



**Wallet-POS Proposal**  
**Version 1.0**  
**03 May 2013**

*This is a Non-binding Permanent Reference Document of the GSMA*

---

**Security Classification: Non-confidential**

Access to and distribution of this document is restricted to the persons permitted by the security classification. This document is confidential to the Association and is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those permitted under the security classification without the prior written approval of the Association.

**Copyright Notice**

Copyright © 2013 GSM Association

**Disclaimer**

The GSM Association ("Association") makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

**Antitrust Notice**

The information contained herein is in full compliance with the GSM Association's antitrust compliance policy.

## Table of Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>   | <b>4</b>  |
| 1.1      | System Overview   | 5         |
| <b>2</b> | <b>List of Abbreviations</b>                                  | <b>5</b>  |
| <b>3</b> | <b>Referenced Documents</b>                                   | <b>6</b>  |
| <b>4</b> | <b>User Story and Workflow</b>                                | <b>6</b>  |
| 4.1      | User Story  | 7         |
| 4.2      | High-Level Workflows  | 8         |
| 4.2.1    | Single Transaction – Couponing                                | 8         |
| 4.2.2    | Single Transaction – Loyalty                                  | 9         |
| 4.2.3    | Single Transaction – Payment                                  | 10        |
| 4.2.4    | Multiple Transactions – Couponing/Loyalty/Payment             | 11        |
| <b>5</b> | <b>Terminal Use Cases</b>                                     | <b>12</b> |
| 5.1      | Use case diagram  | 12        |
| 5.2      | Use cases   | 13        |
| 5.2.1    | getVasData  | 13        |
| 5.2.2    | getCouponData / getVoucherData /getLoyaltyData                | 14        |
| 5.2.3    | putVasData  | 14        |
| 5.2.4    | redeemCoupon / redeemVoucher                                  | 14        |
| 5.2.5    | issueCoupon / issueBonus / issueVoucher / issuePaymentSchemes | 15        |
| 5.2.6    | authenticate  | 15        |
| 5.2.7    | Payment   | 15        |
| <b>6</b> | <b>Wallet/Terminal Interface Specification</b>                | <b>16</b> |
| 6.1      | Data Model  | 16        |
| 6.1.1    | Simple Type   | 16        |
| 6.1.2    | Basic Type  | 16        |
| 6.1.3    | Constructed Types   | 19        |
| 6.1.4    | Tag Overview  | 21        |
| 6.2      | VASAppCardlet APDU Commands                                   | 22        |
| 6.2.1    | Selection of VAS Application                                  | 22        |
| 6.2.2    | GetData   | 24        |
| 6.2.3    | PutData   | 24        |
| 6.2.4    | VASApp Cardlet Response Codes                                 | 26        |
| 6.3      | APDU Call Flows   | 27        |
| 6.3.1    | Call Flow – Coupon Redemption                                 | 27        |
| 6.3.2    | Call Flow – Loyalty Handling                                  | 28        |
| 6.3.3    | Call Flow – Couponing/ Loyalty Handling                       | 29        |
| 6.3.4    | Call Flow – Couponing/Loyalty/Payment Handling                | 30        |
| <b>7</b> | <b>Guidelines for Terminal Developers</b>                     | <b>31</b> |
|          | <b>Document Management</b>                                    | <b>33</b> |
|          | Document History  | 33        |
|          | <b>Other Information</b>                                      | <b>33</b> |

# 1 Introduction


This document sets out a technical proposal for Point of Sale (POS) Terminal Manufacturers and covers interoperable formats for the provisioning and management of mobile NFC Value Added Services.

This document is **NOT** a binding technical standard. It is only a **proposal** outlining a technical framework for POS Terminal Manufacturers which could be used as a platform to design new interoperable mobile NFC-based products and services.

The GSMA recognises the need for an interoperable Value Added Services approach and has therefore constructed this document as an initial technical proposal.

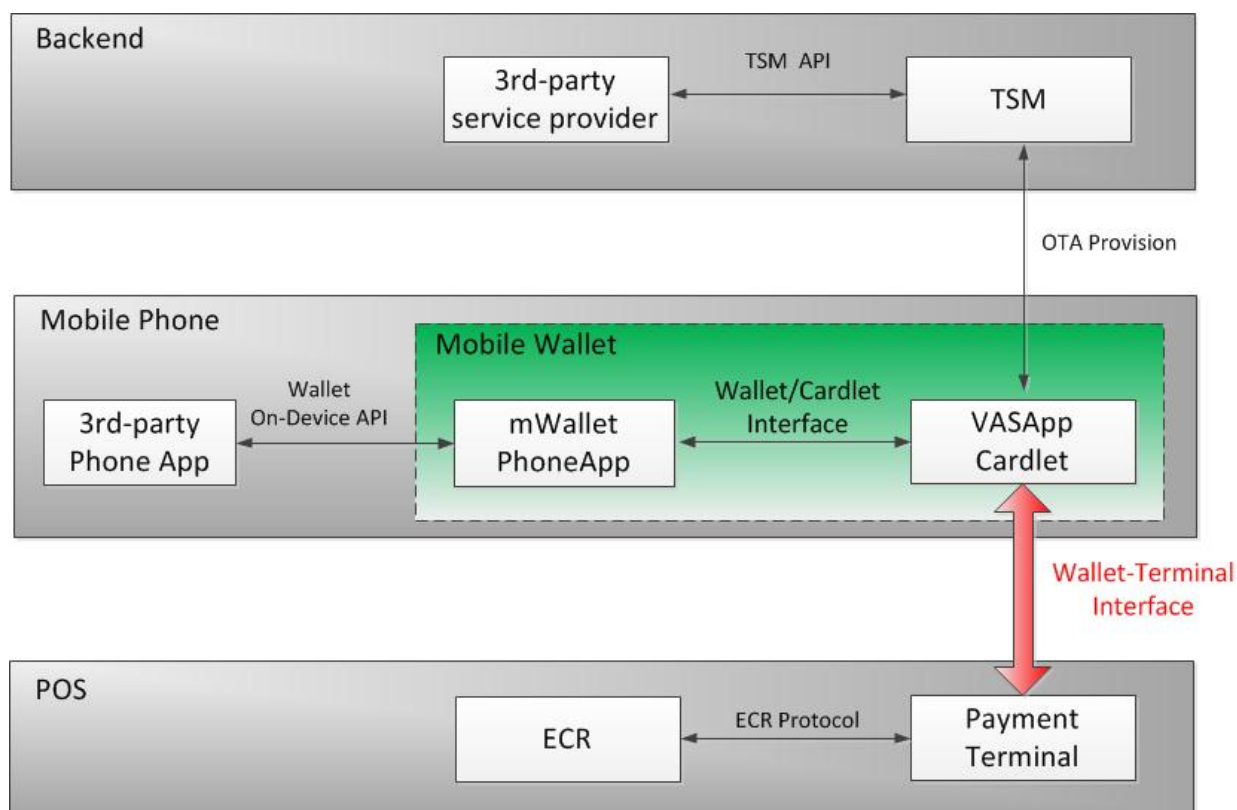
It is the intention of the document authors to deliver a more comprehensive, global framework based on the ideas outlined in this document and feedback from both the mobile and payment industry in Q3 2013.

A Mobile Wallet is an open software framework which is about bringing secure NFC contactless credit cards, ticketing, access, and loyalty/coupons onto the mobile handset in an integrative way. This paradigm opens up the new opportunity to the 3rd-party developers, who want to introduce new attractive proximity services to the wallet customers. To give these developers – especially the Small and Medium-sized businesses (SMB) - a low barrier to entry, we will offer them a set of Wallet APIs.

| Wallet API          | Point Integration | of Targeted Group   | Focus of this document  |
|---------------------|-------------------|---|---|
| On-Device API       | Device            | The VAS developers who have mobile apps products but no contactless solutions |   |
| <b>Terminal API</b> | UICC              | POS System Developers   |  |
| <b>TSM API</b>      | Backend           | The developers who have vertical contactless solutions.                       |   |

**Table 1.1: Wallet API Category**

## 1.1 System Overview



**Figure 1: System Overview**

In this document, we propose the Wallet/Terminal interface. Our target group is POS system developers.

