ticket

This section describes the implementation of the travel card Ticket application object.

5.3.1 TCTI: Overview

The TCTI is an electronic ticket. The TCTI is constructed by a selection of application objects. Some of these elements are optional and some are mandatory. Some elements have to meet certain requirements as to where they can appear in the TCTI. The application object is designed by the PTA by putting together the elements described in this section.

5.3.2 TCTI: Requirements

The TCTI meets the requirements specified in [RKF-0020].

5.3.3 TCTI: Contents

The TCTI is implemented as a type AT1 application, i.e. it has a static element (*TCTI: Static Data*) and 2 dynamic elements (*TCTI: Dynamic Data*), which alternately holds the current dynamic values of the ticket.

The table below show the possible application elements in a TCTI application object and their primary constraints and limitations. The following abbreviations have been used in the two rightmost columns:

In the Constraints column (C):

M:	Mandatory
O:	Optional

① Exactly one of these elements (id. 84 H and 88 H) must be present. The element with identifier 88 H is used, when the dynamic part is bigger than one block.

In the Position column (P):

D:	Must be positioned in the dynamic part
S:	Must be positioned in the static part
D/S:	Can be positioned either in the dynamic or the static part
SF:	Must be positioned as first element in the static part
DL:	Must be positioned as the Last element in the dynamic part
DF:	Must be positioned as the first element in the dynamic part

Id.	Data element group	Size	Main contents	C	P
86 H	TCTI: Header Information	12 b	Version number Pointers to subsequent sectors	M	SF
88 H	Dynamic Information With More Than One Data Block	20 b	Transaction number	①	DF
89 H	TCTI/TCCO, Mandatory Data	88 b	AID and PIX of ticket Sale device	M	D/S
8A H	Dynamic Information With Only One Data Block	8 b	No data	①	DF
93 H	MAC Algorithm Identifiers	32 b	MAC data	M	DL
94 H	Administrative Price Information	28 b	Ticket price	O	D/S
95 H	Issuing Contract/Purse	12 b	Pointer to contract/purse	O	D/S
96 H	TCTI, Period of Validity	68 b	Valid time period	O	D/S
97 H	Validity, Distance	142 b	Valid origin and destination Valid distance Valid run Valid via points places	O	D/S
98 H	Validity, Route	32 b	Valid routes	O	D/S
99 H	Validity, Zone	34 b	Valid zones	O	D/S
9E H	TCTI, Issuing Information	38 b	Journey counters Date and time of validation	O	D/S
9B H	Customer Information	42 b	Customer number	O	D/S
9C H	Administrative Class Information	52 b	Passenger class Passenger group	O	D/S
9F H	Validation Information	14 b	Validation model Validation status	O	D/S

The detailed contents of the data element groups are described in [RKF-0023].

The following illustration outlines the positioning of a TCTI application object on the travel card, assuming that the size of the static data is 2 blocks, and that the size of the dynamic data is 3 blocks.

	B0	B1	B2
Sx	TCTI: Static Data (1/2)	TCTI: Static Data (2/2)	TCTI: Dynamic Data (1) (1/3)
S(x+1)	TCTI: Dynamic Data (1) (2/3)	TCTI: Dynamic Data (1) (3/3)	TCTI: Dynamic Data (2) (1/3)
S(x+2)	TCTI: Dynamic Data (2) (2/3)	TCTI: Dynamic Data (2) (3/3)	

5.3.3.1 TCTI: Header Information

VersionNumber is a unique identification of the version of all elements in the application object.

5.3.3.2 Dynamic Information With More Than One Data Block

This element is mutually exclusive with the element *Dynamic Information With Only One Data Block*. The element contains information about the structure of dynamic part of this TCTI and as a consequence, it must be positioned in the first block of the dynamic part.

ContractTransactionNumber is a counter that is increased by one for each transaction performed on the travel card involving this ticket application object.

5.3.3.3 TCTI/TCCO, Mandatory Data

SaleDevice is a data element that can contain the identification of the front system used for issuing the ticket.

ContractSerialNumber is a unique identification of this ticket.

Status is an indication of the current status of the ticket.

5.3.3.4 Dynamic Information With Only One Data Block

This element is used when a ticket consists of only one dynamic data block. This data element group is mutually exclusive with the above mentioned *Dynamic Information With More Than One Data Block*.

5.3.3.5 MAC Algorithm Identifiers

The MAC authenticator protects the contents of the ticket.

5.3.3.6 Administrative Price Information

The *Price* holds the ticket price.

5.3.3.7 Issuing Contract/Purse

If the ticket is issued from a contract, this element is a reference to the location of the issuing contract. Otherwise it is a reference to the purse used to pay for this ticket.

5.3.3.8 TCTI, Period of Validity

This element describes the validity of the Ticket in terms of *ValidityStartDate*, *ValidityStartTime*, *ValidityEndDate* and *ValidityEndTime*.

5.3.3.9 Validity, Distance

If present, this element specifies the valid area for this ticket. The area is described by a combination of the following:

- *JourneyOriginPlace* and *JourneyDestinationPlace*
- Maximum allowed distance in *JourneyDistance*
- The specific *JourneyRun* of the vehicle
- Up to two valid via places, *JourneyVia1* and *JourneyVia2PLace*
- *JourneyInterchange* containing the maximum number of interchanges permitted on the journey

5.3.3.10 Validity, Route

This information can be repeated up to 10 times on the travel card, and consists of a *JourneyRouteNumber* paired with a preceding *JourneyRouteAID* data element indicating the owner of the route data element.

5.3.3.11 Validity, Zone

This information can be repeated up to 10 times on the travel card, and consists of a *ValidityZonePlace* paired with a preceding *ValidityZoneAID* data element indicating the owner of the zone data element.

5.3.3.12 TCTI, Issuing Information

The two data elements *ValidationLastDate* and *ValidationLastTime* indicate the last date and the last time the ticket was used.

5.3.3.13 Customer Information

The *CustomerNumber* identifies the customer, that owns this ticket application object.

5.3.3.14 Administrative Class Information

This element contains a *PassengerClass* data element indicating the class of travel for the passenger group of the ticket. The passenger group is specified by 3 *PassengerSubGroups* each consisting of a *PassengerType* data element indicating a type of passenger paired with a *PassengerTotal* data element stating the number of passengers of that particular type.

5.3.3.15 Validation Information

The data element *ValidationModel* indicates, if the ticket is used for the ci/co (check-in/check-out) or the ci-dest (check-in destination) validation model.

If ci/co validation i used, *ValidationStatus* indicates if the ticket is currently in an ‘open’ state (after check-in) or a ‘closed’ state (after check-out).

The *ValidationLevel* data element can be used by the PTA to restrict the use of the ticket.

5.3.4 TCTI: Functions

When describing the functions of TCTI, it is assumed that the size of the static data is N_S blocks, while the size of the dynamic data is N_D blocks.

5.3.4.1 TCTI: Initiation Function

The initiation function writes a new ticket onto the travel card.

The initiation function consists of:

	Operation	**Application element**	**Size**
1	Write	TCTI: Static Data	N_S blocks
2	Write	TCTI: Dynamic Data	N_D blocks

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of TCTI must be set depending on which data element is written.

5.3.4.2 TCTI: Read Function

The read function provides the static values and the current dynamic values of the ticket.

The read function consists of:

	Operation	Application element	Size
1	Read	TCTI: Static Data	N_S blocks
2	Read	TCTI: Dynamic Data	N_D blocks

5.3.4.3 TCTI: Update Function

The update function changes values of the dynamic part of the ticket.

The update function consists of:

	Operation	Application element	Size
1	Write	TCTI: Dynamic Data	N_D blocks

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of TCTI must be set depending on which data element is written.

5.3.5 TCTI: Implementation considerations

When designing a TCTI application object, the PTA can choose which of the above mentioned data element groups to use. The restrictions described in section 5.3.3 must be fulfilled.

5.4 TCCO: Travel Card Contract

This section describes the implementation of the travel card Contract application object.

5.4.1 TCCO: Overview

The TCCO is an electronic travel contract. The TCCO is constructed by a selection of data element groups. Some of these elements are optional and some are mandatory. Some elements have to meet certain requirements as to where they can appear in the TCCO. The application object is designed by the PTA by putting together the elements described in this section.

5.4.2 TCCO: Requirements

The TCCO meets the requirements specified in [RKF-0020].

5.4.3 TCCO: Contents

The TCCO is implemented as a type AT1 application, i.e. it has a static element (*TCCO: Static Data*) and 2 dynamic elements (*TCCO: Dynamic Data*), which alternately holds the current dynamic values of the ticket.

The table below show the possible data element groups in a TCCO application object and their primary constraints and limitations. The following abbreviations have been used in the two rightmost columns:

In the Constraints column (C):

M:	Mandatory
O:	Optional
①	Exactly one of these elements (id. 84 H and 88 H) must be present. The element with identifier 88 H is used, when the dynamic part is bigger than one block.

