# Mobile Commerce NFC Coupons and Loyalty Acceptance - Technical Proposal

## Version 1.0

## 01 July 2014

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

### Security Classification: Non-confidential



## Copyright Notice



## Disclaimer



## Antitrust Notice

**Table of Contents**

# 1 Introduction

## 1.1 Overview

This technical proposal follows on from the "Mobile Commerce in Retail: Loyalty and Couponing" white paper from the GSMA [6].

Mobile network operators (MNOs) in a national market can work together to develop a consistent approach to loyalty and couponing. A consistent approach will generate economies of scale across the value chain. Similarly, the use of open standards for technical delivery can engender consistency and simplicity, helping extend the reach of loyalty programmes and generate scale.

Until now, there has been no single, common, open specification for a digital platform that combines offers, coupons and loyalty to form an interoperable end to end solution; beneficial to merchants, manufacturers and wider service provider ecosystems. This document bridges that gap as it outlines an architecture model and the component interfaces that can be adopted to produce a common approach for coupon and loyalty transactions using mobile handsets. As this proposal is aimed at multiple markets and a global audience, it covers multiple propositions and a maximum number of possible use cases. Implementation-specific details, specifically security, have not been specified. However, a security model detailing the options and a generic cost/benefit analysis is provided. It is not possible to provide detailed specifications at this proposal stage as these will be implementation-specific.

Yet this document is comprehensive enough to enable all ecosystem players to assess the overall architecture, and be able to plan the provision of an interoperable loyalty and couponing solution within specific markets.

## 1.2 Scope

The GSMA wallet proposition as described in the Mobile Wallet whitepaper [7] offers an ecosystem architecture incorporating the universal integrated circuit card (UICC) as a multi-functional smartcard hosting several service applets. These applets are written in Java and implement service-specific functions such as payment, access control and public transport ticketing. These functions are typically used at contactless acceptance points equipped with near field communication (NFC) readers.

This proposal will utilise the UICC as described above and provide an architecture proposal for digital couponing and loyalty, taking into account its existing ecosystem, and any new development that needs to take place.

The following chapters are structured as follows:

- Chapter 2 Use cases: Starting with specific real-world use cases around coupons and loyalty, this document will outline the overall customer journey, user experience at the PIN entry device (PED) and at the point of sale (POS).
- Chapter 3 Solution architecture: High level solution architecture, along with its various assumptions and limitations.

- Chapter 4 Data and control flows: Description of the overall data model and messaging flows that make the system work. This includes the coupon information flow for the overall system, specific messaging to support the distribution of coupons to a handset and subsequent retrieval of coupons from the handset at the redemption terminal.
- Chapter 5 Ecosystem considerations: Wider ecosystem considerations and a suggested list of items that merchants, manufacturers and other loyalty providers need to accomplish to support this architecture.
- Chapter 6 Application programming interface (API): Details around each of the APIs including their scope, specific use cases, any special considerations and message flows. The APIs defined in this document are to complement existing, traditional coupon and loyalty systems, and work with existing backend technologies, where possible.
- Chapter 7 Security: The security discussion includes various models, levels and the relative cost implications.
- Annex A: An example of coupons, loyalty cards and their acceptance.
- Annex B: System diagrams.
- Annex C: VAS Applet data structure.
- Annex D: Inter-process Communication – Android OS.

### 1.2.1 Guiding principles

The following are the guiding principles considered while developing the overall architecture:

- **Minimum effort by merchants**: Limit the technical effort to enable these propositions. The majority of the work remains with MNOs and their vendors to develop the necessary infrastructure and to put it in place. Effort needed by electronic point of sale (ePOS) and PED vendors is to be generic and as such could be reused in multiple implementations.
- **Flexible architecture**: Support multiple business use cases and scenarios to cater for different geographies and varied needs depending on merchant type.
- **Contactless technology agnostic**: Architecture supports usage of any other bidirectional communication systems between the handset and retail system. We have used the NFC card emulation mode as a reference proposal in this document. Further technical work will be needed if another communication system is used.
- **Abstract the complex handset workflow**: Propose high level APIs that mask the complexity of implementing an applet and communicating directly with it. Service providers such as merchants and loyalty providers can focus on developing their apps and their customer journey and not on the technical details of deploying value-added services to the applet.

### 1.2.2 Out of scope

The items listed as ‘out of scope’ within the document are items that are directly related to the work in this project but are deemed out of the scope for this version of the documentation.

**Data sharing arrangements**: It is expected that the information relating to consumer transactions and coupon preferences will become a valuable commodity to merchants, manufacturers and advertising agencies. This document will focus on the mechanism for transactions and redemption, but will not specify a mechanism for data tracking, or sharing information gathered using such a system.

**Data privacy considerations:** In order to build trust and encourage user adoption of new and innovative services, the GSMA urges service provides to ensure that any propositions respect mobile users' privacy and consider the GSMA's Mobile Privacy Principles [8] and Privacy Design Guidelines [9] when developing such services. In particular, user trust can be engendered when service providers offer:

- Transparency and notice – helping users understand who has access to their personal data, what they intend to do with it and why, and any implications to their privacy
- Choice and control – enabling users to express granular preferences over access and use of their personal data, but in non-frustrating ways
- Security (of their data and their authentication) – Making users feel confident that their personal information (whether on device or held by service providers and 3rd parties) is secure
- Ability to disassociate with a device / SIM – Users should know whether it's possible to wipe clean or transfer their data elsewhere (Particularly important when the device is transferable/sold)

Detailed data privacy considerations are out of scope for this proposal and member implementations are expected to vary in how they address these.

**Ticketing**: Ticket distribution and redemption is seen as a concept with very similar mechanisms to coupon distribution and redemption. This is recognised, but ticket systems are not part of this document.