## 2 List of Abbreviations

| Term    | Description  |
|---------|--|
| AID     | Application Identifier                                 |
| APDU    | Application Protocol Data Unit                         |
| Cardlet | Java Card applet running on a smart card, such as UICC |
| ECR     | Electronic Cash Register                               |
| mWallet | Mobile Wallet  |
| MNO     | Mobile Network Operator                                |
| NFC     | Near Field Communication                               |
| OTA     | Over the Air   |
| POS     | Point of Sale System                                   |
| RID     | Registered application provider Identifier             |

|      |                                   |
|------|-----------------------------------|
| SMB  | Small and Medium-sized businesses |
| TLV  | Tag Length Value                  |
| TSM  | Trusted Service Manager           |
| UICC | Universal Integrated Circuit Card |
| VAS  | Value Added Service               |

### 3 Referenced Documents

| Term | Description   |
|------|---|
| [1]  | ISO/IEC 7816-4<br>Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 4: Interindustry commands for interchange   |
| [2]  | ISO/IEC 7816-5<br>Identification cards -- Integrated circuit cards -- Part 5: Registration of application providers   |
| [3]  | ISO/IEC 8825-1:2008<br>Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)   |
| [4]  | GMSA White Paper: Mobile NFC in Retail (Oct. 2012)<br><a href="http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/Mobile-NFC-in-Retail-White-Paper-Oct-2012.pdf">http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/Mobile-NFC-in-Retail-White-Paper-Oct-2012.pdf</a>           |
| [5]  | GS1 – Global Standards One<br><a href="http://www.gs1.org/">http://www.gs1.org/</a>   |
| [6]  | GlobalPlatform Card Specification v2.2.1<br><a href="http://www.globalplatform.org/specificationscard.asp">http://www.globalplatform.org/specificationscard.asp</a>   |
| [7]  | Contactless Services – GlobalPlatform Card Specification v 2.2 - Amendment C v1.0.1<br><a href="http://www.globalplatform.org/specificationscard.asp">http://www.globalplatform.org/specificationscard.asp</a>  |
| [8]  | GSMA White Paper: The Mobile Wallet (V1.0, September 2012)<br><a href="http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/GSMA-Mobile-Wallet-White-Paper-Version-1-0.pdf">http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/GSMA-Mobile-Wallet-White-Paper-Version-1-0.pdf</a> |
| [9]  | GSMA NFC Handset APIs & Requirements (V 3.0, October 2012)<br><a href="http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/GSMA-NFC-Handset-APIs-Requirements-V3.01.pdf">http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/GSMA-NFC-Handset-APIs-Requirements-V3.01.pdf</a>     |
| [10] | GSMA NFC UICC Requirements (V 3.0, October 2012)<br><a href="http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/GSMA-NFC-UICC-Requirements-Specification-V3.01.pdf">http://www.gsma.com/mobilenfc/wp-content/uploads/2012/10/GSMA-NFC-UICC-Requirements-Specification-V3.01.pdf</a>   |

### 4 User Story and Workflow

Definition and scope of Loyalty/Couponing:

**Loyalty** Loyalty refers to a point collection centric bonus program that is targeted at one or more merchants. Here, the user gets a certain amount of points which are calculated based on a logic which the loyalty scheme provider defines. Therefore, every time the user makes a purchase, he forwards his loyalty customer identification number to the POS system via NFC which calculates the bonus points and adds it to the user's balance. This loyalty type is POS IT backend centric, i.e. the wallet is only responsible for storing and forwarding

the user's loyalty id. The loyalty bonus points processing and management is done via the POS system and loyalty provider's IT.

**Coupons** enable particular user groups to get discounts or other benefits when making a purchase. Coupons can e.g. give a discount (relative or absolute) on the whole basket or on dedicated products. Certain coupons can also have preconditions which need to be fulfilled by the user in order to redeem the particular coupon (e.g. prepaid, get two and pay for one). The validation of coupons is done by the POS system. The coupon issuer's motivation is to steer the user's behavior by incentivizing certain products or purchase pattern. In this document, different kinds of couponing services will be covered under "couponing", such as discount coupons, vouchers, prepaid gift cards, etc.

## 4.1 User Story

| No. | User story   |
|-----|--|
| 1   | As a Mobile Wallet user I want to redeem my coupons when tapping my phone against the contactless reader, so I can profit from the benefits provided by the coupons.   |
| 2   | As a Mobile Wallet user I want to participate in the retailer's loyalty program when tapping my phone against the contactless reader, so I can collect bonus points.   |
| 3   | As a Mobile Wallet user I want to make a payment when tapping my phone against the contactless reader, so I don't need any other means of payment.   |
| 4   | As a Mobile Wallet user I want to redeem coupons and make a payment when tapping my phone against the contactless reader, so I can profit from the benefits provided by the coupons and don't need any other means of payment.   |
| 5   | As a Mobile Wallet user I want to redeem coupons and participate in the retailer's loyalty program when tapping my phone against the contactless reader, so I can profit from the benefits provided by the coupons and collect bonus points.                           |
| 6   | As a Mobile Wallet user I want to participate in the retailer's loyalty program and make a payment when tapping my phone against the contactless reader, so I can collect bonus points and don't need other means of payment.  |
| 7   | As a Mobile Wallet user I want to participate in the retailer's loyalty program, redeem coupons and make a payment when tapping my phone against the contactless reader, so I can collect bonus points, profit from the coupons and don't need other means of payment. |

**Table 4.1: User Story**

## 4.2 High-Level Workflows

In this chapter, we propose the high-level communication protocols between mWallet and POS/Terminal for the contactless loyalty/couponing/payment scenarios. In the communication, ECR software works in master mode towards the payment terminal and can initialize different transaction modes, such as couponing mode, loyalty mode, and payment mode.

The sections 2.2.1 – 2.2.3 propose separate transaction flows for loyalty, couponing and payment, respectively, section 2.2.4 depicts one of the options detailing how a composited transaction would look like.

The sequence of a composited transaction flows and the number of taps required for a transaction is dependent on the retail requirements and POS implementation.

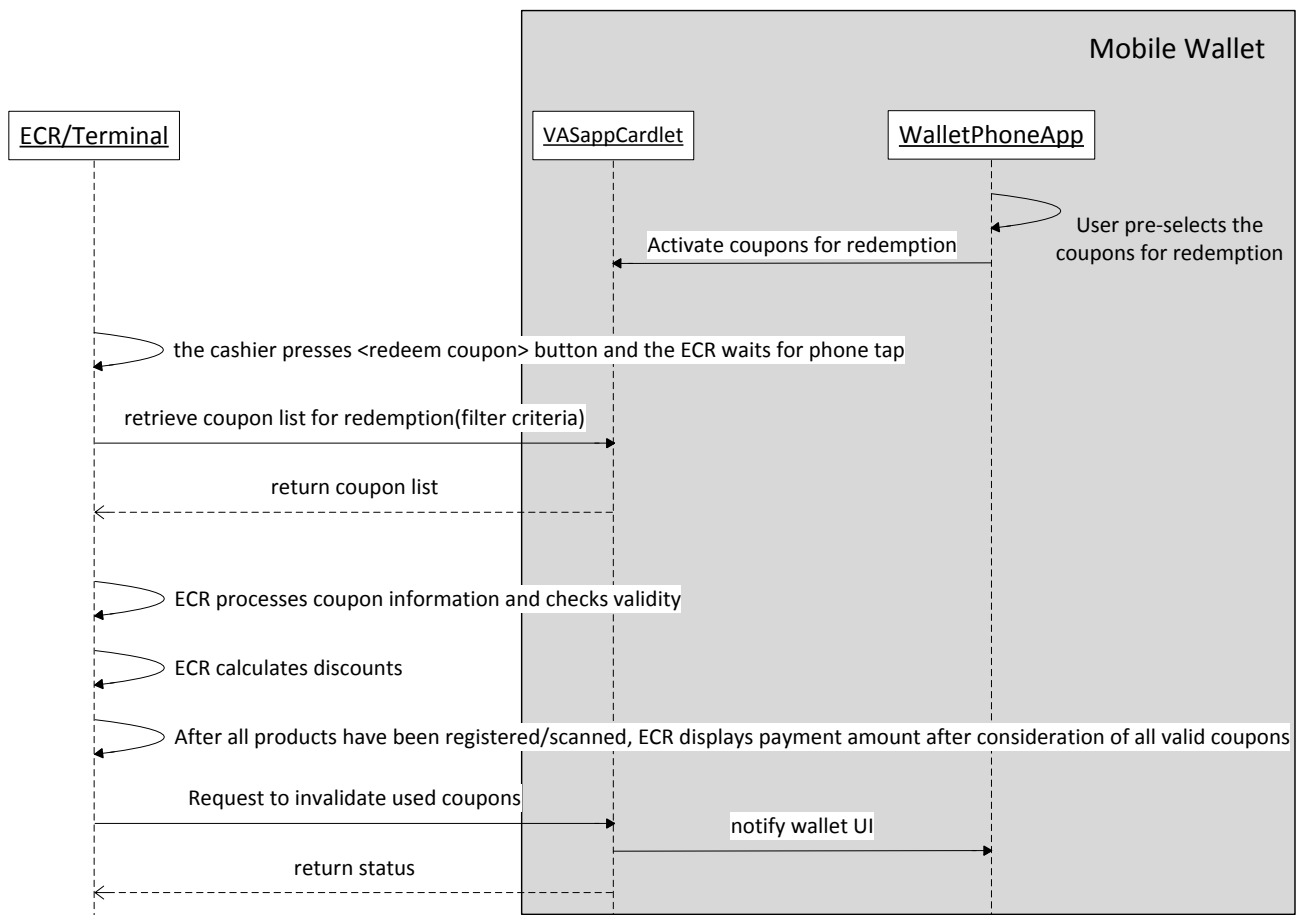
Remarks:

- 1) To ensure signalling efficiency only the applicable couponing/loyalty/payment data for the given retail shop will be transferred from Wallet to ECR system. The parameter “filter criteria” is used in the transaction protocol is for this purpose. This document only supports the use of Retailer ID and Coupon/Loyalty Issuer ID. Transfers of data such as geo-location of retail stores could be included in the next version of the specification..
- 2) New functionality, such as automated couponing and loyalty program could be introduced in the next version of the POS specification.