In the Position column (P):

D:	Must be positioned in the dynamic part
S:	Must be positioned in the static part
D/S:	Can be positioned either in the dynamic or the static part
SF:	Must be positioned as first element in the static part
DL:	Must be positioned as the Last element in the dynamic part
DF:	Must be positioned as the first element in the dynamic part

Id.	Data element group	Size	Main contents	C	P
87 H	TCCO: Header Information	14 b	Version number Pointers to subsequent sectors	M	SF
88 H	Dynamic Information With More Than One Data Block	20 b	Transaction counter	➀	DF
89 H	TCTI/TCCO, Mandatory Data	88 b	AID and PIX of contract Sale device	M	D/S
8A H	Dynamic Information With Only One Data Block	8 b	No data	➀	DF
93 H	MAC Algorithm Identifiers	32 b	MAC data	M	DL
94 H	Administrative Price Information	28 b	Contract price	O	D/S
95 H	Issuing Contract/Purse	12 b	Pointer to contract/purse	O	D/S
9D H	TCCO, Period of Validity	114 b	Valid time period	O	D/S
97 H	Validity, Distance	142 b	Valid origin and destination Valid distance Valid run Valid via points places	O	D/S
98 H	Validity, Route	32 b	Valid routes	O	D/S
99 H	Validity, Zone	34 b	Valid zones	O	D/S
9E H	TCCO, Issuing Information	54 b	Journey counters Date and time of validation	O	D/S
9B H	Customer Information	42 b	Customer number	O	D/S
9C H	Administrative Class Information	52 b	Passenger class Passenger group	O	D/S
9F H	Validation Information	14 b	Validation model Validation status	O	D/S

The detailed contents of the data element groups are described in [RKF-0023].

The following illustration outlines the positioning of a TCCO application object on the travel card, assuming that the size of the static data is 2 blocks, and that the size of the dynamic data is 3 blocks.

	B0	B1	B2
Sx	TCCO: Static Data (1/2)	TCCO: Static Data (2/2)	TCCO: Dynamic Data (1) (1/3)
S(x+1)	TCCO: Dynamic Data (1) (2/3)	TCCO: Dynamic Data (1) (3/3)	TCCO: Dynamic Data (2) (1/3)
S(x+2)	TCCO: Dynamic Data (2) (2/3)	TCCO: Dynamic Data (2) (3/3)	

5.4.3.1 TCCO: Header Information

VersionNumber is a unique identification of the version of all elements in the application object.

5.4.3.2 Dynamic Information With More Than One Data Block

This element is mutually exclusive with the element *Dynamic Information With Only One Data Block*. The element contains information about the structure of dynamic part of this TCTI and as a consequence, it must be positioned in the first block of the dynamic part.

ContractTransactionNumber is a counter that is increased by one for each transaction performed on the travel card involving this ticket application object.

5.4.3.3 TCTI/TCCO, Mandatory Data

SaleDevice is a data element that can contain the identification of the front system used for issuing the ticket.

ContractSerialNumber is a unique identification of this ticket.

Status is an indication of the current status of the ticket.

5.4.3.4 Dynamic Information With Only One Data Block

This element is used when a ticket consists of only one dynamic data block. This data element group is mutually exclusive with the above mentioned *Dynamic Information With More Than One Data Block*.

5.4.3.5 MAC Algorithm Identifiers

The MAC authenticator protects the contents of the ticket.

5.4.3.6 Administrative Price Information

The *Price* holds the contract price.

5.4.3.7 Issuing Contract/Purse

If the contract is issued from another contract, this element is a reference to the location of the issuing contract. Otherwise it is a reference to the purse used to pay for this contract.

5.4.3.8 TCCO, Period of Validity

This element describes the validity of the Ticket in terms of *ValidityStartDate*, *ValidityStartTime, ValidityEndDate* and *ValidityEndTime*.

Furthermore, *ValidityDuration* can specify the length of a period, and *ValidityLimitDate* the end of a period. *PeriodJourneys* restricts the number of allowed journeys, and *RestrictDay* and *RestrictTimeCode* can restrict the valid days and time for using the contract.

5.4.3.9 Validity, Distance

If present, this element specifies the valid area for this ticket. The area is described by a combination of the following:

- *JourneyOriginPlace* and *JourneyDestinationPlace*
- Maximum allowed distance in *JourneyDistance*
- The specific *JourneyRun* of the vehicle
- Up till two valid via places, *JourneyVia1* and *JourneyVia2Place*
- *JourneyInterchange* containing the maximum number of interchanges permitted on the journey

5.4.3.10 Validity, Route

This information can be repeated up to 10 times on the travel card, and consists of a *JourneyRouteNumber* paired with a preceding *JourneyRouteAID* data element indicating the owner of the route data element.

5.4.3.11 Validity, Zone

This information can be repeated up to 10 times on the travel card, and consists of a *ValidityZonePlace* paired with a preceding *ValidityZoneAID* data element indicating the owner of the zone data element.

5.4.3.12 TCCO, Issuing Information

ValidationTotalIssuedJourneys and *ValidationTotalIssuedJourneyswithinPeriod* can be used to restrict the allowed number of journeys.

The two data elements *ValidationLastDate* and *ValidationLastTime* indicate the last date and the last time the ticket was used.

5.4.3.13 Customer Information

The *CustomerNumber* identifies the customer, that owns this contract application object.

5.4.3.14 Administrative Class Information

This element contains a *PassengerClass* data element indicating the class of travel for the passenger group of the ticket. The passenger group is specified by 3 *PassengerSubGroups* each consisting of a *PassengerType* data element indicating a type of passenger paired with a *PassengerTotal* data element stating the number of passengers of that particular type.

5.4.3.15 Validation Information

The data element *ValidationModel* indicates, if the ticket is used for the ci/co (check-in/check-out) or the ci-dest (check-in destination) validation model.

If ci/co validation i used, *ValidationStatus* indicates if the ticket is currently in an ‘open’ state (after check-in) or a ‘closed’ state (after check-out).

The *ValidationLevel* data element can be used by the PTA to restrict the use of the ticket.

5.4.4 TCCO: Functions

When describing the functions of TCCO, it is assumed that the size of the static data is N_S blocks, while the size of the dynamic data is N_D blocks.

5.4.4.1 TCCO: Initiation Function

The initiation function writes a new contract onto the travel card.

The initiation function consists of:

	Operation	**Application element**	**Size**
1	Write	TCTI: Static Data	N_S blocks
2	Write	TCTI: Dynamic Data	N_D blocks

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of TCTI must be set depending on which data element is written.

5.4.4.2 TCCO: Read Function

The read function provides the static values and the current dynamic values of the contract.

The read function consists of:

	Operation	Application element	Size
1	Read	TCTI: Static Data	N_S blocks
2	Read	TCTI: Dynamic Data	N_D blocks

5.4.4.3 TCCO: Update Function

The update function changes values of the dynamic part of the contract.

The update function consists of:

	Operation	Application element	Size
1	Write	TCTI: Dynamic Data	N_D blocks

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of TCTI must be set depending on which data element is written.

5.4.5 TCCO: Implementation considerations

When designing a TCCO application object, the PTA can choose which of the above mentioned data element groups to use. The restrictions described in section 5.4.3 must be fulfilled.

5.5 TCST-ci/co: Travel Card Special Ticket (ci/co)

In this section the Special Ticket application object for ci/co validation is described.

5.5.1 TCST-ci/co: Overview

The TCST is a special ticket application object, designed to be used with ci/co (check-in/check-out) validation. The application object can also be used for ci-dest (check-in destination) validation. Ci/co validation implies, that the passenger must perform a check-in validation each time a journey or a sub journey is commenced and a check-out validation when the journey comes to an end.

5.5.2 TCST-ci/co: Requirements

The TCST-ci/co meets the requirements specified in [RKF-0020].

5.5.3 TCST-ci/co: Contents

The TCST-ci/co can be implemented either as a type AT2 application or as a type AT5 application.

Below is a table showing the contents of the TCST-ci/co application object.

Id.	Data element group	Size	Main contents
A3 H	TCST-ci/co	1 sector	Passenger group Validation model and validation status Price of the ticket Time and place for origin place and destination place Excess fare information

Below is an illustration of a typical TCST-ci/co application object implemented as a type AT2 application with 2 two tickets alternately holding the current ticket, and 4 tickets used for log purposes:

	B0	B1	B2
Sx	TCST-ci/co (1) (1/3)	TCST-ci/co (1) (2/3)	TCST-ci/co (1) (3/3)
S(x+1)	TCST-ci/co (2) (1/3)	TCST-ci/co (2) (2/3)	TCST-ci/co (2) (3/3)
S(x+2)	TCST-ci/co (3) (1/3)	TCST-ci/co (3) (2/3)	TCST-ci/co (3) (3/3)
S(x+3)	TCST-ci/co (4) (1/3)	TCST-ci/co (4) (2/3)	TCST-ci/co (4) (3/3)
S(x+4)	TCST-ci/co (5) (1/3)	TCST-ci/co (5) (2/3)	TCST-ci/co (5) (3/3)
S(x+5)	TCST-ci/co (6) (1/3)	TCST-ci/co (6) (2/3)	TCST-ci/co (6) (3/3)

The TCST-ci/co Application object is divided into 8 groups of data elements:

A. Passenger Group

B. Validation

C. Price

D. Journey Information

E. Excess Fare

F. Control

G. MAC

A. Passenger Group

A *PassengerClass* data element indicates the class of travel for the passenger group of the ticket. The passenger group is specified by 3 *PassengerSubGroups* each consisting of a *PassengerType* data element indicating a type of passenger paired with a *PassengerTotal* data element stating the number of passengers of that particular type.