**Campaign management** (and associated business process). Campaign management is seen as a large part of the coupon and loyalty ecosystem and can be handled in many different ways in backend systems. This process is outside of the architecture and data channel requirements of this document.

**Offer creation**: This proposal assumes that the creation of offers is a propriety backend operation that is deemed out of scope for this proposal. This proposal will focus on the transfer and usage of an offer that has already been created.

**Offline validation**: This proposal assumes that the process of validating a coupon will take place as a backend system using information about an offer that is referenced using a coupon code. This proposal does not deal with the process of passing encrypted transaction information to the terminal that can be used to validate the coupon without a live connection to the backend.

**Validation of a coupon at the PIN entry device (PED):** Extended information needed for validation of coupons and loyalty IDs at the PED are not covered within this version of the proposal.

### 1.2.3 Considerations for future releases

This section specifies a number of topics that are planned to be covered in future versions of this document.

**Analytics**: A future version of this documentation could include information on how analytic information can be passed through the system to track consumer behaviour such as usage patterns and drop off rates for offers.

**Offline validation**: A future version of this proposal will need to specify the data set to be passed to the terminal in order to allow for validation of coupons and loyalty without the need to communicate with an online backend system. This will allow for quick transactions at the point of sale and increase the speed of customers moving through the checkout.

**Identity**: The consumer's identity could potentially be used to reference a payment and/or value-added service (VAS) system in a backend system and thus allow a user to pay for goods or services at a terminal using a personal identity reference. This is seen as a future feature of the technical work.

**Enablement/on-boarding of merchants/manufacturers**: This version of the proposal does not include information on how merchants or manufacturer organisations will be enabled as service providers within the system. This will be included in a future version.

**Coupon clearance identifier for retrieving coupons from handsets**: In this version of the proposal, it is assumed that coupon clearance information will be stored within the backend system. It is expected that a future version of this proposal could include details of how this information could be transferred through the handset.

**Application using alternate technologies for handset/ePOS communication**: This version of the proposal will focus on using NFC secure element card emulation mode technology as the method to transfer information from the handset to the terminal. It is expected that a future version of this documentation could also outline a mechanism to transmit information using quick response (QR) codes, host card emulation (HCE), Bluetooth low energy (BLE) and other means.

**Standardisation of coupon and loyalty validation**: It is assumed that a standard for messaging and data structures will be needed between the point of sale and backend system. This is expected to become "API 7" in a future version of this proposal.

**Offers coming back from point of sale**: Offers being delivered directly from the POS to the handset via the same NFC channel, either within the same tap, or as a separate new one. This is simply an extension of the existing 'new offers' use case, but because of its instant nature (at a time when the customer is at the checkout), can be a beneficial customer journey.

**User Experience (UX) guidelines:** A consistent user experience for users and POS operators across multiple handsets and ePOS systems is an important consideration for implementations. However, UX guidelines are not appropriate for this particular document and are considered out of scope.

## 1.3 Definitions

| Term | Description |
|---|---|
| App | An abbreviation for the term 'software application', typically installed on a mobile handset. |
| Applet | A small software application that is executed on a UICC. |
| Application | A software program that is executed on the mobile device. See app above. |

| Term | Description |
|---|---|
| Backend system | The retail transaction system, loyalty scheme management system and retail infrastructure that exists beyond the ePOS. |
| Manufacturer | An identity for an organisation or product that differentiates that organisation or product from other organisations or products that operates in the same fields. |
| Cloud system | A web-based service to store and retrieve information directly on the handsets and other devices. It typically requires needs a network connection for communication with devices. |
| Contactless | A short range, wireless communication protocol that allows devices to transfer information without physical contact. |
| Coupon | A paper or digital code that can be presented by a consumer to initiate a discount on specified goods or services. Coupons can be unique to a consumer, or for general use. See [6] |
| Loyalty scheme | A reward system that allows members to gain rewards for repeat business. See [6] |
| Cryptographic Nonce | A cryptographic nonce is an arbitrary number or bit string used only once, in security engineering. |
| Redemption | The act of exchanging a coupon for goods, services or loyalty points to the value of the coupon. |
| Stamp Card | A loyalty scheme that allows the consumer to collect a series of stamps relating to visits or purchases. Once a stamp card is completed or milestones on the card are reached, the consumer is offered a reward. |
| Token | Any form of a data that is needed to be stored on the VAS applet. For this proposal, it can be a coupon ID or loyalty ID. |
| Transaction | The exchange of any combination of goods, coupons, payment, loyalty points or services. |
| Validation | A process to determine whether a service such as a loyalty scheme, offer or coupon can be used at a specific time by a consumer who meets qualifying criteria defined by the scheme, offer or coupon's business rules. |
| Wallet | A set of services that provides various user applications secure access to store customers' card credentials, as described in the Mobile Wallet Whitepaper [7]. A wallet can provide a customer facing user interface to a subset of these services. |

## 1.4 Abbreviations

| Term | Description |
|---|---|
| AID | Application ID |
| APDU | Application Protocol Data Unit |
| API | Application Programming Interface |
| BCD | Binary Coded Decimal |
| CVM | Cardholder Verification Method |
| CW | Core Wallet |
| EMV | Europay MasterCard Visa (Card - Terminal Standard) |
| ePOS | Electronic Point Of Sale |

| Term | Description |
|---|---|
| FMCG | Fast Moving Consumer Goods (Branded items available through multiple merchants) |
| GS1 | Name of global supply chain standards organisation (http://www.gs1.org/) |
| GSMA | GSM Association (http://www.gsma.com/) |
| KB | Kilobyte |
| IPC | Inter-process Communication (Messaging between applications) |
| MNO | Mobile Network Operator |
| NFC | Near Field Communication |
| OTA | Over The Air (data/Wi-Fi connection) |
| OS | Operating System |
| PED | PIN Entry Device |
| PIN | Personal Identifier Number |
| POI | Point of Interaction |
| POS | Point Of Sale |
| PPSE | Proximity Payment System Environment |
| PSP | Payment Service Provider |
| QR | Quick Response (QR code - Matrix tag) |
| RID | Registered Application Provider Identifier |
| SDK | Software Development Kit |
| SP-MMI | Service Provider Man-Machine Interfaces |
| UI | User Interface |
| UICC | Universal Integrated Circuit Card |
| UN | Unpredictable Number |
| UX | User Experience |
| VAS | Value-Added Service |