This document focuses on the user pre-selection of loyalty/couponing data.

### 4.2.1 Single Transaction – Couponing

Scenario description: The user taps the mobile wallet against the contactless terminal, in order to get the selected coupons redeemed.

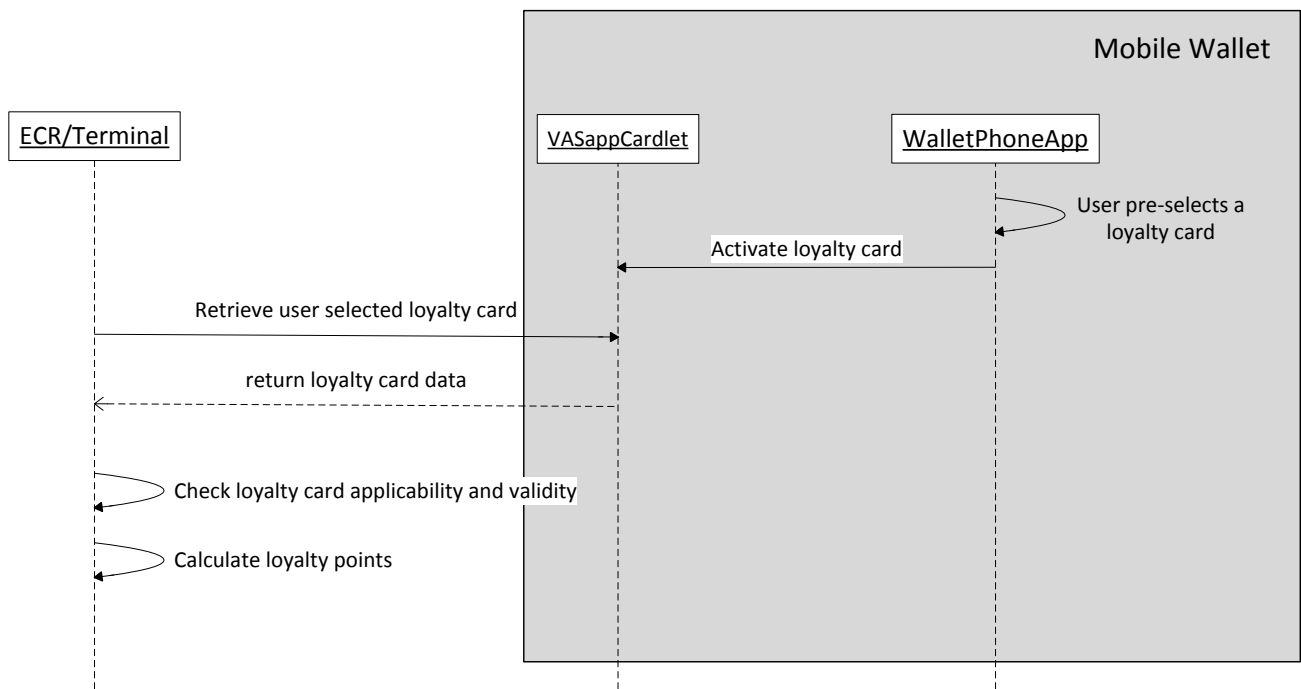


**Figure 2: Single Transaction – Couponing Handling**

#### 4.2.2 Single Transaction – Loyalty

Scenario description: The user taps the mobile wallet against the contactless terminal, in order to collect loyalty points.

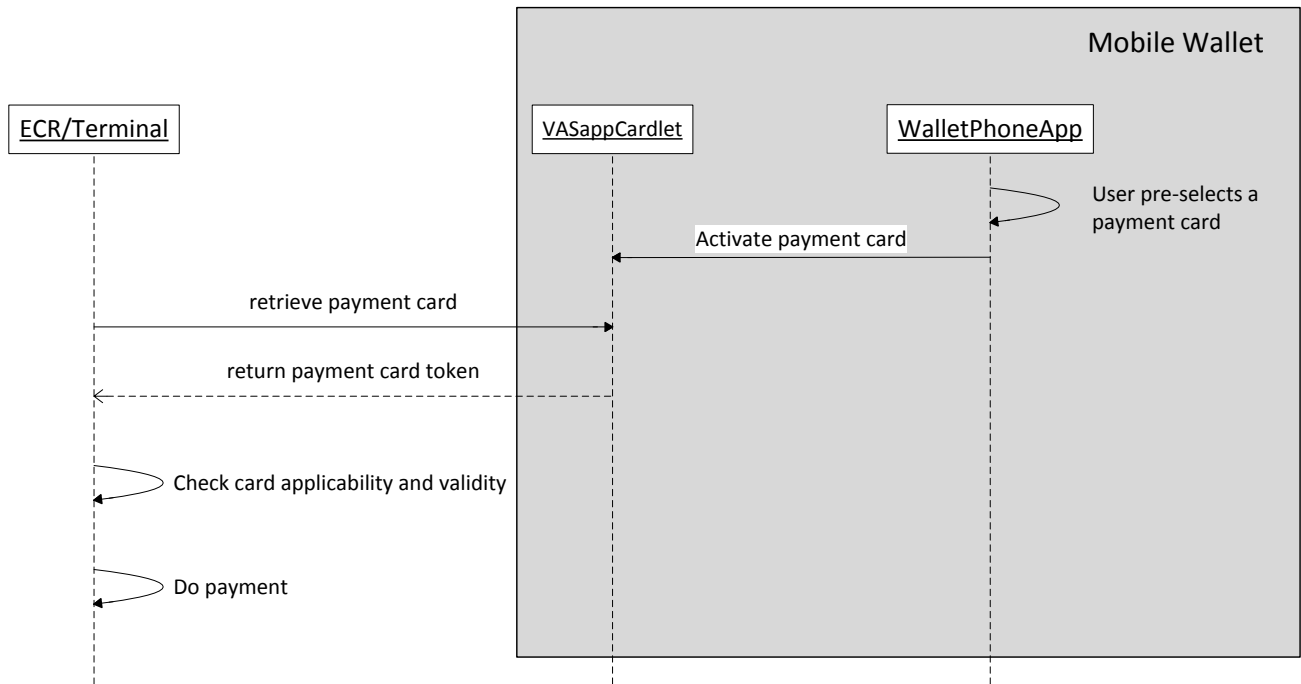




**Figure 3: Single Transaction – Loyalty Handling**

#### 4.2.3 Single Transaction – Payment

Scenario description: The user taps the mobile wallet against the contactless terminal, in order to make a payment.