B. Validation

The data element *ValidationModel* indicates, if the ticket is used for the ci/co (check-in/check-out) or the ci-dest (check-in destination) validation model.

If ci/co validation i used, *ValidationStatus* indicates if the ticket is currently in an ‘open’ state (after check-in) or a ‘closed’ state (after check-out).

The *ValidationLevel* data element can be used by the PTA to restrict the use of the ticket.

C. Price

The *Price* data element is the current price of the journey. If *ValidationStatus* is ‘Open’, the price can be a maximum price, and the price will be calculated correctly, when *ValidationStatus* becomes ‘Closed’.

PriceModificationLevel can be used to modify the price up or down.

D. Journey Information

This section of data holds information on the travelled journey:

- *JourneyOriginPlace* and *JourneyDestinationPlace* designating the start and end places of the journey
- *JourneyFurthestPlace* is the validation place furthest away from *JourneyOriginPlace*
- Date and time of *JourneyOriginPlace, JourneyDestinationPlace, and JourneyFurthestPlace*

E. Excess Fare

The *ExcessFareStatus* data element indicates whether or not the current journey includes distances where an excess fare should be added to the normal price of the journey. When a sub journey is in progress and it includes counting distance for excess fare calculation, the *ExcessFareOrigintPlace* data element contains the check-in place, at which the excess fare distance counting began. Each time an excess fare sub journey ends ,the accumulated distance measured in kilometers is added to the *ExcessFareDistance* data element.

F. Control

The data elements *LatestControlPlace* and *LatestControlTime* hold information of the latest ticket control performed by personel.

G. MAC

The MAC authenticator protects the contents of the Application object.

5.5.4 TCST-ci/co: Functions

5.5.4.1 TCST-ci/co: Initiation Function

The initiation function writes a new TCST-ci/co application object onto the travel card. The number of tickets allocated is decided by the PTA.

The initiation function consists of:

	Operation	Application element	Size
1	Write	TCST-ci/co	N sectors

Related card changes:

- The ticket/log area data elements of *TCAS: Applications Status* describing the TCST-ci/co must be set.

5.5.4.2 TCST-ci/co: Read Function

The read function provides the current values of the TCST.

The read function consists of:

	Operation	Application element	Size
1	Read	TCST-ci/co (current)	1 sector

Section **Fel! Okänt växelargument.** describes how TCAS determines which sector to read.

5.5.4.3 TCST-ci/co: Update Function

The update function writes a new current TCST-ci/co ticket to the ticket/log area,.

The update function consists of:

	Operation	Application element	Size
1	Write	TCST-ci/co (current)	1 sector

Section 7.6 describes how TCAS determines which sector to write, and how to update TCAS.

Related card changes:

- The ticket/log area data elements of *TCAS: Applications Status* describing the TCST-ci/co must be updated.

5.5.5 TCST-ci/co: Implementation considerations

A TCST-ci/co application object can only be either a AT2 or a AT5 application and it will therefore have a minimum size of two full sectors if implemented as an AT5 application. If an AT 2 application is chosen, 3 to 8 sectors must be allocated, resulting in 1 to 6 tickets used as a log.

5.6 TCCP: Travel Card Customer Profile

5.6.1 TCCP: Overview

The travel card Customer Profile application object is used to describe the customer or the group of passengers for whom the travel card has been issued.

5.6.2 TCCP: Requirements

The TCCP meets the requirements specified in [RKF-0020].

5.6.3 TCCP: Contents

The TCCP application object can be implemented only as a type AT4 application since it contains only a static part, and is not updated in connection with regular validation transactions. It contains only one element: Customer Profile (A2 H). The table below shows the data elements included in this element.

Id.	Data element group	Size	Main contents
A2 H	Customer Profile	2 sectors	Personal information and preferences of the customer Default passenger group

The detailed contents of the Application object are described in [RKF-0023].

Below is an illustration of a typical TCCP implementation.

	B0	B1	B2
Sx	TCCP (1/2)	TCCP (2/2)	

The TCCP application object describes a default passenger group. A *PassengerClass* data element indicates the default class of travel for the passenger group of the ticket. The passenger group is specified by 3 *PassengerSubGroups* each consisting of a *PassengerType* data element indicating a type of passenger paired with a *PassengerTotal* data element stating the number of passengers of that partic ular type.

The default *ValidationLevel* is included.

The *Birthday* data element contains the month of birthday for the customer. This data element will enable the PTA to suggest changes to the card, when the customer changes from e.g. child to adult.

In order to present information in the correct language the *Language* data element indicates the language preferred by the customer.

The *DialoguePreferences* data element can contain preferences for the customer when it comes to the criteria for communicating with the travel card units. Possible use could be special considerations towards hearing impaired or visually impaired customers.

SubscriptionOrCreditCompany identifies the company with which the customer deals when settling his account.

The customers chosen type of subscription or credit is indicated by the *SubscriptionOrCreditType* data element.

5.6.4 TCCP: Functions

5.6.4.1 TCCP: Initiation Function

The initiation function writes a new customer profile onto the travel card.

The initiation function consists of:

	Operation	Application element	Size
1	Write	TCCP	2 blocks

5.6.4.2 TCCP: Read Function

The read function provides the static values and the current dynamic values of the purse.

The read function consists of:

	Operation	Application element	Size
1	Read	TCCP	2 blocks

5.6.5 TCCP: Implementation considerations

The PTA must decide at which sector to position the TCCP.

5.7 TCDB: Travel Card Discount Basis

This section describes the travel card Discount Basis application object used to store information about the level of discount earned by the customer.

5.7.1 TCDB: Overview

The travel card Discount Basis application object enables a PTA to store information of a customers use of a travel card over a three month period of time, thereby enabling the PTA to grant discount based on previous use.

5.7.2 TCDB: Requirements

The TCDB meets the requirements specified in [RKF-0020].

5.7.3 TCDB: Contents

When creating a TCDB application object it can be structured as a type AT1 application consisting of a static part and two dynamic parts.

Id.	Data element group	Size	Main contents
A1 H	Discount Basis, Static Data	1 block	Three discount types
–	Discount Basis, Dynamic Data	1 block	Three discount basis blocks

The detailed contents of the Application elements is described in [RKF-0023].

The TCDB occupies 1 sector of the travel card, and it can be positioned in any of the sectors 2..15. The following illustration outlines the positioning of the TCDB application object on the travel card:

	B0	B1	B2
Sx	TCDB: Static Data	TCDB: Dynamic Data (1)	TCDB: Dynamic Data (2)

5.7.3.1 Discount Basis, Static Data

The static part of the TCDB contains three data elements *DiscountType (1)*, *DiscountType (2)* and *DiscountType (3)* which contains an ID that respectively determine the type of discount in the three *DiscountBasisBlock*s. It is for the PTA to decide which discount types should be used in the application object.

5.7.3.2 Discount Basis, Dynamic Data

The dynamic element of the TCDB application object contains a *FirstMonth* data element that contains an ID of the month for which the accumulated discount values apply.

The TCDB application object stores information of accumulated discount for up to three types of discounts stored in the set of data elements *DiscountBasisBlock (1)*, *DiscountBasisBlock (2)* and *DiscountBasisBlock (3)*. It is up to the PTA to decide how these blocks are interpreted.

Each of these sets of data elements consist of four data elements: *DicountCounterJourneys* which is a counter that accumulates the number of journeys during the month of interest, a *DiscountCounterDistance* that contains the accumulated amount of kilometers traveled, and three data elements *DiscountLevel (1)*, *DiscountLevel (2)* and *DiscountLevel (3)* describing the obtained level of discount for each of the three months preceding the month for which the two above mentioned counters accumulate information.

5.7.4 TCDB: Functions

5.7.4.1 TCDB: Initiation Function

The initiation function writes a new purse onto the travel card.

The initiation function consists of:

	Operation	Application element	Size
1	Write	TCDB: Static Data	1 block
2	Write	TCDB: Dynamic Data	1 block

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of TCDB must be set depending on which dynamic data element is written.

5.7.4.2 TCDB: Read Function

The read function provides the static values and the current dynamic values of the purse.

The read function consists of:

	Operation	Application element	Size
1	Read	TCDB: Static Data	1 block
2	Read	TCDB: Dynamic Data	1 block

5.7.4.3 TCDB: Update Function

The update function changes the value (or other dynamic values) of the purse.

The update function consists of:

	Operation	Application element	Size
1	Write	TCDB: Dynamic Data	1 block

Related card changes:

- The *SectorStatus* of *TCAS: Applications Status* describing the sector of TCDB must be set depending on which dynamic data element is written.

5.7.5 TCDB: Implementation considerations

The sector number for the TCDB must be decided. Different sector numbers can be used for different types of cards within one PTA.