## 1.5 Reference Documents

| Ref | Doc Reference | Title |
|---|---|---|
| [1] | NFC.11 | NFC.11 GSMA NFC Wallet-POS Proposal (Version 1.0) http://www.gsma.com/digitalcommerce/gsma-nfc-wallet-pos-proposal-v1-0 |
| [2] | NFC.10 | NFC.10 GSMA NFC Core Wallet Requirements (Version 2.0) http://www.gsma.com/digitalcommerce/nfc-core-wallet-requirements-version-2-0-august-2013 |
| [3] | RFC 2119 | “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997 http://www.ietf.org/rfc/rfc2119.txt |
| [4] | GS1-DCM | “Digital Coupon Management” GS1, Issue 1.0, Jun-2012 http://www.gs1.org/docs/gsmp/b2c/Digital_Coupon_Management_i1.pdf |

| Ref | Doc Reference | Title |
|---|---|---|
| [5] | Mobile Commerce in Retail | GSMA Mobile Commerce White Paper: Mobile Commerce in Retail<br>http://www.gsma.com/digitalcommerce/mobile-commerce-in-retail-white-paper-25th-july-2013 |
| [6] | Mobile Commerce in Retail: Loyalty and Couponing | GSMA Mobile Commerce White Paper: Mobile Commerce in Retail: Loyalty and Couponing<br>http://www.gsma.com/digitalcommerce/white-papermobile-commerce-in-retail-loyalty-and-couponing-january-2014 |
| [7] | The Mobile Wallet White Paper | GSMA Mobile NFC White Paper: The Mobile Wallet (Version 1.0)<br>http://www.gsma.com/digitalcommerce/wp-content/uploads/2012/10/GSMA-Mobile-Wallet-White-Paper-Version-1-0.pdf |
| [8] | Mobile Privacy Principles | Mobile Privacy Principles - Promoting a user-centric privacy framework for the mobile ecosystem<br>http://www.gsma.com/publicpolicy/mobile-and-privacy/mobile-privacy-principles |
| [9] | Privacy design guidelines for mobile applications | Privacy design guidelines for mobile application development<br>http://www.gsma.com/publicpolicy/mobile-and-privacy/design-guidelines |
| [10] | ISO-8601 | "Data elements and interchange formats – Information interchange – Representation of dates and times"<br>http://www.iso.org/ |
| [11] | ISO/IEC 14443 | ISO 14443 defines a protocol stack from the radio layer up to a command protocol<br>http://www.iso.org/ and http://www.iec.ch/ |
| [12] | ISO/IEC 18092 | ISO/IEC 18092:2004 Near Field Communication Interface and Protocol<br>(NFCIP-1)<br>http://www.iso.org/ and http://www.iec.ch/ |
| [13] | ISO/IEC 7816 | "Identification cards -- Integrated circuit cards"<br>http://www.iso.org/ and http://www.iec.ch/ |
| [14] | ISO 639 | International Standard for language codes. Establishes internationally recognised codes (either 2, 3, or 4 letters long) for the representation of languages or language families.<br>http://www.iso.org/ |
| [15] | ITU-T E.212 | International Telecommunication Union (ITU) standard: Mobile Network Codes (MNC) for the international identification plan for public networks and subscriptions<br>http://www.itu.int/dms_pub/itu-t/opb/sp/T-SP-E.212B-2013-PDF-E.pdf |
| [16] | 3rd Party Patents | Aspects of this technical proposal may correspond to patents owned by various third parties. Interested parties can contact digitalcommerce@gsma.com for further information. |

# 2 Use cases

## 2.1 Traditional use cases

A traditional retail transaction consists of a series of steps that take place at the time of checkout.

A consumer will present a ‘basket’ of goods or services that could be scanned one-by-one using a product code reader such as a barcode reader or entered manually into an ePOS by an ePOS operator.

The consumer could present one or more coupons that relate to offers that they have received. Each coupon could be visually checked by the operator before the coupon code is either scanned by the product reader or entered manually onto the ePOS.

The consumer could present one or more loyalty codes or loyalty cards to be updated with information about the current transaction. Depending upon the loyalty scheme, the consumer might use loyalty points as part or complete payment for the goods or services in the transaction.

Once any discounts have been applied, the invoice total is calculated and the consumer is able to pay using any combination of cash, credit cards, debit card, gift vouchers or store credit.

Upon the transaction being completed, the goods or services are released to the consumer along with any incentives that they might be entitled to in the form of free gifts, loyalty points or offers that might be used in the future.