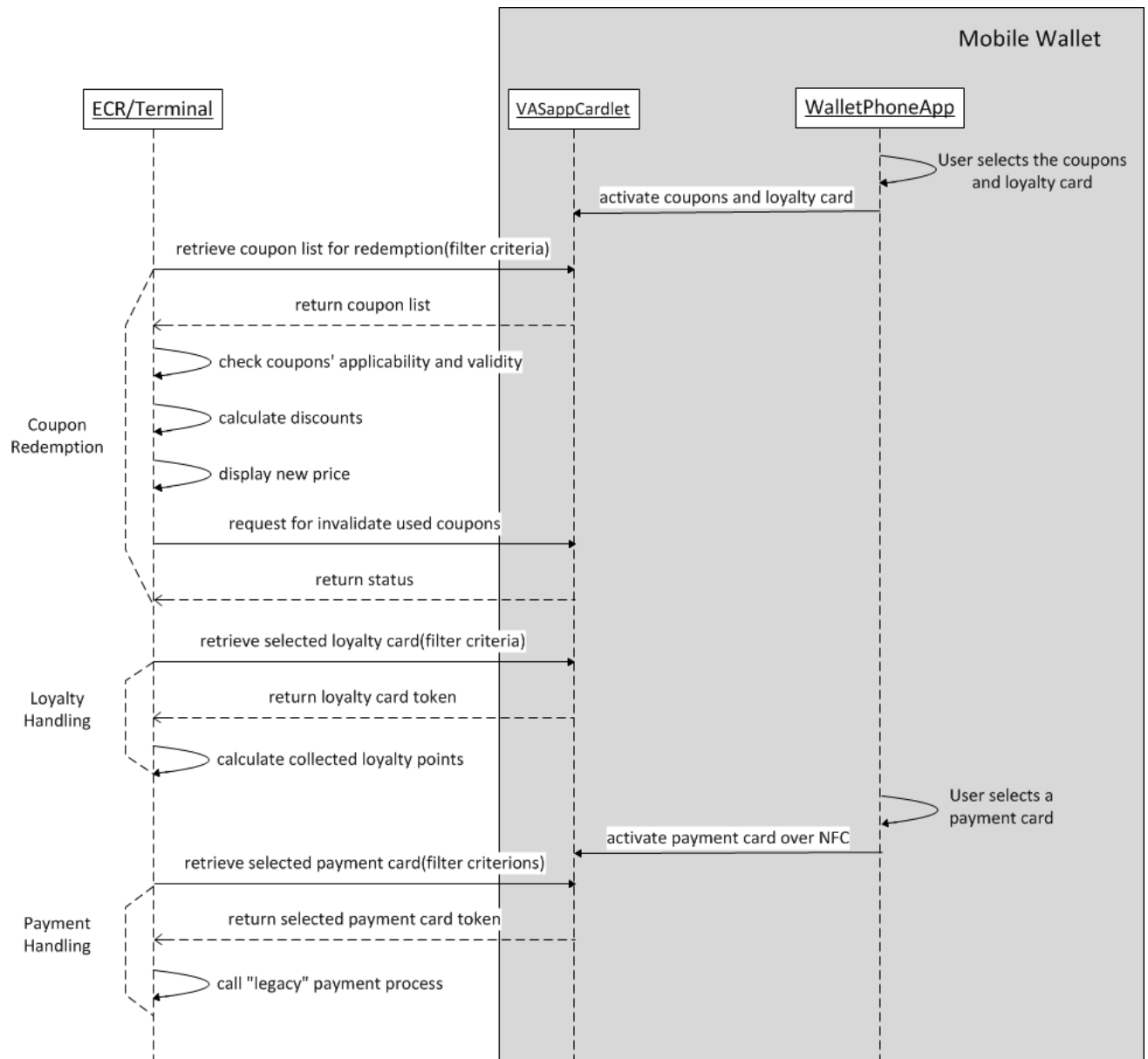


**Figure 4: Single Transaction – Payment Handling**

#### 4.2.4 Multiple Transactions – Couponing/Loyalty/Payment

Scenario description: The user taps the mobile wallet against the contactless terminal, in order to redeem coupons, make payment, as well as collect loyalty points. How to organize the transaction sequence and realize the above scenario is dependent on the POS implementation.

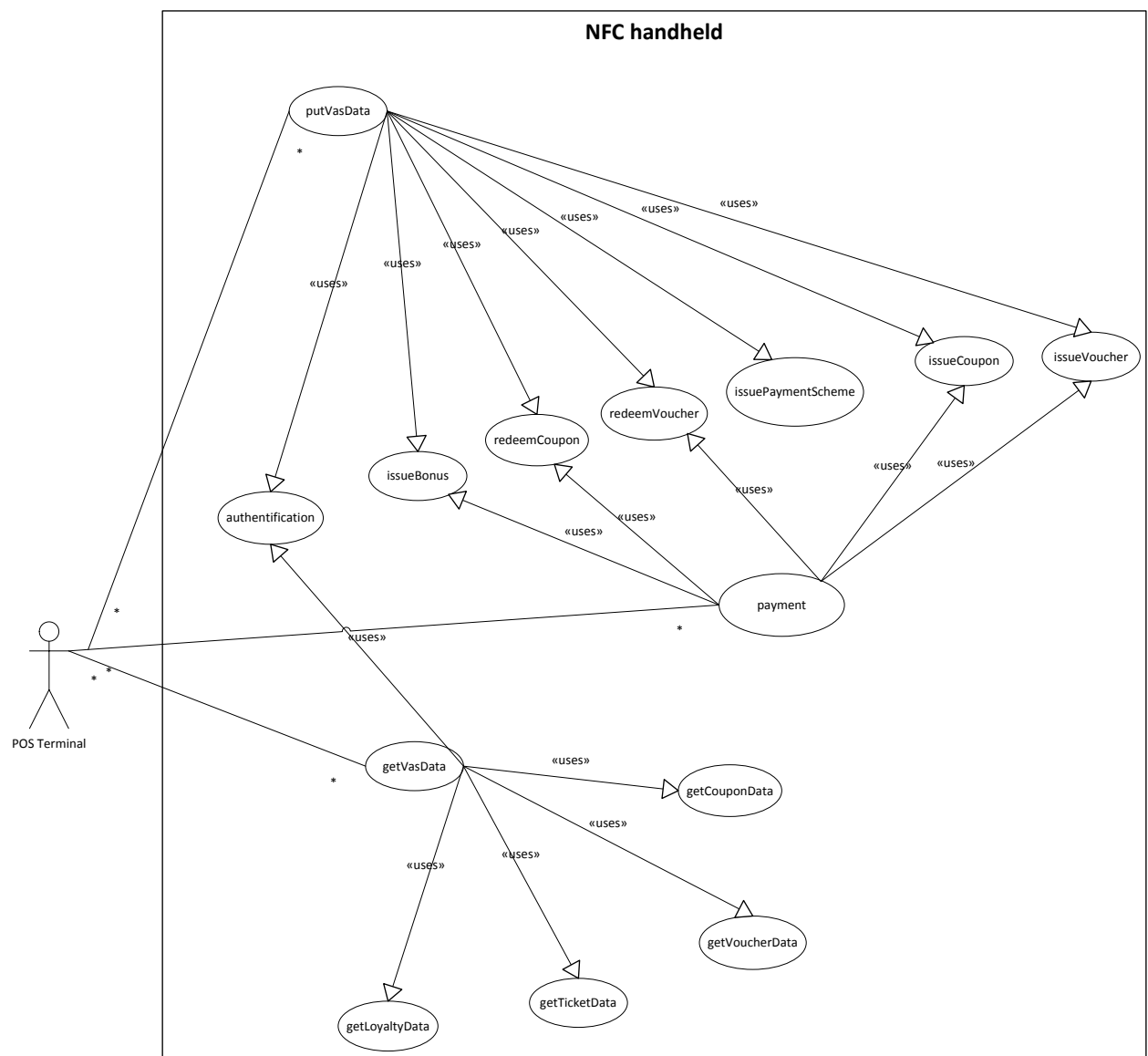
Figure 5 depicts just one of the possible options.



**Figure 5: Multiple Transactions – ouponing/Loyalty/Payment Handling**

## 5 Terminal Use Cases

### 5.1 Use case diagram



**Figure 6: Use case diagram.**

## 5.2 Use cases

A use cases is a description of WHAT a system should do and not HOW a system should do something. The following chapters will give a detailed description of the most important use cases.

### 5.2.1 getVasData

The use case getVasData uses some other use cases such as getCouponData, getVoucherData, to get data from the Wallet.

| No. | Description   |
|-----|---|
| 1   | POS terminal sends a command to the Wallet to get the |

| No. | Description     |
|-----|-----------------|
|     | requested data. |

**Table 2: Options to get data from the Wallet**

### 5.2.2 **getCouponData / getVoucherData /getLoyaltyData**

This use case describes the retrieval of a list of all the user selected coupon/voucher/loyalty data from the Wallet.

| No. | Description  |
|-----|--|
| 1   | POS terminal sends a command to the Wallet to get a list of all selected coupon/voucher /loyalty data. |

**Table 3: Options to get coupon/voucher/ticket/loyalty data from the Wallet**

### 5.2.3 **putVasData**

This use case putVasData combines other use cases such as issueCoupon, issueVoucher, to write/put data to the Wallet in one step.

| No. | Description   |
|-----|---|
| 1   | POS terminal sends a command to the Wallet to write/put data to the Wallet. |

**Table 4: Options to write/put data to the Wallet**

### 5.2.4 **redeemCoupon / redeemVoucher**

This use case confirms the redemption of a list of coupons/vouchers which were **retrieved** from the Wallet.

In case of an error (e.g., NFC communication failure, etc.), the complete transaction will be reversed!

| No. | Description  |
|-----|--|
| 1   | POS terminal sends a command to the Wallet to redeem a list of coupons/vouchers. |

**Table 5: Options to redeem coupons/vouchers in the Wallet**

### 5.2.5 **issueCoupon / issueBonus / issueVoucher / issuePaymentSchemes**

This use case describes the issuing of a list of new coupons/bonus/vouchers/payment Schemes to the Wallet.

The transaction is only considered to be successful if each individual element is successfully issued.

In case of any error the complete transaction has to be reversed!

| No. | Description  |
|-----|--|
| 1   | POS terminal sends a command to the Wallet to issue a list of coupons/bonus/vouchers/paymentSchemes to the Wallet. |

**Table 6: Options to issue new coupons/bonus/voucher**

### 5.2.6 **authenticate**

This use case describes the complete authentication process

Remark: Currently, the <Retailer ID> is used as identification. The authentication between POS terminal and Wallet will be included in a future version of the specification.

| No. | Description   |
|-----|---|
| 1   | POS terminal processes the complete authentication with the Wallet. In the first phase only the retailer ID is written to the Wallet. In the next phases a real authentication process will be implemented. |

**Table 7: Options to process the authentication process with the Wallet**

### 5.2.7 **Payment**

This use case describes the complete payment process and it can be extended with other use cases such as redeemCoupon, issueCoupons etc.

| No. | Description   |
|-----|---|
| 1   | POS terminal processes the complete payment with the Wallet and writes/puts data to the Wallet. |

**Table 8: Options to process the payment process with the Wallet**

## 6 Wallet/Terminal Interface Specification

This section proposes a Wallet-Terminal APDU interface to introduce the new NFC VAS services, such as, loyalty and couponing. The APDU communication is carried out via NFC card emulation mode and the POS serves as master of the communication. Additionally, the data structures used for communication are specified. Currently, no security considerations are taken into account – these will be added at a later stage by adding authentication APDUs, constraints, and access control lists to the specification.