5.8 TCRE: Travel Card Reservation

The TCRE application object is not described in this specification, but the name is reserved for later use.

6 SECURITY

The card technology of type 1 RKF travel card is contactless IC-cards according to [ISO/IEC 14443], type A, 1 kB (e.g. Mifare® Standard).

Existing and relevant security mechanisms of this card technology shall be implemented.

Additional security requirements defined in [RKF-0020] shall be implemented on top of the security architecture of the card technology.

6.1 Smart card security

6.1.1 Personalisation

The normal personalisation of traditional ISO 7816 smart cards is not applicable for the RKF type 1 card [ISO 7816-1], [ISO 7816-2], [ISO 7816-3], [ISO 7816-4], [ISO 7816-5], [ISO 7816-6].

After the personalisation of the type 1 card, the card shall include:

Mandatory:

- *CMI: Card Manufacturer Information*
- *TCCI: Card Information*
- *TCDI: Card Directory*
- *TCAS: Applications Status* (2 instances)
- Relevant access conditions:

  Keys that are common for the RKF travel card needs to be defined for the common areas on the card. All the transport keys of the card should be exchanged. Relevant access rights should be defined so that authorised users can read/write information to the card and free areas can be used.

Optional:

- Application objects are optional subject to the rules of section 7.3.1

6.1.2 Secure messaging

[ISO/IEC 9798-2] is used for authentication and encryption.

6.1.3 Authentication

[ISO/IEC 9798-2] is used.

6.1.4 Access Authorisation

Two or several application objects must never be stored in one sector because the card technology only permits blocking on sector level.

This access authorisation is divided into two levels so that one authentication can permit only reading and the other authentication can permit both reading and writing.

When relevant and possible, the keys used for access authorisation should be used separately:

One key should be used for read access and one key should be used for write access. This should be specified per block.

6.1.5 Card Blocking

The card is blocked and unblocked by setting the value *CardStatus* in the *TCCI: Card Information* and recalculate the MAC associated with this file.

6.1.6 Key Management

Every information on the card, for example the TCPU, the TCDI and the TCEL shall be stored on separate sectors and also protected with its' own keys. The keys can however not be different per card. Every TCPU on every card has to be protected with the same keys. This implies that every PTA using the TCPU or any other common information on the card need to share keys with each other.

The keys that protect the information should be exchanged regularly and in a very secure way.

At least three generations of keys should be stored in the card accepting device (CAD) for each relevant system object and application object.

6.2 Information Security Requirements

6.2.1 User Authentication and Data Integrity

Every system object and application object on the travel card that includes a MAC (Message Authentication Code) shall use it. The MAC should be verified every time the information is read and used.

The MAC shall be recalculated every time the system object or application object is changed.

Every information on the card shall have its own master key. The key used to calculate the MAC on an information should be unique per type of information, card and sectors.

desMAC [FIPS 113] should be used to calculate the unique key.

desMAC [FIPS 113] should be use to calculate the MAC.

The master key should be changed regularly and in a very secure way.

At least three generations of master keys should be stored and available in the back office application.

The master key should at least be protected with a password when it is stored in software.

6.2.2 Transaction Security

The type 1 card technology itself has no guarantees that all data in the transaction is written on the card. To prevent that information is lost during a transaction all data that shall be updated and stored in the dynamic data are for the relevant system object or application object.

This solution combined with the functionality of *TCAS: Applications Status* makes it possible to ensure that transactions are performed indivisibly. This subject is treated in detail in section 7.1.

The directory function is a dynamic data area that is not stored twice on the card. If the directory is corrupt, reading the first block in each sector where relevant directory information also is stored can create a new directory table.

7 IMPLEMENTATION RULES

7.1 Transaction Concept

7.1.1 Transaction Fundamentals

As defined in the glossary of [RKF-0017], a travel card transaction is the sum of all elementary actions necessary to complete a desired and controlled change of the card. Examples of travel card transactions are:

- Card initialisation
- Check-in validation

A travel card transaction consists of a number of elementary actions:

- Card reads
- Data processing - e.g. distance and fare calculation
- Card writes

Section 'General Transaction Procedure' of [RKF-0024] suggests how the sequence of elementary actions can be organized.

7.1.2 Indivisible Transactions

A transaction will read and update various application objects. The card technology of type 1 travel cards does not guarantee, that the card writes are performed successfully. The communication between the travel card and the CAD can break at any moment, either because the customer removes the card from the read/write data element of the CAD too early, or because the communication is disturbed. Even the writing of one single block on the card can break down resulting in undefined contents of the block.

If sufficient measures are not taken, the travel card can easily end up in an undefined and useless state, if a transaction is not completed successfully.

Therefore, it is necessary to ensure that each transaction is completed *indivisibly*, i.e. either all updates of application objects are completed, or none of these updates are completed.

The following principles are utilized to ensure indivisible transactions:

- While performing a transaction, new contents of application objects is written in separate blocks of the travel card, so that the previous contents is not overwritten or otherwise changed.
- One system object, the 1-block *TCAS: Applications Status* holds information on the position of the valid data of all application objects.
- As the writing of a TCAS block can break down, the TCAS is duplicated in two instances (TCAS (1) and TCAS (2)). If transactions are completed

normally, the two instances will be identical. The two instances of TCAS are written as the very last blocks of all transactions updating the card as described in section 4.3.4.3.

- When reading a travel card, the two instances of TCAS must be read before application objects are read. If the two instances of TCAS are different, the instance to use for interpreting the application objects is determined as described in section 4.3.4.2.

To obtain interoperability, all PTAs have to follow these principles.

Application objects are structured as one of the application types AT1, ..., AT5 (cf. section 7.2). Each application type describes how the two instances of dynamic contents of the application object are handled.

The following example shows the consequences of a transaction breaking down at different times during the writing of data to the travel card. In this example, data is written to TCPU, TCTI, and TCAS.

Time of transaction breakdown	Consequences
Writing TCPU or TCTI	The transaction is not completed. Next transaction will use TCAS (1).
Writing of TCAS (1)	The transaction is not completed. By the start of the next transaction, TCAS (1) will be inconsistent, i.e. TCAS (2) is used.
Writing of TCAS (2)	The transaction is completed. By the start of the next transaction, TCAS (2) will be inconsistent, i.e. TCAS (1) is used.
No breakdown	The transaction is completed. Next transaction will use TCAS (1) (identical to TCAS (2)).

If the travel card has become inconsistent, it may in some instances be possible to re-establish the consistency of the card by qualified personnel using special equipment.

7.1.3 Transaction Numbers

Different types of transaction numbers are used to identify transactions:

Transaction number	Data element	Usage
Card transaction number	*TransactionNumber* of *TCAS*	Identifies complete transactions. The transaction number is local to each travel card.
Device transaction Number	*DeviceTransactionNumber* of *TCEL: Event Log Record*	Identifies transactions completed by a front system. The transaction number is local to each front system device.
Purse transaction Number	*PurseTransactionNumber* of *TCPU: Dynamic Data*	Identifies purse transactions. The transaction number is local to each TCPU.
Ticket/contract transaction Number	*ContractTransactionNumber* of *Dynamic Information With More Than One Data Block*	Identifies ticket/contract transactions. The transaction number is local to each TCTI and TCCO.

Transaction numbers are generally transmitted from front systems to the back office system as a part of the data describing each transaction. In the back office system the transaction numbers are used to ensure safe handling of transactions.

The card transaction number of TCAS is used by the back office system to:

- Perform simple and secure identification of the transaction in the back office database of all transactions.
- Identify transactions, that erroneously are duplicated.
- Identify transactions, that erroneously are not received by the back office system.
- Secure sequential reception of transactions from front system to the back office system.
- Identify illegally copied travel cards.

The card transaction number of TCAS is a relatively short number (8 b). The 256 different values are used cyclically, and the back office system must use the date of transactions to differentiate two transactions having the same transaction number.

7.2 Types of Application Objects

Travel card application objects are categorised into 5 types:

AT1: The application consists of one static application element and two dynamic application elements. The two dynamic application elements alternately contains the current contents of the application.

AT2: The application consists of three or more dynamic application elements. Two of the application elements alternately contains the current contents of the application. The rest of the application elements constitute a log of old application elements. The sectors of the travel card occupied by an AT2 application is called a ticket/log area.

AT3: This is the Event Log (TCEL) application consisting of a number of one block log records used cyclically.

AT4: The application consists of one static application element. (AT4 is a special case of AT1 without dynamic application elements.)

AT5: The application consists of two dynamic application elements. The two dynamic application elements alternately contains the current contents of the application. (AT5 is a special case of AT1 without a static application element; AT5 is a special case of AT2 with only two application elements.)

Each type of application type is managed differently as explained in the following sections.

7.2.1 Application Type AT1

The application consists of:

- One static application element. The contents of this element is only defined or changed by an initiation transaction.
- Two dynamic application elements with identical data structure. The two dynamic application elements alternately contains the current contents. The contents can be changed by a validation transaction.

The current dynamic element is determined by the relevant *SectorStatus* of *TCAS: Applications Status*, i.e. the *SectorStatus* representing for the first sector of the application.