This cycle is modelled in Figure 1, ‘Traditional Retail Transaction Flow’

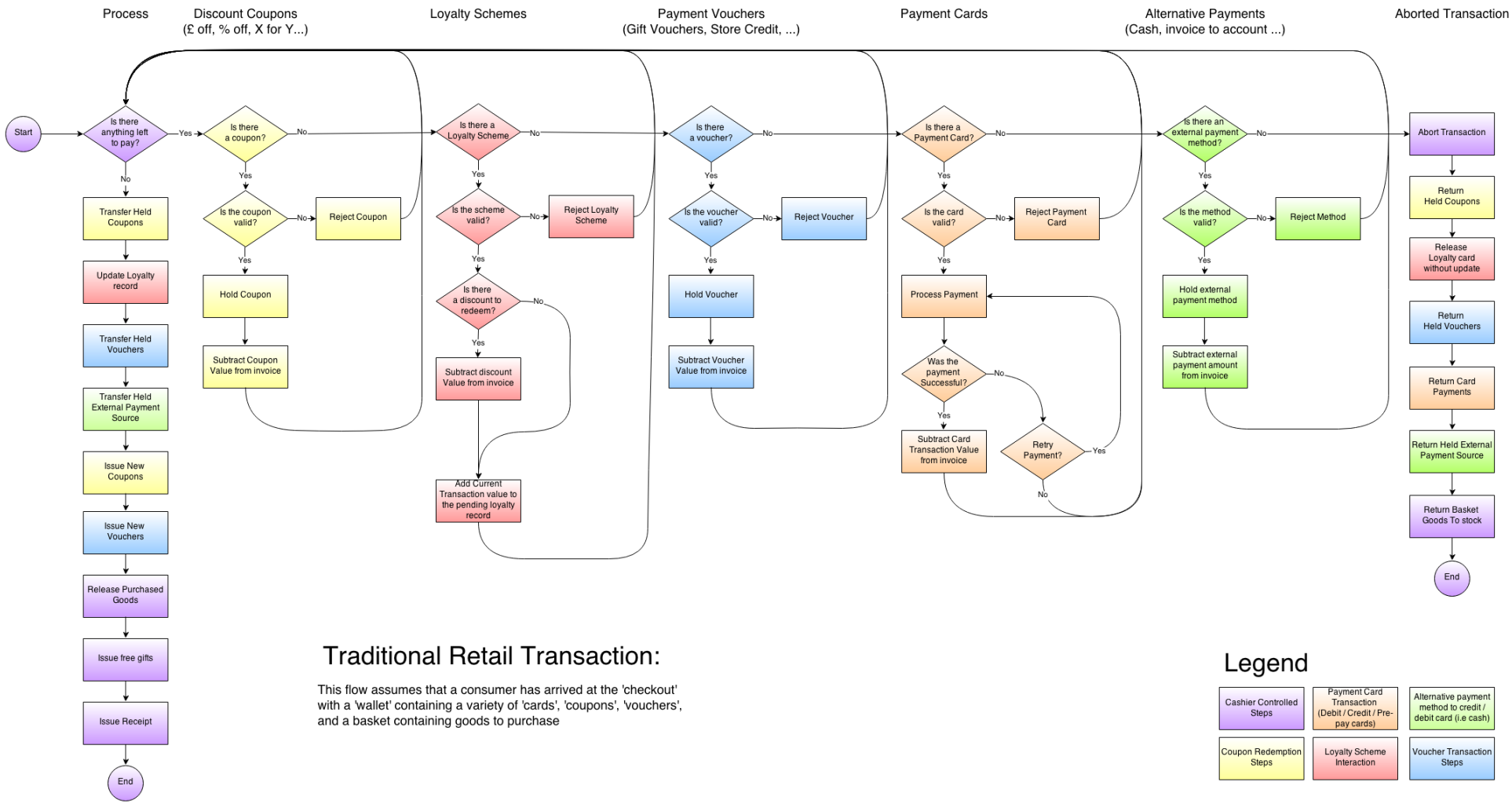


Figure 1**: Traditional Retail Transaction Flow**

This proposal will focus on the coupon, loyalty and voucher (as a specific type of coupon) stages of the retail transaction flow. These are coloured yellow, red and blue in Figure 1.

## 2.2 Coupons

### 2.2.1 High-level use cases

This proposal seeks to enable the high level coupon use cases described in the GS1 Digital Coupon Management Process [4], targeting some specific steps and also adding mobile-specific steps. The GS1 process is defined as:

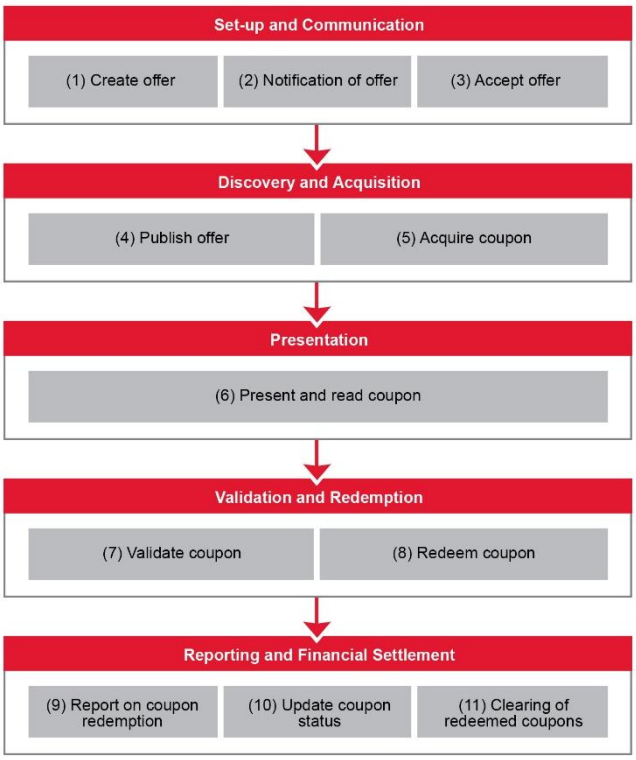


Figure 2**: GS1 specification for coupon validation steps (Reproduced from the 'Digital Coupon Management Standard Specification' Issue 1.0, Jun 2012 - [4])**

For the purposes of this proposal, the flow is defined as:

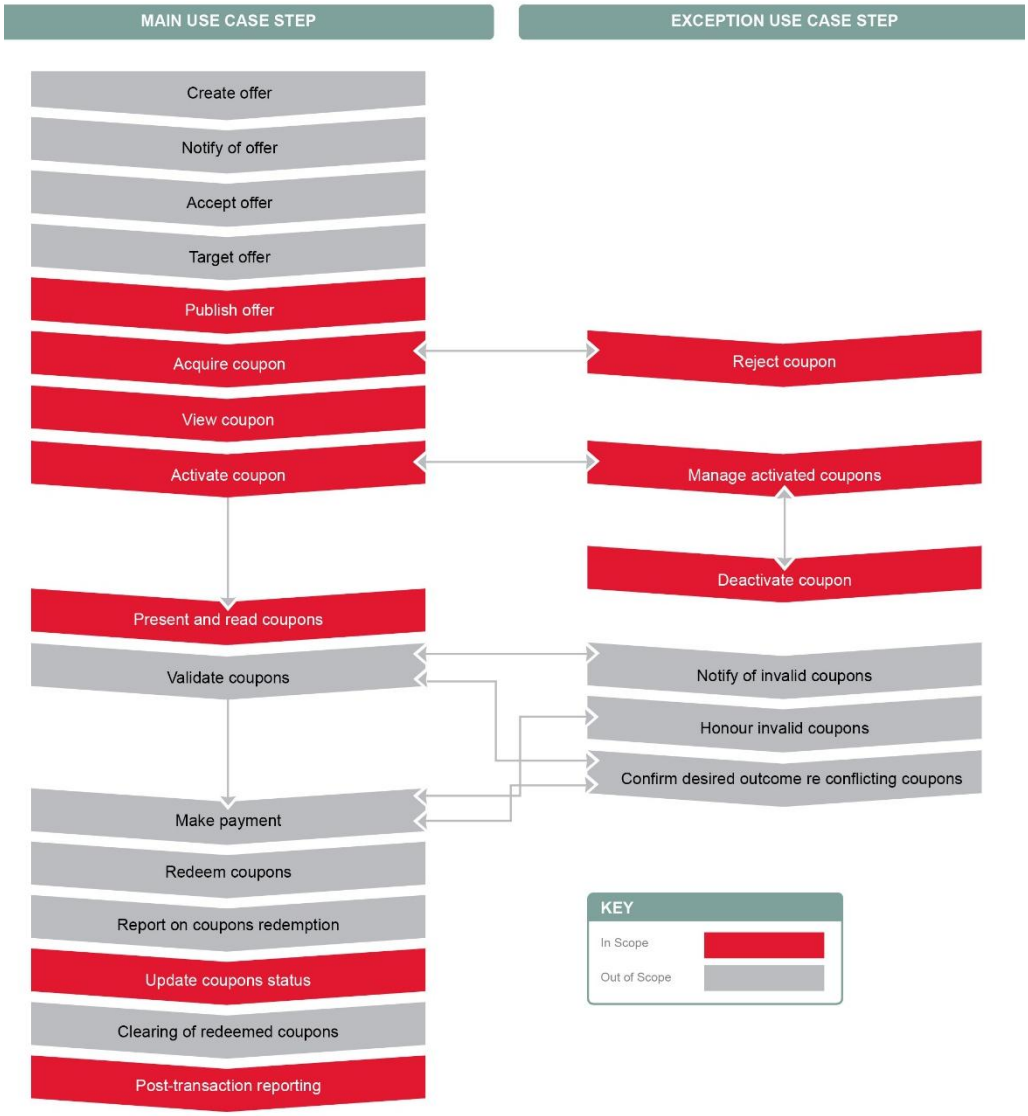


Figure 3**: Coupon use case steps**

GS1 defines a number of defined roles for system actors; such as offer issuer and offer validator. This document adopts the standard GS1 terminology, which can be found defined fully in section 3.3 of the GS1 specification [4]. A summary of GS1 roles and responsibilities is defined in Table 1.

| GS1 Role | Description |
|---|---|
| Offer Issuer | Party issuing the coupons, bearing the commercial and financial responsibility for the coupons. |
| Offer Awarder | Party responsible for the redemption of the coupon. |
| Offer User | Customer using the coupon when making a purchase, and ultimately receiving the reward. |
| Offer Distributor | Party responsible for the distribution of the coupons to the consumers on behalf of the Offer Issuer. |

| GS1 Role | Description |
|---|---|
| Offer Validator | Party assisting the Offer Awarder in the validation of the coupons by offering a centralised service. |
| Offer Awarder Clearing Agent | Party responsible for the financial clearance of redeemed coupons on behalf of the Offer Awarder. |
| Offer Issuer Clearing Agent | Party responsible for the financial clearance of redeemed coupons on behalf of the Offer Issuer. |

**Table 1: System actors defined in the GS1 Digital Coupon Management specification [4]**

For the purpose of this document, the use case steps are described as:

| Coupon Use Case Step | Precise Description | Scope |
|---|---|---|
| Create offer | The Offer Issuer creates the initial offer. | Out of scope |
| Notify of offer | The Offer Issuer communicates the coupon offer details to the Offer Awarder. | Out of scope |
| Accept offer | The Offer Awarder accepts to support the offer at his premises, for example at the POS. | Out of scope |
| Target offer | The Offer Distributor uses data held regarding Offer Users to choose target recipients for the offer | Out of scope |
| Publish offer | The Offer Distributor publishes the offer for access by potential Offer Users via the MNO wallet app and/or service provider app. This can be achieved in 3 ways:<br>• Offer Distributor advertises the coupons program to consumers<br>• Offer User subscribes to Offer Distributors coupons program<br>• Offer Distributor publishes offers to subscribed users. | In scope |
| Acquire coupon | The Offer User acquires a digital coupon via the MNO wallet app or service provider app. | In scope |
| Reject coupon | The Offer Users' request to acquire a digital coupon via the MNO Wallet App or service provider app is rejected due to coupon rules. | In scope exception |
| View coupon | The Offer User views the digital coupon via the MNO wallet app or service provider app. | In- scope |
| Activate coupon | The Offer User marks the digital coupon to be automatically presented to the next potential Offer Awarder via the MNO wallet app and/or service provider app. | In scope |