### 6.1 Data Model

The data objects are encoded to Basic Encoding Rules (BER).0

#### 6.1.1 Simple Type

| Type    | Length | Values   | Description  |
|---------|--------|--|--|
| Boolean | 1      | '00', '01' (False, True)   | Boolean  |
| Byte    | 1      | '00' ... 'FF' (-128 ... 127)   | 8-bit signed number                                  |
| UInt8   | 1      | '00' ... 'FF' (0 ... 255)  | 8-bit unsigned number                                |
| Short   | 2      | '00 00' ... 'FF FF' (-32768..32767)  | 16-bit signed short (high byte, low byte)            |
| UInt16  | 2      | '00 00' ... 'FF FF' (0 ... 65535)  | 16-bit unsigned short (high byte, low byte)          |
| UInt32  | 4      | '00 00 00 00' ... 'FF FF FF FF' (0 ... 4294967295)                                   | 32 bit unsigned integer, most significant byte first |
| UInt64  | 8      | '00 00 00 00 00 00 00 00' ... 'FF FF FF FF FF FF FF FF' (0 ... 18446744073709551615) | 64 bit unsigned integer, most significant byte first |

**Table 9: Simple Data Type Table**

#### 6.1.2 Basic Type

The basic data types defined in this section are simple data types embedded into BER-TLV tags and will be used through the protocol specifications.

UniqueID

Unique identifier of a token, needed for later reference on the wallet side.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 20 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 10Table 6.2: Basic Type – UniqueID**

TokenData

Transparent token data, which is not interpreted by the wallet, but by retail system.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 21 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 11Table 6.3: Basic Type – TokenData**

WalletData

This data is interpreted by the wallet, in order to get the loyalty and coupons presented in the wallet.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 22 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 12: Basic Type – WalletData**

ActionSpecifier

Describes the action, which should be performed on an item with a PUT DATA command. Create(0x00), Delete(0x01), Update(0x02), mark as redeem(0x03), Invalidate(0x04).

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 23 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 13: Basic Type – ActionSpecifier**

ProtocolVersion

Contains version information for the used protocol.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 24 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 14: Basic Type – ProtocolVersion**

VASappID

Unique identification of the UICC application “VASappCardlet”.  
This ID is used by the terminal to select the application. (See chapter 4.2.1)

| Field | Type | Length | Value |
|-------|------|--------|-------|
|-------|------|--------|-------|



|       |      |      |       |
|-------|------|------|-------|
| TAG   | Byte | 2    | 9F 25 |
| LEN   |      | var. |       |
| VALUE | Byte | var. |       |

**Table 15: Basic Type – VASapPId**

#### SecuritySpecifier

Describes, what type of security has to be applied for a certain token.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 26 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 16: Basic Type – SecuritySpecifier**

#### Signature

Digital signature for token data.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 27 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 17: Basic Type – Signature**

#### RetailerID

The unique identification of a retailer. This id is used by the wallet as one of the filter criteria to transfer the applicable coupon data to the target POS system. To be noted, we define only the meta-data-structure of the RetailerID in TLV format, and it is open to adopt any standard for its data. For example, use the relevant GS1 standards 0.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 28 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 18: Basic Type – RetailerID**

#### LoyaltyIssuerID

The unique identification number of a loyalty program. This id is used by the wallet as one of the filter criteria to transfer the applicable loyalty data to POS. To be noted, we define only the meta-data-structure of the LoyaltyIssuerID in TLV format, and it is open to adopt any standard for its data. For example, use the relevant GS1 standards 0.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 29 |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 19: Basic Type – LoaytlssuerID**

#### VoucherIssuerID

The unique identification number of a couponing issuer. This id is used by the wallet as one of the filter criteria to transfer the applicable coupon data to POS. To be noted, we define only the meta-data-structure of the VoucherIssuerID in TLV format, and it is open to adopt any standard for its data. For example, use the relevant GS1 standards 0.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 2A |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 20: Basic Type – VoucherIssuerID**

#### PaymentSchemeID

The unique identification number of a payment scheme.

This id is used by the wallet to check if the selected payment scheme is possible. If not, the wallet could suggest the user a valid payment scheme.

| Field | Type | Length | Value |
|-------|------|--------|-------|
| TAG   | Byte | 2      | 9F 2B |
| LEN   |      | var.   |       |
| VALUE | Byte | var.   |       |

**Table 21: Basic Type – PaymentSchemeID**

### 6.1.3 Constructed Types

The constructed data types defined in this section are composed of BER-TLV encoded basic types or further constructed types and will be used through the protocol and memory layout specifications.

NOTE: The sequence of the different tags within constructed tags may be variable.

#### Token

Collection of all data, related to a certain token.

| Field | Type | Length | Value | Mand./Opt. |
|-------|------|--------|-------|------------|
| TAG   | Byte | 1      | 0xB0  | M          |

|                       |            |      |   |
|-----------------------|------------|------|---|
| LEN                   |            | var. | M |
|                       | UniqueID   | var. | M |
|                       | TokenData  | var. | M |
|                       | WalletData | var. | O |
| Further optional data | ....       |      | O |

**Table 22: Constructed Type - Token**

PaymentTokenList

List of payment tokens

| Field         | Type  | Length | Value | Mand./Opt. |
|---------------|-------|--------|-------|------------|
| TAG           | Byte  | 1      | 0xA0  | M          |
| LEN           |       | var.   |       | M          |
| Token 1 ... n | Token | var.   |       | O          |

**Table 23: Constructed Type - PaymentTokenList**

CouponTokenList

List of coupon tokens

| Field         | Type  | Length | Value | Mand./Opt. |
|---------------|-------|--------|-------|------------|
| TAG           | Byte  | 1      | 0xA1  | M          |
| LEN           |       | var.   |       | M          |
| Token 1 ... n | Token | var.   |       | O          |

**Table 24: Constructed Type - CouponTokenList**

LoyaltyTokenList

List of loyalty tokens

| Field         | Type  | Length | Value | Mand./Opt. |
|---------------|-------|--------|-------|------------|
| TAG           | Byte  | 1      | 0xA2  | M          |
| LEN           |       | var.   |       | M          |
| Token 1 ... n | Token | var.   |       | O          |

**Table 25: Constructed Type – LoyaltyTokenList**

## CouponTokenList

List of coupon tokens

| Field         | Type  | Length | Value | Mand./Opt. |
|---------------|-------|--------|-------|------------|
| TAG           | Byte  | 1      | 0xA3  | M          |
| LEN           |       | var.   |       | M          |
| Token 1 ... n | Token | var.   |       | O          |

**Table 26: Constructed Type – CouponTokenList**

## TicketTokenList

List of ticket tokens

| Field         | Type  | Length | Value | Mand./Opt. |
|---------------|-------|--------|-------|------------|
| TAG           | Byte  | 1      | 0xA4  | M          |
| LEN           |       | var.   |       | M          |
| Token 1 ... n | Token | var.   |       | O          |

**Table 27: Constructed Type – TicketTokenList**

## AccessTokenList

List of access tokens

| Field         | Type  | Length | Value | Mand./Opt. |
|---------------|-------|--------|-------|------------|
| TAG           | Byte  | 1      | 0xA6  | M          |
| LEN           |       | var.   |       | M          |
| Token 1 ... n | Token | var.   |       | O          |

**Table 28: Constructed Type – AccessTokenList**

## Tag list

List of Tags

| Field                | Type  | Length | Value | Mand./Opt. |
|----------------------|-------|--------|-------|------------|
| TAG                  | Byte  | 1      | 0x5C  | M          |
| LEN                  |       | var.   |       | M          |
| Tag of Token 1 ... n | Bytes | var.   |       | O          |

**Table 29: Constructed Type – TagList**

## 6.1.4 Tag Overview

| Tag  | Name     |
|------|----------|
| 9F20 | UniqueID |

|      |                   |
|------|-------------------|
| 9F21 | TokenData         |
| 9F22 | WalletData        |
| 9F23 | ActionSpecifier   |
| 9F24 | ProtocolVersion   |
| 9F25 | VASappID          |
| 9F26 | SecuritySpecifier |
| 9F27 | Signature         |
| 9F28 | RetailerID        |
| 9F29 | LoyaltyIssuerID   |
| 9F2A | VoucherIssuerID   |
| 9F2B | PaymentSchemeID   |
| B0   | Token             |
| A0   | PaymentTokenList  |
| A1   | CouponTokenList   |
| A2   | LoyaltyTokenList  |
| A3   | VoucherTokenList  |
| A4   | TicketTokenList   |
| A5   | reserved          |
| A6   | AccessTokenList   |
| 5C   | Tag list          |

**Table 30: Tag overview**

## 6.2 VASAppCardlet APDU Commands

The VASAppCardlet is a JavaCard applet running on UICC and it manages the VAS communication with POS system via NFC card emulation mode. The environment, i.e., Handset/UICC/mWalletPhoneApp should fulfil the requirements defined by GSMA specification according to 00**Error! Reference source not found.**

### 6.2.1 Selection of VAS Application

Like other UICC applets, the VASAppCardlet is identified and selected by its application identifier (AID). The standard selection command 0 is called by POS system to make the VASAppCardlet selected and ready for further APDU processing.