A travel card may have any number of applications of type AT1.

Examples of applications of type AT1:

- TCPU
- TCTI

The following Figure 2 illustrates an application of type AT1:

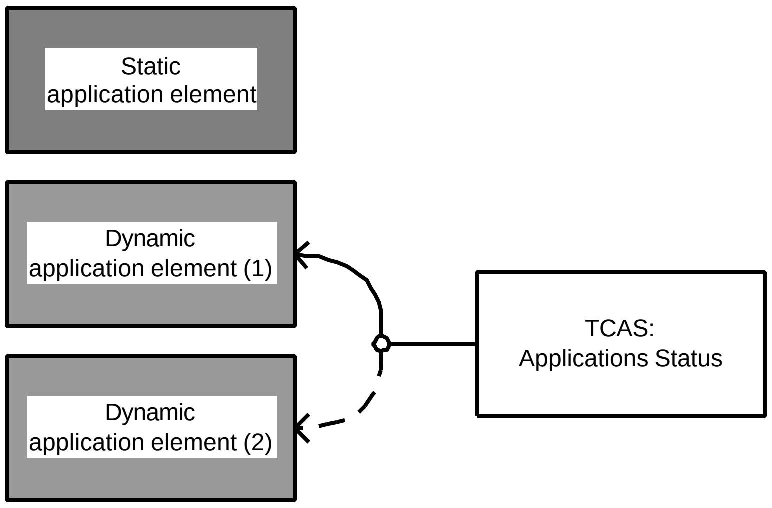


Figure 2: Application Type 1

7.2.2 Application Type AT2

The application consists of:

- Three to eight dynamic application elements with identical data structure. Two of the application elements alternately contains the current contents. The rest of the application elements constitute a log of old application elements. The contents can be changed by validation transaction transaction.

The size of each dynamic application element will typically be 1 sector. Application element sizes of 2 or 3 are possible.

The so-called ticket/log area of the travel card containing an application of type 2 is managed by the *TicketLogAreaSectorPointer* and *TicketLogSectorPointer(x)* data elements of *TCAS: Applications Status*. This management is described in section 7.6

A travel card may have one application of type AT2.

Example of application of type AT2:

- TCST-ci/co

Figure 3 below illustrates the structure of an application type AT2.

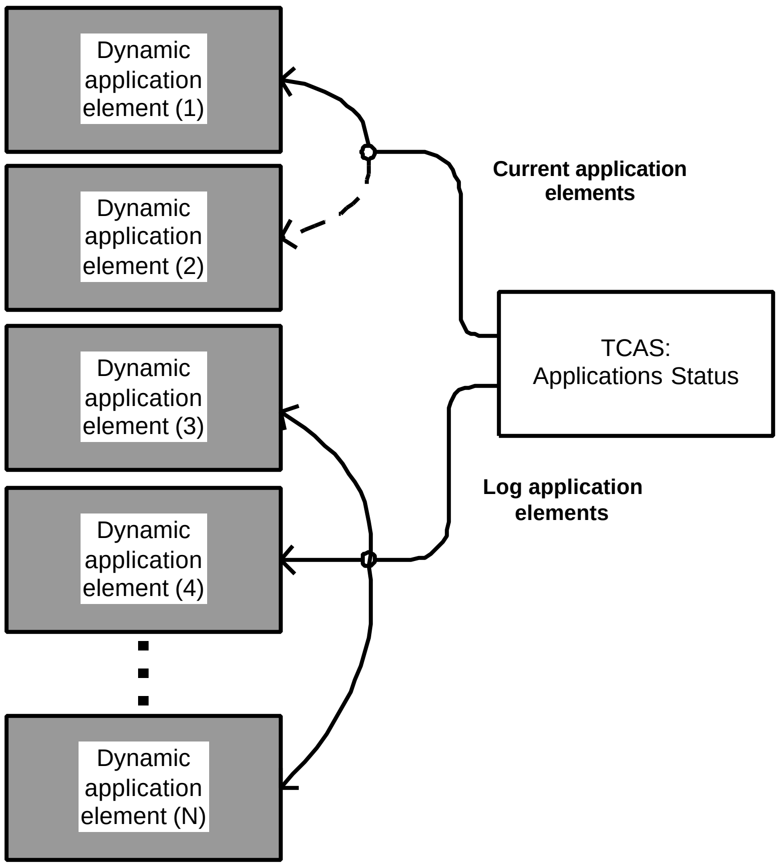

Dynamic application element (1)
Dynamic application element (2)
Dynamic application element (3)
Dynamic application element (4)
Dynamic application element (N)
Current application elements
TCAS: Applications Status
Log application elements

Figure 3: Application type 2

7.2.3 Application Type AT3

The application consists of:

- A number of sectors each having 3 one block log records. All log records have identical data structure. The log records are used cyclically. The contents can be changed by a validation transaction.

The most recently used log record is determined by *EventLogRecordNumber* of *TCAS: Applications Status*.

A travel card may have one application of type AT3.

Example of application of type AT3:

- TCEL

The structure of application type AT3 is illustrated in Figure 4 below.

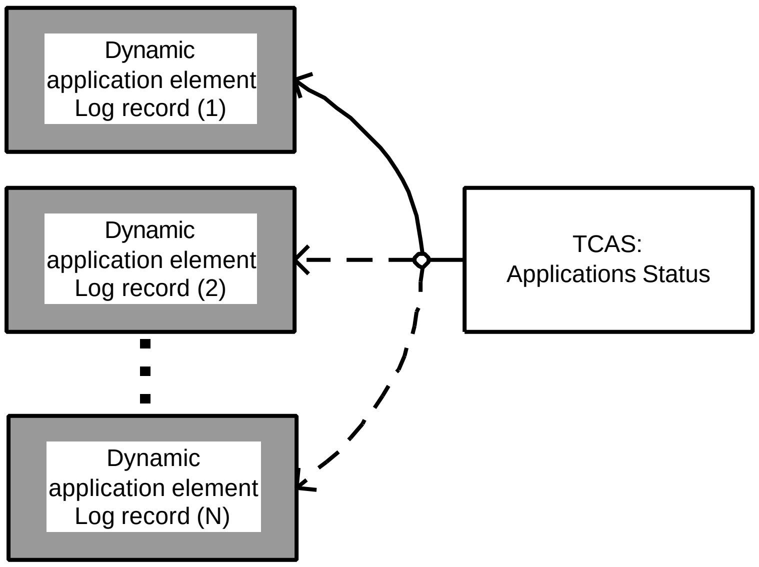


Figure 4: Application type 3

7.2.4 Application Type AT4

The application consists of:

- One static application element. The contents of this element is only defined or changed by an initiation transaction.

A travel card may have any number of applications of type AT4.

Example of application of type AT4:

- TCCP

The following Figure 5 illustrates an application of type AT4:

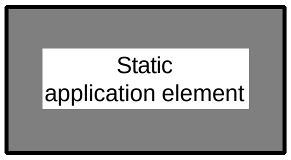


Figure 5: Application type 4

7.2.5 Application Type AT5

The application consists of:

- Two dynamic application elements with identical data structure. The two dynamic application elements alternately contains the current contents. The contents can be changed by a validation transaction.

The current dynamic element is determined by the relevant *SectorStatus* of *TCAS: Applications Status*, i.e. the *SectorStatus* representing for the first sector of the application.

A travel card may have any number of applications of type AT5.

Application type AT5 is a special case of AT1 with no static data element.

There are no present examples of applications of type AT5.

The following Figure 6 illustrates an application of type AT5:

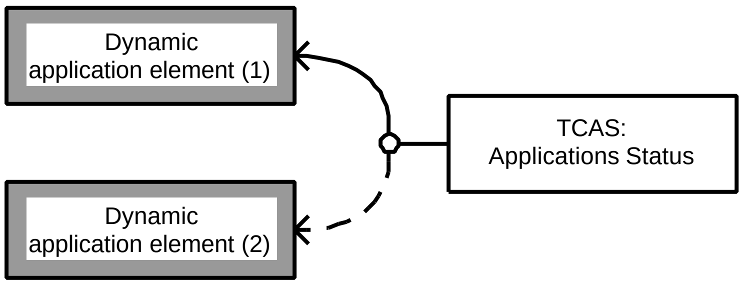

Dynamic application element (1)
TCAS: Applications Status
Dynamic application element (2)

Figure 6: Application type 5

7.3 General Rules for Travel Card Data

This section specifies general rules for positioning and contents of system objects, application objects, data element groups, and data elements.

7.3.1 Rules for Application Objects

- The number of electronic purse TCPU application objects shall not exceed 1.
- The number of event log TCEL application objects shall not exceed 1.
- The number of application objects of type AT2 shall not exceed 1.