| Coupon Use Case Step | Precise Description | Scope |
|---|---|---|
| Manage activated Coupons | The Offer User is informed that this coupon cannot be activated without the deactivation of some other coupons and is presented with a list of currently activated coupons from which to choose coupon(s) to be deactivated, via the MNO wallet app and/or service provider app. | In scope exception |
| Deactivate coupon | The Offer User marks the digital coupon not to be automatically presented to the next potential Offer Awarder via the MNO wallet app and/or service provider app. | In scope exception |
| Present and Read coupons | The Offer User presents relevant activated digital coupons to the potential Offer Awarder by physically presenting the mobile handset to an NFC reader. In case an implementation only passes a customer identifier, Offer Awarder will use this to download the coupons from an online service. | In scope |
| Validate coupons | The Offer Awarder validates whether each digital coupon is authentic and whether the purchase meets the agreed business rules for all digital coupons without conflicts. | Out of scope |
| Notify of invalid coupons | The Offer Awarder notifies the Offer User of invalid digital coupon(s). | Out of scope |
| Honour invalid coupons | The Offer Awarder awards the Offer User invalid digital coupon(s) as good will. | Out of scope |
| Confirm desired outcome on conflicting coupons | The Offer Awarder requests the Offer User to clarify which conflicting digital coupon(s) to validate and which not to. | Out of scope |
| Make payment | The Offer User pays the remaining balance, if any, for their purchase. | Out of scope |
| Redeem coupons | The Offer Awarder awards the Validated digital coupons to the Offer User and records the redemption information in his system. | Out of scope |
| Report on coupons redemption | The Offer Awarder communicates the redemption information to the Offer Issuer. | Out of scope |
| Update coupons status | The Offer Distributor updates the status of the redeemed coupons and notifies the Offer User. | In scope |
| Clearing of redeemed coupons | The Offer Issuer or the Offer Issuer Clearing Agent checks the awarded coupon transmissions and whether the rules were applied, and together with the Offer Awarder or the Offer Awarder Clearing Agent agrees on the coupons to be compensated. | Out of scope |
| Post-transaction reporting | The wallet app server records contextual information regarding redeemed coupons and provides agreed information to Offer Distributor. | In scope |

**Table 2: Coupon use caseuse case – step descriptions**

These descriptions are based on GS1's Digital Coupon Management Process [4] use case step descriptions, where applicable, as applied to mobile NFC couponing specifically.

### 2.2.2 Assumptions

Out of scope use case steps are enabled by other parties in the ecosystem, often by existing processes.

Few of the steps described above, though distinct, can be implemented as a single step depending on the desired user experience. As an example, this can be achieved by merging '*Acquire coupon*' and '*Activate coupon*', though this could lead to undesirable effects on the quality of user experience.

'Make payment' refers to a payment made by any acceptable means for the merchant.

## 2.3 Loyalty

### 2.3.1 High-level use cases

This proposal seeks to enable all existing loyalty processes enabled by the transmission of a loyalty ID to the merchant systems at the point of sale. For the purposes of this proposal, the flow is defined as.