The VASAppAID is the application identifier of the GSMA and related MNO's VASAppCardlet.

The encoding of the VASappCardlet AID is defined according to the following table:

| Meaning   | AID-Coding                                      |
|---|---|
| <i>RID</i><br>This RID is internationally unique and reserved for applications according this specification               | 0xA0:0x00:0x00:0x05:0x59                        |
| <i>Application Code</i>   | 0x00:0x01 (e.g. GSMA application VASappCardlet) |
| <i>Mobile Country Code</i><br>(three digits BCD, right aligned, padded with F, e.g. 0xF2:0x62 Germany)                    | 0xFF:0xXX                                       |
| <i>Mobile Network Code</i><br>(two or three digits BCD, right aligned, padded with F, e.g. Telekom Deutschland 0xFF:0x01) | 0xFF:0xXX or 0xFF:0xXX                          |
| <i>Application Provider Field (optional, length 0-5)</i><br>Additional provider specific data                             | 0xXX:0xXX:0xXX:0xXX:0xXX                        |

**Table 31: VASappCardlet AID**

| Field name | Type  | Length | Value             | Description |
|------------|-------|--------|-------------------|-------------|
| CLA        | Byte  | 1      | 0x00              |             |
| INS        | Byte  | 1      | 0xA4              |             |
| P1         | Byte  | 1      | 0x04              |             |
| P2         | Byte  | 1      | 0x00              |             |
| Lc         | UInt8 | 1      | 0xXX              |             |
| Data       |       |        | VASappCardlet AID |             |
| Le         | Byte  | 1      | Length expected   |             |

**Table 32: Select APDU request**

| Field | Type         | Length | Value | Description                         |
|-------|--------------|--------|-------|-------------------------------------|
| Data  | FCI template | 0..255 |       | File control information (optional) |

|             |       |   |       |                                  |
|-------------|-------|---|-------|----------------------------------|
| Status Code | Short | 2 | XX XX | SW1 SW2 (defined in Table 14)    |
| e.g.        |       |   | 90 00 | Operation successfully completed |
|             |       |   | 6A 82 | File not found                   |

**Table 33: Select APDU response**

### 6.2.2 GetData

With the command GetData, data objects can be retrieved from the handset. The requested objects are specified within the TagList in the data field of the command. The response data field shall be the concatenation of the data objects referenced in the TagList, in the same order. The advantages of this command are that multiple data objects can be read out in one single step, and that the POS side can decide very flexibly, which data objects it wants to read.

| Field | Type    | Length | Value           | Description |
|-------|---------|--------|-----------------|-------------|
| CLA   | Byte    | 1      | 0x00            |             |
| INS   | Byte    | 1      | 0xCB            |             |
| P1    | Byte    | 1      | 0x00            |             |
| P2    | Byte    | 1      | 0x00            |             |
| Lc    | UInt8   | 1      | 0xFF            |             |
| Data  | TagList | Var.   |                 |             |
| Le    | Byte    | 1      | Length expected |             |

**Table 34: GetData APDU request**

| Field       | Type  | Length | Value | Description                            |
|-------------|-------|--------|-------|--|
| Data        |       | 0..255 |       |  |
| Status Code | Short | 2      | XX XX | SW1 SW2 (defined in Table 14)          |
|             |       |        | 90 00 | Operation successfully completed       |
|             |       |        | 67 00 | Wrong data length                      |
|             |       |        | 6A 80 | Incorrect parameters in the Data field |

**Table 35: GetData APDU response**

### 6.2.3 PutData

With the command PutData, data objects can be sent to the handset, e.g. the RetailerID. Furthermore, the status of tokens can be updated, e.g. coupons can be marked as redeemed. For this purpose, the tokens have to be referenced by their UniqueID and an ActionSpecifier has to be given. With the PutData command, the POS side has the

possibility to transport multiple data objects and update multiple tokens in one step flexibly by simply adding them to the data field respectively the corresponding list and adding the list to the data field.

| Field | Type | Length | Value     | Description |
|-------|------|--------|-----------|-------------|
| CLA   | Byte | 1      | 0x00      |             |
| INS   | Byte | 1      | 0xDB      |             |
| P1    | Byte | 1      | 0x00      |             |
| P2    | Byte | 1      | 0x00      |             |
| Lc    | Byte | 1      | Data Size |             |
| Data  |      | 0..255 |           |             |

**Table 36: PutData APDU request**

| Field       | Type  | Length | Value | Description   |
|-------------|-------|--------|-------|---|
| Status Code | Short | 2      | XX XX | SW1 SW2 (defined in Table 14)                               |
|             |       |        | 90 00 | Operation successfully completed                            |
|             |       |        | 9C 01 | Insufficient memory onto the UICC to complete the operation |
|             |       |        | 9C 06 | Access denied   |

**Table 37: Data APDU response**



#### 6.2.4 VASApp Cardlet Response Codes

| SW1 | SW2 | Description  |
|-----|-----|--|
| 90  | 00  | Operation successfully completed (ISO)   |
| 63  | 00  | Unsuccessful authentication (for an ISO Verify). Multiple consecutive failures cause the PIN to block. (ISO) |
| 67  | 00  | Wrong data length (regards L <sub>e</sub> & L <sub>c</sub> ) (ISO)   |
| 69  | 83  | The PIN referenced into an ISO Verify command is blocked. (ISO)  |
| 6A  | 80  | Incorrect parameters in the Data field. (ISO)  |
| 6A  | 82  | File not found. (ISO)  |
| 6A  | 86  | Incorrect parameters P1-P2 (ISO)   |
| 6A  | 88  | Referenced data not found. (ISO)   |
| 6D  | 00  | Wrong instruction code. (ISO)  |
| 6E  | 00  | CLA (for INS) not supported. (ISO)   |
| 9C  | 01  | Insufficient memory onto the UICC to complete the operation  |
| 9C  | 02  | Unsuccessful authentication.   |
| 9C  | 03  | Operation not allowed because of the internal state of the VASapp Cardlet                                    |
| 9C  | 05  | The requested feature is not supported either by the UICC or by the VASapp Cardlet                           |
| 9C  | 06  | Unauthorized (access denied)   |
| 9C  | 07  | An object either explicitly or implicitly involved in the operation was not found                            |
| 9C  | 08  | Object already exists  |
| 9C  | 0B  | The signature provided in a verify operation was incorrect.  |
| 9C  | 0C  | Authentication operation not allowed because specified identity is blocked.                                  |
| 9C  | 0D  | An error occurred. No further information is given.  |
| 9C  | 0E  | Input data provided either in the APDU or by means of the input object is invalid.                           |
| 9C  | 10  | Incorrect P1 value   |
| 9C  | 11  | Incorrect P2 value   |
| 9C  | 12  | Expected length of the received data is not correct.   |

**Table 38: VASappCardlet response status codes**

## 6.3 APDU Call Flows

Based on the defined APDU commands and data model in the previous sections, we can now map the high-level transaction workflows defined in 4.2 to the X-Taps(X=1/2) based APDU call flows. The number of taps required for a specific transaction depends on service provider. In the following subsections, we use a 2-taps pattern as example to show how the mapping works in practice using the defined APDU commands.