7.3.2 Rules for Positioning of Application Objects

- An application object occupies a number of consecutive sectors of the travel card. I.e. an application object always starts at a sector boundary. (Consecutive sectors are defined as: S0, S1, S2, ..., S15.)
- If an application object does not utilise all blocks of its assigned sectors, the unused blocks must be the last blocks, i.e. either block 2 of the last sector, if 1 block is unused, or blocks 1 and 2 of the last sector, if 2 blocks are unused.
- The typical size of an application object is 1 to 4 sectors. The maximum size of an application object is 13 sectors.
- Each static and dynamic application element occupies a number of consecutive blocks of the travel card. I.e. a static or dynamic part of an application object starts at a block boundary. (Consecutive blocks are defined as: S0:B0, ..., S5:B2, S6:B0, S6:B1, S6:B2, S7:B0, ..., S15:B2.)

7.3.3 Rules for System Object and Application Object Contents

- The first data elements of data element groups are subject to the following rules:

  Rule 1:

  This rule applies to:

  - The first data element group of each system object and application object

  The first data elements of a data element group are:

  - Identifier value (8 b)
  - Version number (6 b)

  Rule 2:

  This rule applies to:

  - Data element groups (except *TCTI, Header Information* and *TCCO, Header Information*) that are part of TCTI or TCTO

  The first data element of a data element group is:

  - Identifier value (8 b)

  In these cases neither rule 1 nor rule 2 applies (i.e. the first data elements are not subject to restrictions):

  - Data element groups of special system objects (CMI, TCCI, TCDI)
  - Data element groups that are part of dynamic application elements of application objects of type AT1 (except TCTI and TCCO)

- The identifier values of data element groups must be unique across all PTAs. Section 8.1 holds the current list of allocated identifier values.
- The version number of the first data element group of a system object or an application object identifies a specific design of the object. Version numbers must be unique across all PTAs.
- All application objects must contain an AID data element designating the PTA that owns the application object. Normally, the AID data element immediately follows the first 2 data elements holding the identifier value and the version number.

  This rule does not apply to special application objects (TCEL).

  Several application objects will in addition contain one or more AID data elements for other purposes.

- All application objects must contain a status data element (8 b) designating the status of the object.
- All system objects and application objects holding some kind of valuable and/or critical contents must be protected by a MAC value. The last data elements of the last data element group of these objects must be:

- MAC algorithm identifier (2 b)
- MAC key identifier (6 b)
- MAC authenticator (16 b, 32 b, 48 b, or 64 b)

This rule does not apply to special system objects (CMI, TCCI, TCDI) and special application objects (TCEL).

MAC data elements are positioned in application objects according to the application type:

AT1, AT5:	MAC data elements in each of the two dynamic application elements.
AT2:	MAC data elements in all dynamic application elements.
AT3:	MAC data elements in all application elements constituting the log records (note, that the application elements of TCEL have no MAC data elements).
AT4:	MAC data element in the static application element.

- All blocks that do not contain a MAC authenticator data element must contain a CRC-8 checksum data element (8 b).

  The blocks of a few special system objects do not contain a checksum, even though they do not contain a MAC authenticator either (CMI, TCDI).

7.3.4 Rules for Positioning Data Elements within Blocks

- The basic rule is that data elements are positioned closely packed without gaps of unused bits.
- In data element groups containing a MAC authenticator, this value is always positioned as the last bits of the last block. This will in most cases result in a gap between the MAC authenticator data element and the previous (MAC key identifier) data element. All "padding" bits in this gap must be 0.
- In a block containing a checksum, the checksum is positioned as the last byte of the block.
- In application elements of size 2 or more blocks, data elements will cross blocks and checksum boundaries if necessary, leaving no gaps of unused bits at the end of blocks between a data element and the checksum, except possibly in the last block.
- If a block is the last block of a system object or an application element, and if the block does not contain a MAC authenticator, there will be a gap between the checksum data element in the last byte of the block and the last proper data element of the block. All "padding" bits in this gap must be 0.

7.4 Rules for Coding Data Elements

7.4.1 Coding Rules

- A block of 16 bytes (numbered 0, 1, 2, ..., 15) is interpreted as a sequence of 16 x 8 = 128 bits. The order of the bits in this sequence is specified by:
  1. The bits of byte 0 precede the bits of byte 1 and so on.
  2. Within one byte, the least significant bit (LSB) is first in the sequence and the most significant bit (MSB) is last.
- A data element within a block is represented by a consecutive section of bits from the total sequence of bits representing the whole block.
- Normally, data elements are interpreted as unsigned binary numbers. The order of the bits is specified by:
  - The LSB of the binary number is the first bit of the section of bits representing the data element
  - Consequentially, the MSB of the binary number is the last bit of the section
- If a data element is interpreted a signed binary number, the number is represented in twos complement (for example -1 is represented by all 1 bits), and the order of the bits is specified by:
  - The LSB of the binary number is the first bit of the section of bits representing the data element
  - The MSB of the binary number is the last but one bit of the section
  - The last bit of the section is the sign:

    0: Positive number or 0

    1: Negative number
- If a data element is interpreted as a BCD number, the length of the data element must be a multiple of 4 bits, and the order of the bits is specified by:
  - The least significant decimal digit of the number is the first 4 bits of the section of bits representing the data element
  - The most significant decimal digit of the number is the last 4 bits of the section
  - Within each subsection of 4 bits representing a decimal digit, the number is represented as an unsigned binary number in the range 0..9, and the order of the bits is specified by:
  - The LSB of the binary number is the first of the 4 bits
  - The MSB of the binary number is the last of the 4 bits
- If a data element is divided between 2 consecutive blocks B_1 and B_2, the sections of bits from the two blocks are concatenated before interpreting

the data element by one of the preceding rules. The sequence of bits representing the data element consists of the bit section of block B_1 succeeded by the bit section of block B_2. Note that the last byte block B_1 is a checksum that is positioned between the 2 bit sections (cf. sections 7.3.3 and 7.3.4).

7.4.2 Example of Coding of Data elements

In this example the first 4 data elements of a block are A, B, C, and D.

Data element A is a 2 bit data element:

MSB	LSB
A_2	A_1

Data element B is a 14 bit data element:

MSB													LSB
B_{14}	B_{13}	B_{12}	B_{11}	B_{10}	B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1

Data element C is a 6 bit data element:

MSB					LSB
C_6	C_5	C_4	C_3	C_2	C_1

Data elements D is a 12 bit data element:

MSB											LSB
D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1

The 4 data elements are coded into the first 5 bytes of the block as follows:

	MSB							LSB
Byte 0	B_6	B_5	B_4	B_3	B_2	B_1	A_2	A_1
Byte 1	B_{14}	B_{13}	B_{12}	B_{11}	B_{10}	B_9	B_8	B_7
Byte 2	D_2	D_1	C_6	C_5	C_4	C_3	C_2	C_1
Byte 3	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3
Byte 4							D_{12}	D_{11}

7.5 Interpretation of the Directory

The *TCDI: Directory* is positioned in sector 1 of a type 1 travel card. The directory describes the contents of sectors 1 through 15 by an AID value and a PIX value. The contents of sector 0 has fixed interpretation, therefore it needs not be described in the directory. The structure of the directory is further described in section 4.2.

When processing a transaction, the directory should be read prior to reading system objects and application objects in sector 3 through 15, as the directory describes which objects are present at which positions.

Note that, according to the rules of section 7.3.2, all application objects start at a sector boundary.

The directory is interpreted according to the following rules, where low rule numbers indicate high priority rules. The rules are based on the special AID and PIX values, that are common to all PTAs. The rules can be applied naturally, while the directory is scanned from sector 1 to sector 15. In the description of rules 'aid' designates any AID value, and 'pix' designates any PIX value.

Rule	AID	PIX	
1	00 H	pix	Sector free
2	01 H	pix	Sector defective
3	02 H	pix	Sector reserved
4	05 H	00 H	Sector contains in the first block the second instance of *Applications Status* (TCAS (2))
5	06 H	00 H	Sector contains *Directory* (TCDI)
6	0A H	00 H	Sector contains the first 3 log records of *Event Log* (TCEL)
7	0A H	01 H	Sector contains 3 log records of *Event Log* (TCEL)
8	0B H	pix	Sector contains *Purse* (TCPU)
9	0C H	pix	Sector contains a possible PTA-specific area bound to the *Purse* (TCPU)
10	aid (> 02 H)	01 H	The sector is the 2nd or subsequent sector of an area of sectors implementing an application object. The first sector of the area of the application object is the nearest preceding sector having AID value aid and a PIX value different from 01 H
11	aid (> 02 H)	pix	The sector is the first sector of an application object identified by AID value (identification of PTA) aid and PIX value pix (PTA specific type of application)

For a thorough description of AID and PIX values, see sections 8.2 and 8.3.

AID values are defined by Resekortsföreningen.

To obtain a desired level of interoperability, co-operating PTAs must agree on the interpretation of PIX values.

7.6 Ticket/Log Area

A number of instances (3 to 8) of tickets (or contracts) can be allocated on the travel card to form a so-called ticket/log area. The tickets or contracts form an application of type AT2.

7.6.1 Background

The idea behind a ticket/log area is to combine two dynamic application elements, which alternately hold the current ticket (or contract) contents, with 1 to 6 old dynamic application element, which constitute a log of tickets (or contracts).