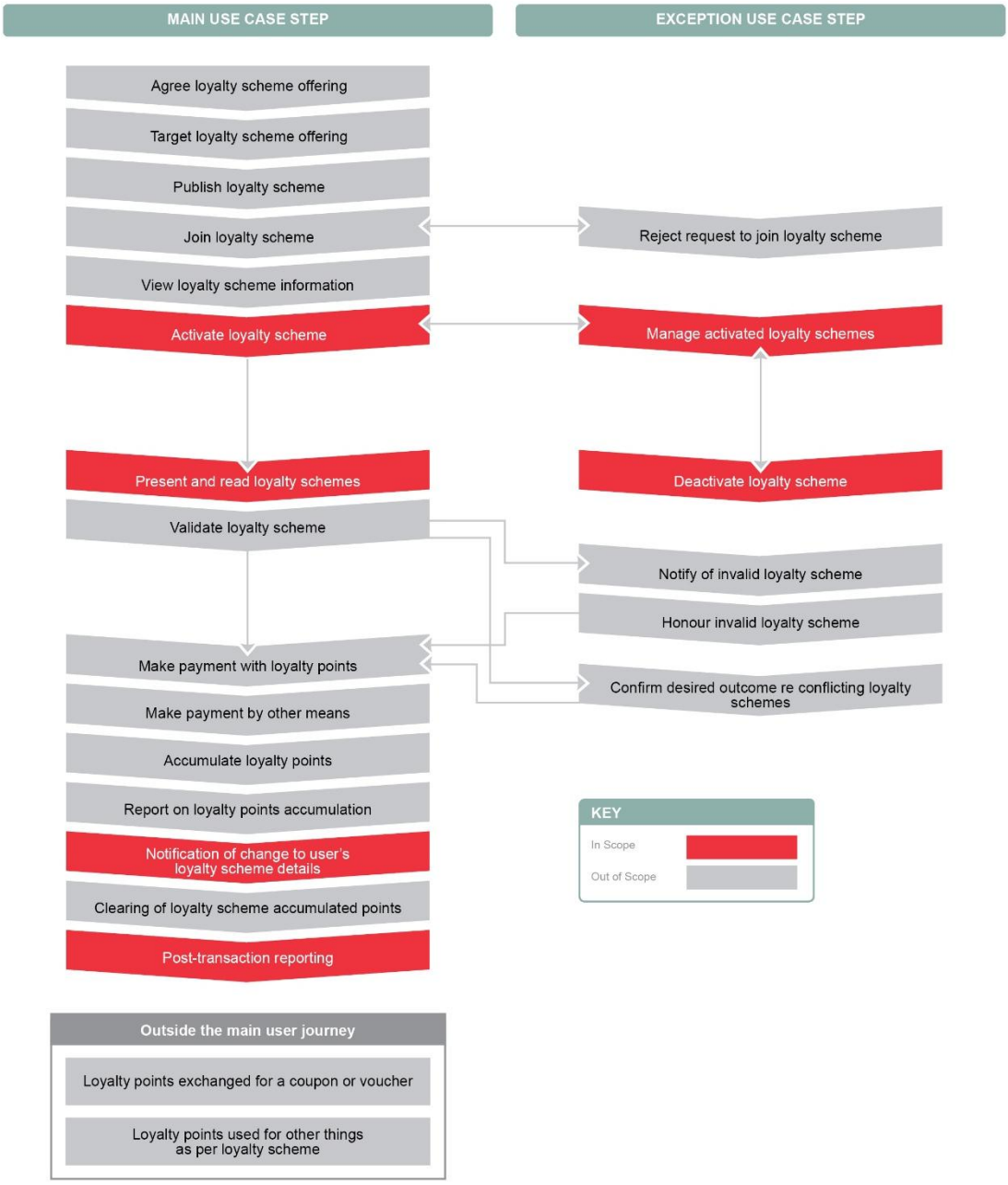


Figure 4**: Example loyalty use case steps**

These use case steps are described as follows:

| Loyalty Use Case Step | Precise Description | Scope |
|---|---|---|
| Agree loyalty scheme offering | The Loyalty Scheme Issuer agrees the initial loyalty scheme with one or more manufacturers/merchants. | Out of scope |
| Target loyalty scheme offering | The Loyalty Scheme Promoter uses data held regarding consumers to choose target recipients. | Out of scope |
| Publish loyalty scheme | The Loyalty Scheme Promoter promotes the loyalty scheme for access by potential Loyalty Scheme Users via the MNO wallet app and/or service provider app. | Out of scope |
| Join loyalty scheme | The Loyalty Scheme User submits or agrees to submit the necessary information to acquire a digital token for the loyalty scheme via the MNO wallet app or service provider app. | Out of scope |
| Reject request to join loyalty scheme | The Loyalty Scheme User request to join the loyalty scheme via the MNO wallet app or service provider app is rejected due to Loyalty Scheme rules. | Out of scope |
| View loyalty scheme Information | The Loyalty Scheme User views their loyalty scheme details via the MNO wallet app or service provider app. | Out of scope |
| Activate loyalty scheme | The Loyalty Scheme User marks the loyalty scheme to be automatically Presented to the next potential Loyalty Scheme Points Awarder via the MNO wallet app and/or service provider app. | In scope |
| Manage activated loyalty schemes | The Loyalty Scheme User is informed that this loyalty scheme cannot be activated without the deactivation of some other loyalty schemes and is presented with a list of currently activated loyalty schemes from which to choose loyalty scheme(s) to be deactivated, via the MNO wallet app and/or service provider app. | In scope exception |
| Deactivate loyalty scheme | The Loyalty Scheme User marks the loyalty scheme not to be automatically presented to the next potential loyalty scheme Points Awarder via the MNO wallet app and/or service provider app. | In scope exception |
| Present and Read loyalty schemes | The Loyalty Scheme User presents relevant activated loyalty scheme digital tokens to the potential Loyalty Scheme Points Awarder by physically presenting the mobile handset to an NFC reader. | In scope |
| Validate loyalty scheme | The Loyalty Scheme Points Awarder validates whether each loyalty scheme digital token is authentic and valid. | Out of scope |
| Notify of invalid loyalty scheme | The Loyalty Scheme Points Awarder notifies the Loyalty Scheme User of the loyalty scheme(s) failing validation. | Out of scope |
| Honour invalid loyalty scheme | The Loyalty Scheme Points Awarder awards the Loyalty Scheme User points to be accumulated to a valid loyalty scheme as good will. | Out of scope |
| Confirm desired outcome of conflicting loyalty schemes | The Loyalty Scheme Points Awarder requests the Offer User to clarify which conflicting loyalty schemes to use and which not to use. | Out of scope |

| Loyalty Use Case Step | Precise Description | Scope |
|---|---|---|
| Make payment with loyalty points | The Loyalty Scheme User pays some or all of the balance remaining, using available loyalty scheme points. | Out of scope |
| Make payment by other means | The Loyalty Scheme User pays the remaining balance remaining for their purchase, if any, by any means other than loyalty scheme points. | Out of scope |
| Accumulate loyalty points | The Loyalty Scheme Points Awarder awards relevant amount of points for the validated loyalty scheme(s) to the Loyalty Scheme User and records the points' accumulation information in his system. | Out of scope |
| Report on loyalty points accumulation | The Loyalty Scheme Points Awarder communicates the points' accumulation information to the Loyalty Scheme Issuer and Loyalty Scheme Promoter, to the agreed level of detail. | Out of scope |
| Notification of change to user's loyalty scheme details | The Loyalty Scheme User is automatically presented with or prompted to view their loyalty scheme details via the MNO Wallet App or service provider App. | In scope |
| Clearing of loyalty scheme accumulated points | The Loyalty Scheme Issuer checks the awarded loyalty scheme points and whether the rules were applied and together with the Loyalty Scheme Points Awarder agrees on the appropriate compensation. | Out of scope |
| Post-transaction reporting | The Wallet App Server records contextual information regarding this loyalty transaction and provides agreed information to Loyalty Scheme Promoter and/or Loyalty Scheme Issuer. | In scope |
| Loyalty points exchanged for a coupon | The Loyalty Scheme User requests the Loyalty Scheme Issuer to exchange available loyalty scheme points for one or more coupons. | Out of scope |
| Loyalty points used for other things as per loyalty scheme | The Loyalty Scheme User requests the Loyalty Scheme Issuer to use available loyalty scheme points for some other purpose as per the Loyalty Scheme offering. | Out of scope |

**Table 3: Loyalty use case – step descriptions**

These descriptions are based on the GS1 Digital Coupon Management Process [4] use case steps, where analogy to the longer term engagements of an extended loyalty relationship is appropriate, as applied to mobile NFC loyalty specifically.