### 6.3.1 Call Flow – Coupon Redemption

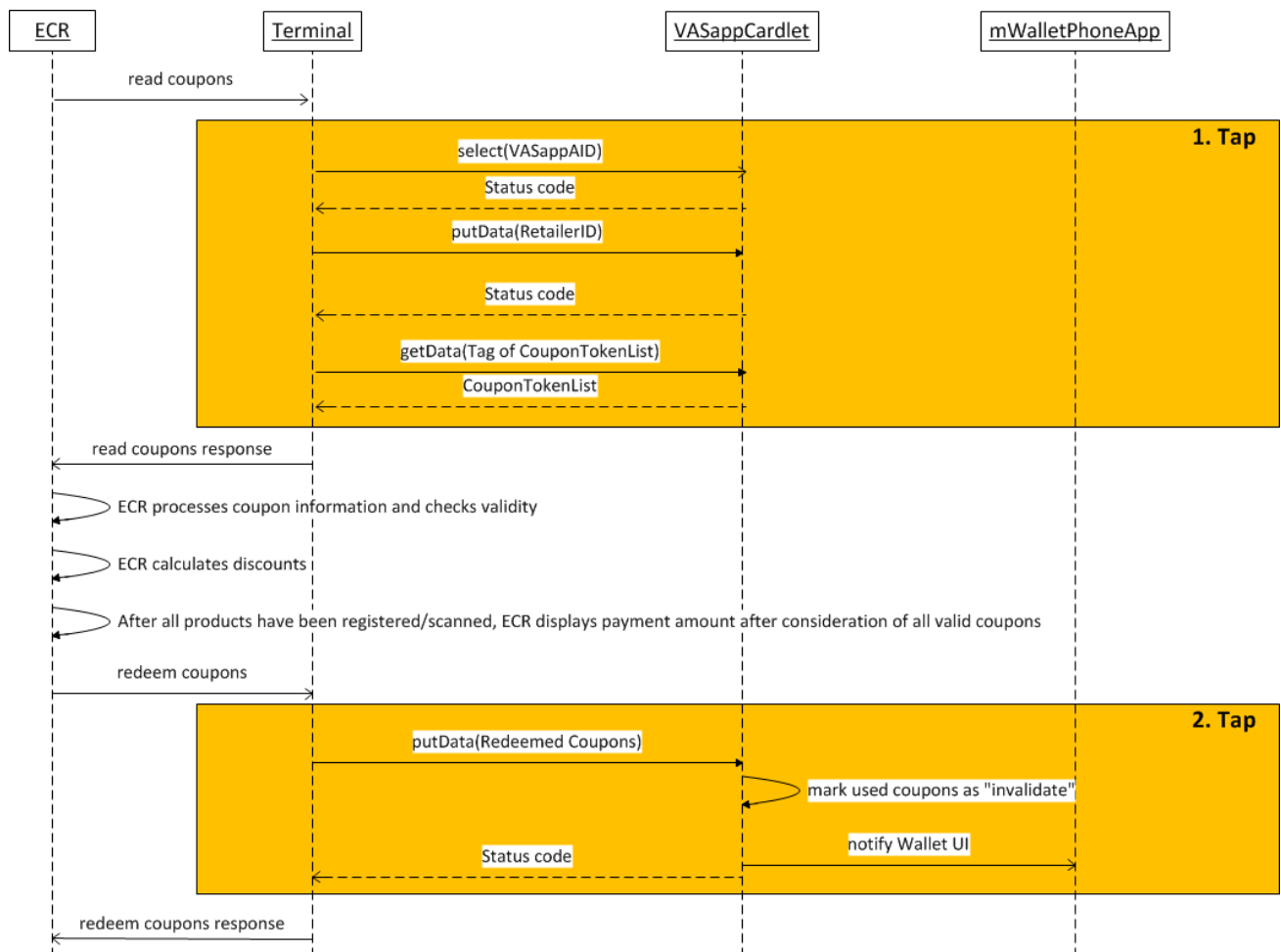
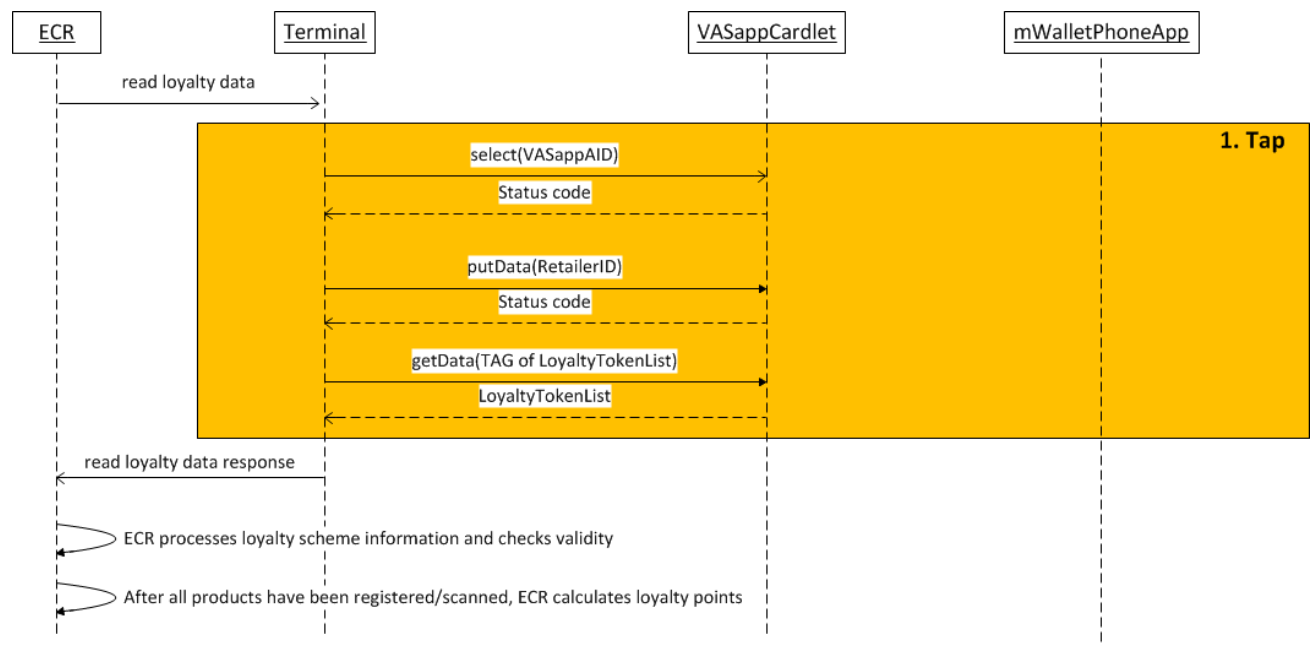


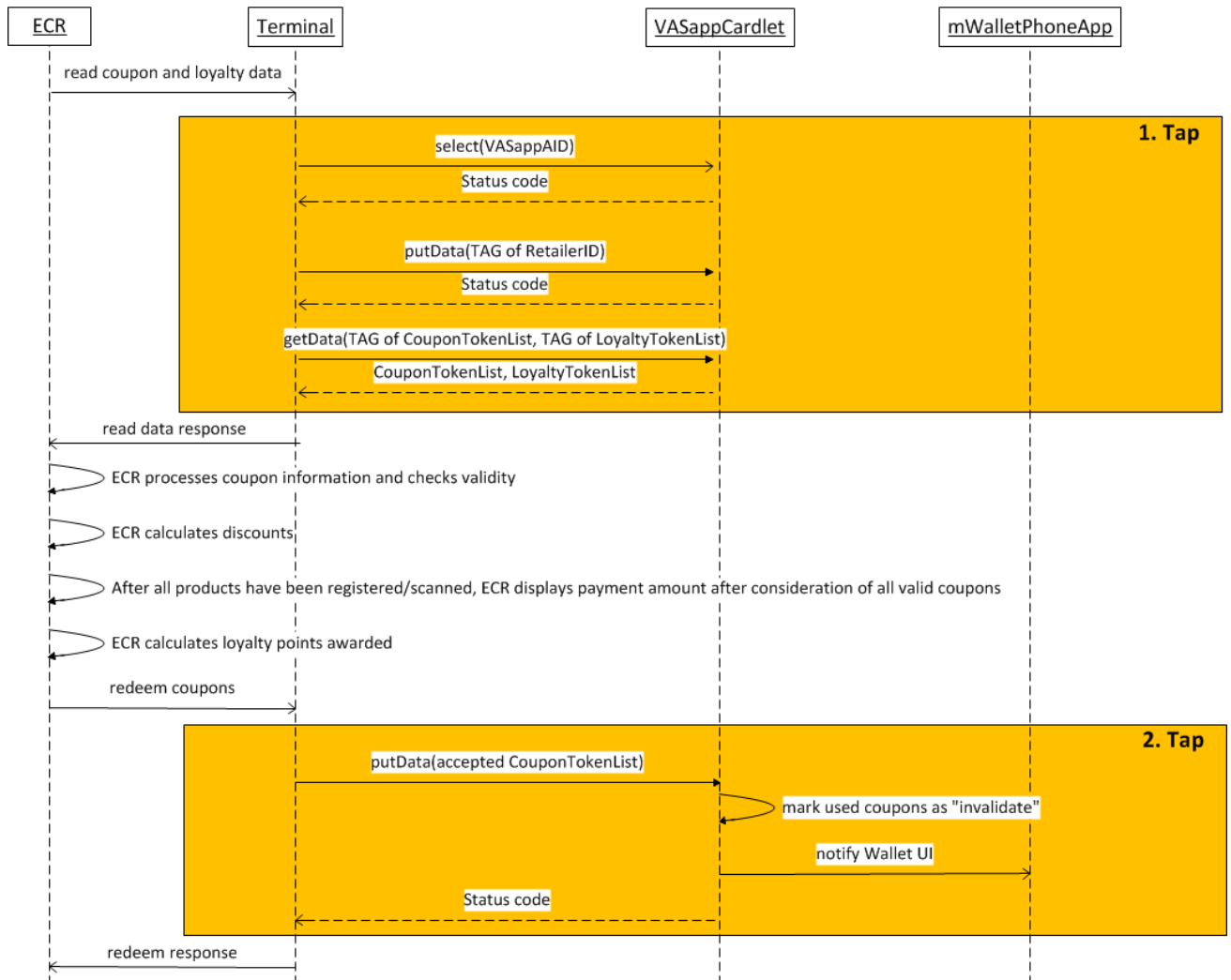
Figure 7: APDU Call-Flow: Coupon Redemption with mWallet