If a ticket/log area is used for a specific kind of ticket (or contract), validation transactions involving application objects of the area will typically not generate log records in *TCEL: Event Log*. Instead the old tickets will hold necessary log information.

The advantage of using the ticket/log area, is that the number of blocks written during a validation transaction is reduced, as a log record will not be written in TCEL, thereby reducing the overall transaction time.

The disadvantage is that the structure of the travel card will be more complicated. Specifically, the time to read the complete log of TCEL and ticket/log area will be increased. This is, however, not a part of normal time critical validation transactions.

To ensure indivisible transactions, data elements in *TCAS: Applications Status* are used to manage the ticket/log area by pointing to the application elements. This management further ensures that the contents of application objects do not have to be moved on the travel card.

7.6.2 Principles of the Ticket/Log Area

The following principles apply to the use of a ticket/log area. Assume that the area has N application objects (tickets or contracts).

- The *TicketLogAreaSectorPointer* data elements of TCAS points to the first application object of the ticket/log area:
- The *TicketLogSectorPointer (x)* ($1 \leq x \leq 8$) data elements of TCAS point to the application objects of the ticket/log area:
  - *TicketLogSectorPointer (1)* and *TicketLogSectorPointer (2)* points to the two application objects, which alternately contains the current contents of the newest application object. *TicketLogSectorPointer (1)* points to the current application object. *TicketLogSectorPointer (2)* points to the application object, which will become the next application object.
  - *TicketLogSectorPointer (x)* ($3 \leq x \leq N$) points to the application objects, which constitute a log of old application objects. *TicketLogSectorPointer (x+1)* is older than *TicketLogSectorPointer (x)*.
- A transaction involving update of an application object of the ticket/log area can perform the update in two ways:
  1. *The current application object is updated:* The new contents is written onto the application object pointed to by *TicketLogSectorPointer (2)*. When finishing the transaction, the TCAS is updated by

swapping the values *TicketLogSectorPointer (1)* and *TicketLogSector-Pointer (2)*.

2. *A new application object is needed:* The contents of the new application object is written onto the application object pointed to by *TicketLogSectorPointer (2)*. When finishing the transaction, the TCAS is updated:
   - *TicketLogSectorPointer (1)* becomes the previous *TicketLog-SectorPointer (2)*
   - *TicketLogSectorPointer (2)* becomes the previous *TicketLog-SectorPointer (N)*
   - *TicketLogSectorPointer (3)* becomes the previous *TicketLog-SectorPointer (1)*
   - *TicketLogSectorPointer (x)* ($4 \leq x \leq N$) becomes the previous *TicketLogSectorPointer (x-1)*

- If the travel card has both a *TCEL: Event Log* and a ticket/log area, the log records of TCEL and the old application objects of the ticket/log area must together be regarded as the log of the travel card.
- All application objects of the ticket/log area must be identical, i.e. have identical structure and data elements.
- The size of the application objects of the ticket/log area must be at most 3 sectors.
- In the present specification, the ticket/log area can only be used for TCST-ci/co application objects.

7.6.3 Examples of the Ticket/Log Area

In this example 6 1-sector TCST-ci/co tickets are allocated on the travel card as a ticket/log area in sectors 6 through 11.

The following table shows the contents of the *TicketLogSectorPointer* ("TLSP") data elements of TCAS after initiation ("Init."), and after 6 transactions, which are either issuing of a new ticket ("New") or update of the current ticket ("Upd."):

	Transactions							
	0: Init.	1: New	2: Upd.	3: Upd.	4: Upd.	5: New	6: Upd.	
TLSP (1)	6	7	11	7	11	7	10	Current tickets
TLSP (2)	7	11	7	11	7	10	7	
TLSP (3)	8	6	6	6	6	11	11	Log tickets
TLSP (4)	9	8	8	8	8	6	6	
TLSP (5)	10	9	9	9	9	8	8	
TLSP (6)	11	10	10	10	10	9	9	

7.7 Calculation of Checksum and MAC Values

The system objects and application objects of the travel card contain checksum values and MAC values according to the rules of section 7.3. These values are important tools for ensuring the desired level of security by:

- Detecting errors in the IC-card chip or in the transmission of data to and from the card.
- Authentication of card applications, i.e. the possibility of detecting unauthorised parties modifying card applications.
- Making it possible to ensure that transactions are performed indivisibly.

When system objects and application objects are modified, checksum values and MAC values must be calculated and written on the card.

When system objects and application objects are read, checksum values and MAC values must be recalculated and compared with the values read, to find out, if the objects are consistent.

7.7.1 Calculation of Checksum Values

All blocks of system objects and application objects contain 8 b CRC-8 checksum values, except for blocks containing MAC values (cf. section 7.3.3). The checksums are calculated when modifying and reading the blocks.

Checksums are calculated according to the following rules:

- The checksum is standard CRC-8 (Cyclic Redundancy Code).
- All bytes of a block contribute to the checksum calculation with the exception of the checksum byte itself. Even possible padding zeroes between the last proper data element and the checksum byte contribute to the checksum.

7.7.2 Calculation of MAC Values

All system objects (except CMI, TCDI), and all application objects (except TCEL), contain MAC values (cf. section 7.3.3). The MAC values are calculated when modifying and reading the objects.

The MAC value is calculated based on the following data:

- The data to be authenticated, i.e. input to the MAC algorithm (described in subsection below)
- The *MACAlgorithmIdentifier* of the system object or application object, identifying the MAC algorithm
- The *MACKeyIdentifier* of the system object or application object, identifying the master key to the MAC algorithm
- The size of the *MACAuthenticator* data element holding the MAC value. Allowed sizes are 16 b, 32 b, 48 b, and 64 b.

At present, only one *MACAlgorithmIdentifier* value is defined, denoting the desMAC algorithm as described in section 6.2.1.

PTAs accepting each others travel cards by an interoperability agreement must co-ordinate the way in which the *MACKeyIdentifier* denotes one of up to 64 different master keys to the MAC algorithm.

Figure 7 outlines how the MAC key is calculated using the desMAC algorithm, and how the output is truncated to the specified size as the most significant bits of the 64 b output of the algorithm.

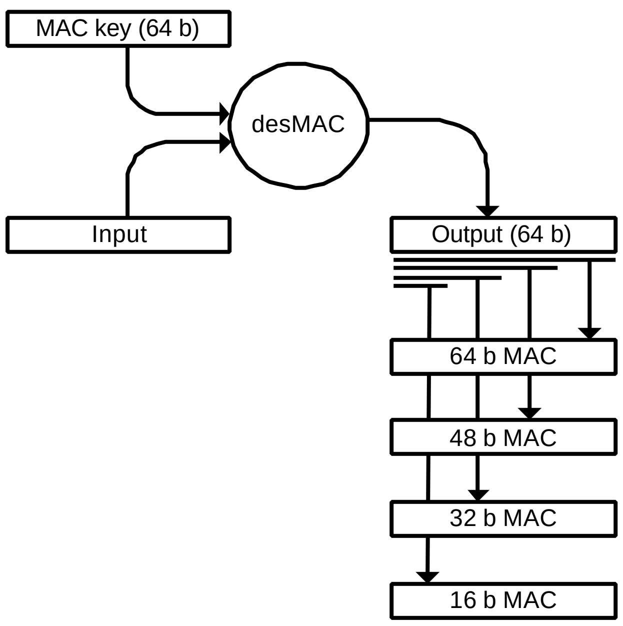


Figure 7 MAC calculation overview

The data to be authenticated, i.e. input to the MAC algorithm, always include the *CardSerialNumber* of *CMI: Manufacturer Information*. As the *CardSerialNumber* is unchangeable, this will prevent system objects or application objects to be copied from one travel card to another, without being detected.

The input to the MAC algorithm is a string of bits. The MAC algorithm specifies, if the input string of bits has to be padded with e.g. zeroes to reach an appropriate length.

The following table defines which data that has to be concatenated as a bit string to become the input to the MAC algorithm (application types are defined in section 7.2).

System object or application object	Input to MAC algorithm
System object TCCI	*CardSerialNumber* of CMI
	TCCI system object ①
System object TCAS	*CardSerialNumber* of CMI
	TCAS system object ①
Application object type AT1	*CardSerialNumber* of CMI
	Static application element ②
	Relevant dynamic application element ③
Application object type AT2	*CardSerialNumber* of CMI
	Relevant dynamic application element ③
Application object type AT3	*CardSerialNumber* of CMI
	Relevant log record application element ③
Application object type AT4	*CardSerialNumber* of CMI
	Static application element ③
Application object type AT5	*CardSerialNumber* of CMI
	Relevant dynamic application element ③

Please note:

① The whole block of the system object contributes to the bit string with the exception of the *MACAuthenticator* data element. Even possible padding zeroes between the last proper data element and *MACAuthenticator* contribute to the bit string.

② All blocks of the application element contributes to the bit string, even the checksum byte, and possible padding zeroes between the last proper data element and the checksum byte.

③ All blocks of the application element contributes to the bit string with the exception of the *MACAuthenticator* data element of the last block. Even possible padding zeroes between the last proper data element and *MACAuthenticator* contribute to the bit string.

- The above definitions of input to the MAC algorithm for each type of application only applies, if the application object does in fact contain MAC values. For example TCEL of type AT3 does not contain MAC values.

7.8 Travel Card Consistency

When performing travel card transactions, it is important to ensure, that the travel card is consistent, whether the transaction is completed successfully or not.

When reading a travel card, the consistency should be checked to a reasonable degree. Checksum values and MAC values must be checked for all system objects and application objects taking part in a transaction.

A travel card is consistent, if the following conditions are satisfied:

- All system objects are consistent
- All application objects are consistent

A system object or an application object is consistent, if the following conditions are satisfied:

- Checksum and MAC values must be correct
- Data elements have legal values
- Pointers to other system objects or application objects must be correct

For application object the above conditions only apply to the application elements, that hold the current data of the application object:

7.9 Special Ticket (TCST) Concept

The TCST concept is framework for defining specialised tickets and contracts targeted at the needs of one PTA or a group of co-operating PTAs.

7.9.1 Background

Tickets and contracts can be created on the basis of the TCTI and TCCO application objects. These application objects have a large degree of flexibility because they can be constructed by a selection from a range of possible data element groups:

- Header Information
- Administrative Price Information
- Administrative Class Information
- Validation Information
- Customer Information
- Issuing Contract/Purse
- Period of Validity
- Validity, Distance
- Validity, Route
- Validity, Zone
- MAC Algorithm Identifiers
- Dynamic Information
- Issuing Information, MAC

The purpose of the generally adaptable TCTI and TCTO application objects is to enable easy creation of usable tickets and contracts, based upon the application elements needed by any given PTA.

However, this approach has some disadvantages:

- The flexible structure causes an overhead in space consumption due to the *Identifier* data element, which is placed in each data element group.
- The use of the predefined data element groups to construct application elements, will often result in unnecessary data elements and/or the lack of needed data elements. This causes a mismatch between what is wanted and what can be implemented.

As a consequence, it might not be possible to create a TCTI or TCCO with required contents, while keeping it within an acceptable size.

This section describes the possibility of defining specialised tickets and contracts adapted to different needs. To ensure the desired level of interoperability, the specialised tickets and contracts have to fulfil the same basic requirements as other application objects.

7.9.2 Special Ticket and Contract Concept

In addition to the TCTI and TCCO application objects, that are created by selecting from a range of data element groups, a new family of application objects can be defined.

The general name is TCST (Travel Card Special Ticket). For a specific TCST, the name must be suffixed by a name indicating the type of special ticket. An example is “TCST-ci/co” where the “ci/co” part of the name indicates, that the tcket is designed for check-in/check-out validation.

Every TCST-x must be assigned a unique identifier. This is done by Resekortsföreningen. It is important as well, that each TCST-x is described in detail on a level similar to the descriptions of TCTI and TCCO. Over time, a TCST-x can exist in several versions identified by version numbers.

It is crucial that the use of these specialised tickets and contracts is guided adequately; otherwise it can have unfavourable influence on the interoperability of the RKF travel card. The process of introducing a new specialised ticket is not unlike the introducing of new versions of existing applications (cf. [RKF-0024] section ‘Backwards Compatibility’).

When introducing a new specialized ticket, It is the responsibility of Resekortsföreningen to:

- Assign ’Identifier’ values and version numbers.
- Ensure that current principles for application objects and application elements are satisfied (cf. chapter 7).
- Ensure that the new application object is fully documented.

7.10 Management of Currency Unit and Money Values

The *CardCurrencyUnit* data element of *TCCI: Card Information* specifies the currency and unit used for all money values of the card, i.e. money amount data elements in TCPU, TCTI, TCCO, TCEL etc.

The interpretation of the *CardCurrencyUnit* element complies with [ISO 1545-2] as described in [RKF-0023] (section ‘Data Types’).

It is possible to handle “money” values that are not connected to a specific currency, e.g. some kind of ‘travel token’.

When travel cards are used in an area where more than one currency is used, the front systems must contain conversion factors to convert to and from the currency of *CardCurrencyUnit*. If this case, the unit of *CardCurrencyUnit* should be a small unit.

7.11 Management of Places

Places designate physical points or areas in tickets and contracts, e.g. as origin and destination of a journey. Places can be stops, points, zones etc.

Each place is identifies by two data elements:

AID:	Describes the PTA “owning” the place according to the list of AID values in [RKF-0017].
Place:	A data element designating a place within the relevant PTA. The interpretation is PTA specific, i.e. specific to the corresponding AID value. Places are 14 b data data element, i.e. each PTA can have up till 16383 different places (0 is the undefined place).

To obtain a desired level of interoperability, co-operating PTAs must agree on the interpretation of place values.

8 CONSTANTS

8.1 Identifier Values

These identifier values identify travel card application objects and data element groups as described in section 7.3.

Value	Description of the identifier
00 H	Undefined identifier value
01 H .. 7F H °	RFU
80 H .. 83 H	NUV-B
84 H	Event log record
85 H	TCPU
86 H	TCTI, Header Information
87 H	TCCO, Header Information
88 H	Dynamic Information With More Than One Data Block
89 H	TCTI/TCCO, Mandatory Data
8A H	Dynamic Information With Only One Data Block
8B H .. 92 H	RFU
93 H	MAC Algorithm Identifiers
94 H	Administrative Price Information
95 H	Issuing Contract/Purse
96 H	TCTI, Period of Validity
97 H	Validity, Distance
98 H	Validity, Route
99 H	Validity, Zone
9A H	TCTI, Issuing Information
9B H	Customer Information
9C H	Administrative Class Information
9D H °	TCCO, Period of Validity
9E H °	TCCO, Issuing Information
9F H	Validation Information
A0 H °	TCSA
A1 H °	TCDB
A2 H °	TCCP
A3 H °	TCST-ci/co
A4 H °	TCRE

Value	Description of the identifier
A5 H .. FF H °	RFU

Identifier values marked with ° are introduced in volume B of this specification.

Resekortsföreningen is responsible for defining new identifier values.

8.2 AID Values

These AID values are special values used in the *Directory* (TCDI) to specify the system objects, application objects, and data element groups common to all PTAs. The AID values defining PTAs etc. are specified in [RKF-0019].

Value	Description of the application identifier
00 H	Sector free
01 H	Sector defective
02 H	Sector reserved
03 H	NUV-B
04 H	NUV-B
05 H	Sector contains in the first block the second instance of *Applications Status* (TCAS (2))
06 H	Sector contains *Directory* (TCDI)
07 H .. 09 H	RFU
0A H	Sector contains 3 log records of *Event Log* (TCEL)
0B H	Sector contains *Purse* (TCPU)
0C H	Sector contains a possible PTA-specific area bound to the *Purse* (TCPU)
0D H .. 1F H	RFU
20 H .. 2F H	Not used in production. Can be used for test and validation of systems.
30 H .. 63 H	RFU

AID values marked with ° are introduced in volume B of this specification.

Resekortsföreningen is responsible for defining new AID values.

8.3 PIX Values

These PIX values are special values used in the *Directory* (TCDI) according to the rules specified in section 7.5. Remaining PIX values are PTA specific. Note however, that [RKF-0023] (section ‘Data Types’) specifies a proposed usage of these PIX values.

Value	Description of the PIX value
00 H °	Undefined PIX value
01 H °	The sector is the 2nd or subsequent sector of an area of sectors implementing an application object. The first sector of the area of the application object is the nearest preceding sector entry of the directory having a PIX value different from 01 H
02 H .. 1F H °	RFU

PIX values marked with ° are introduced in volume B of this specification..

8.4 Event Codes

These 6 bit codes can be used as *EventCode* in *Event Log Record* (cf. section 5.2) designating the type of transaction being logged. The column ‘Event data’ describes the type of *EventData* used in connection with each event code.

Value	Description of the event code	Event data
00 H	Undefined event code	
01 H	Purchase of TCTI using TCPU	A
02 H	Purchase of TCCO using TCPU	A
03 H	Purchase of TCTI with another payment than the TCPU	A
04 H	Purchase of TCCO with another payment than the TCPU	A
05 H	TCTI issued by a TCCO	B
06 H	Check-in validation of TCTI / TCCO / TCST-ci/co	C
07 H	Extension of a TCTI / TCCO	C
08 H	Charge of the TCPU	D
09 H	TCTI removed	C
0A H	TCCO removed	C
0B H	TCPU removed	C
0C H	TCST-ci/co removed	C
0D H	TCCP removed	C
0E H	TCDB removed	C
0F H	TCRE removed	C
10 H .. 15 H	NUV-B	-

Value	Description of the event code	Event data
16 H	Card initialised	A
17 H	Application object created	C
18 H	Application object repurchased/reimbursed	A
19 H	TCCO activated	C
1A H	Purchase of paper ticket using TCPU	A
1B H	TCST-ci/co check-in	C
1C H	Check-out validation of TCST-ci/co	C
1D H	Control validation of TCTI / TCCO / TCST-ci/co	C
1E H	Excess fare validation of TCTI / TCCO / TCST-ci/co	C
1F H .. 2F H	RFU	-
30 H .. 3F H	PTA specific use	-