### 6.3.2 Call Flow – Loyalty Handling



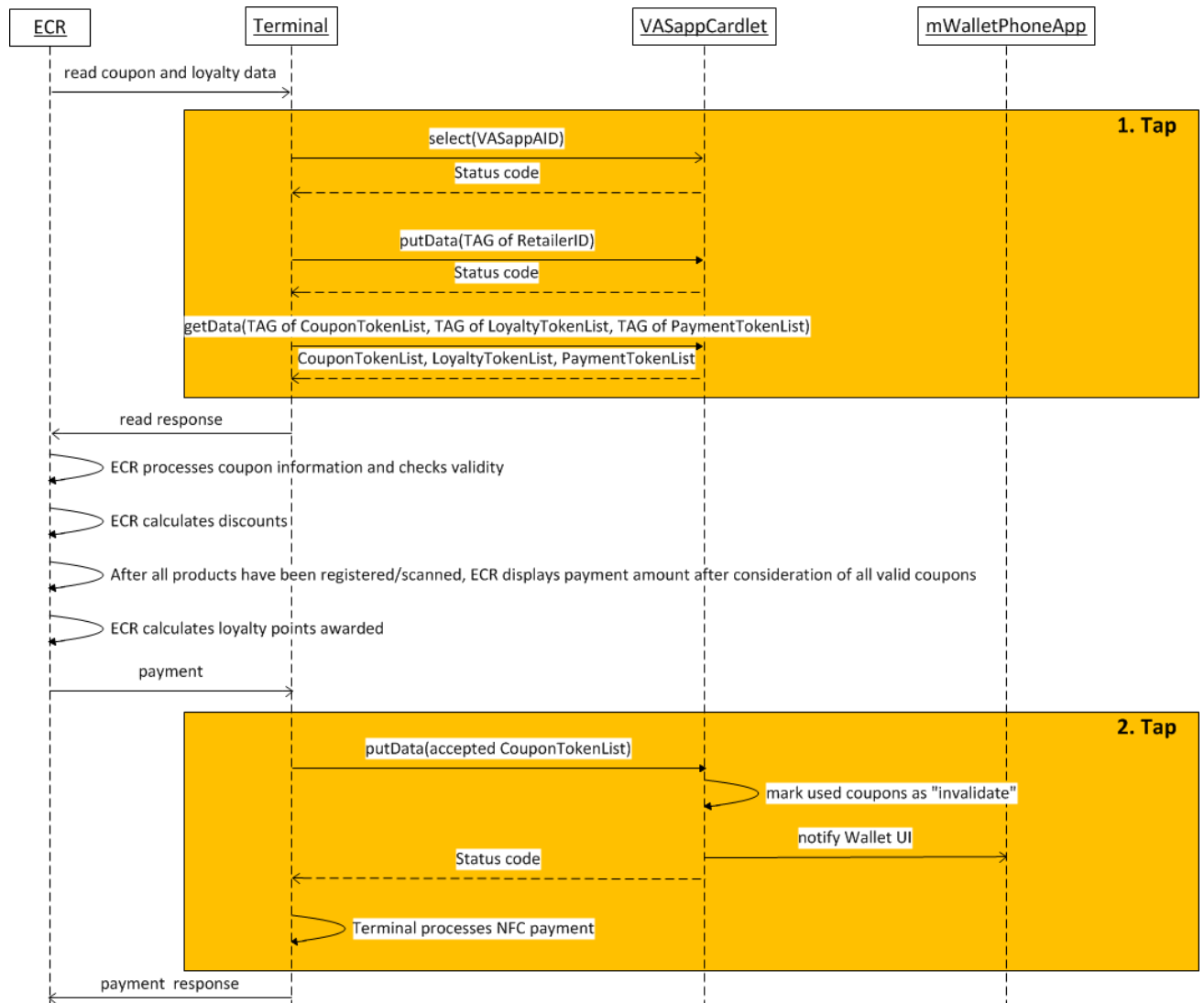
**Figure 8: APDU Call-Flow: Loyalty Handling with mWallet**

### 6.3.3 Call Flow – Couponing/ Loyalty Handling



**Figure 9: APDU Call-Flow: Coupon Redemption and Loyalty Handling with mWallet**

### 6.3.4 Call Flow – Couponing/Loyalty/Payment Handling



**Figure 10: APDU Call-Flow: Coupon/Loyalty/Payment with mWallet**

## 7 Guidelines for Terminal Developers

Questions and Answer Section:

### Question

1. How are multiple TLV objects retrieved using the single GetData command?

Examples:

Retrieve WalletID and list of coupon tokens:

```
send   : 0x00 0xCB 0x00 0x00 0x05 0x5C 0x03 0x9F 0x25 0xA1 0x00
receive: 0x9F 0x25 0x10 0xA0 0x00 0x00 0x00 0x87 0x10 0x03 0xFF 0x49 0x94 0x20
0x89 0xFF 0x01 0x02 0x01
        0xA1 0x17 0xB0 0x15 0x9F 0x20 0x02 0x12 0x34
        0x9f 0x21 0x0D 0x39 0x38 0x32 0x33 0x32 0x36 0x32 0x33 0x36 0x31 0x32
0x30 0x34 0x90 0x00
```

WalletID = A0000000871003FF49942089FF010201

CouponToken:

```
UniqueID =1234
TokenData="9823262361204"
```

Retrieve list of loyalty tokens:

```
send   : 0x00 0xCB 0x00 0x00 0x03 0x5C 0x01 0xA2 0x00
receive: 0xA2 0xFF 0xB0 0xFF 0x9F 0x20 0x02 0x12 0x34 0x9f 0x21 0xFF <loyalty
data> 0x90 0x00
```

```
UniqueID =1234
TokenData=<loyalty data>
```

Retrieve lists of payment and coupon tokens:

```
send   : 0x00 0xCB 0x00 0x00 0x04 0x5C 0x02 0xA0 0xA1 0x00
receive: 0xA1 0x17 0xB0 0x15 0x9F 0x20 0x02 0x12 0x34 0x9F 0x21 0x0D 0x39 0x38
0x32 0x33
        0x32 0x36 0x32 0x33 0x36 0x31 0x32 0x30 0x34 0xA2 0xFF 0xB0 0xFF 0x9F
0x20 0x02
        0x34 0x12 0x9f 0x21 0xFF <loyalty data> 0x90 0x00
```

Coupon:

```
UniqueID =1234
TokenData="9823262361204"
```

Loyalty:

```
UniqueID =3412
TokenData=<loyalty data>
```

Retrieve lists of payment, coupon, loyalty and access tokens:

```
send   : 0x00 0xCB 0x00 0x00 0x06 0x5C 0x04 0xA0 0xA1 0xA2 0xA3 0x00
receive: .....
```

**NOTE:** 0x5C is the tag for a tag list. See [2], chapter 8.5.1.

**NOTE:** from ISO specification [2], chapter 7.4.2: If the information is too long for a single response data field, the card shall return the beginning of the information followed by SW1-SW2 set to '61 XX'. Then a subsequent GET RESPONSE provides 'XX' bytes of information. The process may be repeated until the card sends SW1- SW2 set to '90 00'.

### Question

2. How can multiple TLV objects be written with a single PutData command?

Examples:

Redeem one coupon token:

send : 0x00 0xDB 0x00 0x00 0x0D 0xA1 0x0B 0xB0 0x09 0x9F 0x20 0x02 0x12 0x34  
0x9F 0x23 0x01 0x03  
receive: 0x90 0x00

Token:

UniqueID =1234  
ActionSpecifier=0x03

**NOTE:** If more than one token (per list) is sent, an error in one token will reverse the complete transaction.  
To get a detailed error message, every token has to be send by one putData command.

Send RetailerID to wallet:

send : 0x00 0xDB 0x00 0x00 0x07 0x9F 0x09 0x04 0x47 0x11 0x08 0x15  
receive: 0x90 0x00  
RetailerID = 47110815

### Question

3. How does the Chaining Mode work with GetData/PutData?

When transmitting data too long for one single command, the chaining mechanism, as described in [2], chapter 5.1.1.1, shall be used. Then bit 5 (8 MSB ... 1 LSB) of the CLA byte indicates the last part of a chain:

- If bit 5 is set to 0, then the command is the last or only command of a chain.
- If bit 5 is set to 1, then the command is not the last command of a chain.

## Document Management

### Document History

| Version | Date       | Brief Description of Change                          | Approval Authority | Editor Company /               |
|---------|------------|--|--------------------|--------------------------------|
| 1.0     | 22/03/2013 | First GSMA version submitted for DAG & PSMC Approval | DAG & PSMC#112     | Jürgen Göbel, Deutsche Telekom |
|         |            |  |                    |                                |
|         |            |  |                    |                                |

### Other Information

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at [prd@gsma.com](mailto:prd@gsma.com)

Your comments or suggestions & questions are always welcome.