### 2.3.2 Assumptions

Out of scope use case steps are enabled by other parties in the ecosystem, often by existing processes.

'Make payment by other means' refers to a payment made by any means acceptable for the merchant, excluding the use of loyalty points from this particular loyalty scheme, which is covered by 'Make payment with loyalty points'.

No loyalty point's balance is stored on the VAS applet.

Though it is highly likely that an implementation will offer the functionality to display a loyalty scheme points balance for a given scheme, ‘View Loyalty Scheme Information‘ is an out of scope use case for this proposal. The notification of an update to a loyalty balance, post-transaction, would be achieved by leveraging the notification of change to user's loyalty scheme details’ use case step.

Existing loyalty scheme processes are enabled by the passing of a loyalty ID at the ‘Present and Read Loyalty Schemes’ use case step.

## 2.4 Mixed coupon and loyalty user journeys

Coupon and loyalty user journeys, although outlined separately above, could be concurrent at key use case steps, as illustrated in Figure 5.

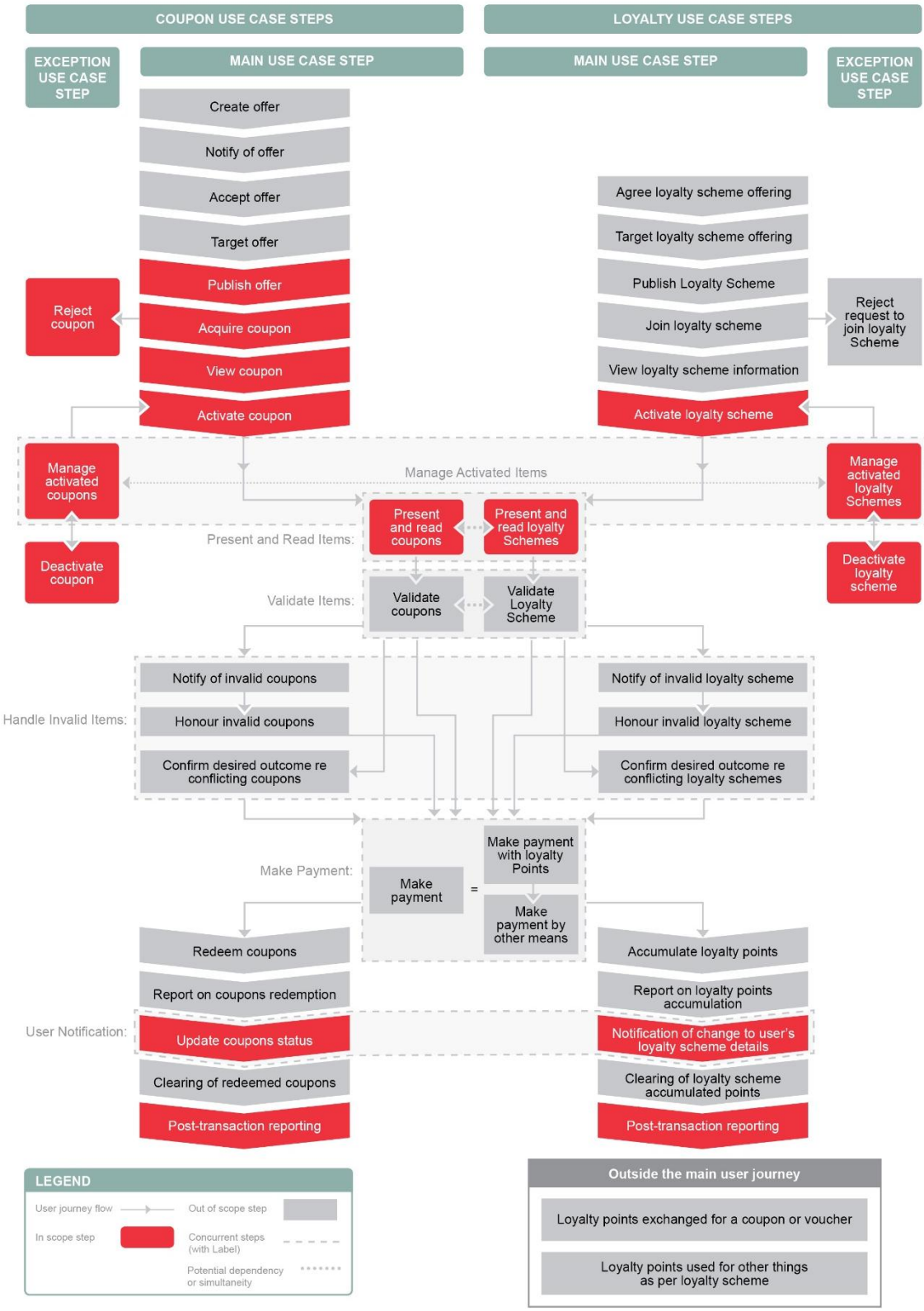


Figure 5**: Mixed coupon and loyalty use case steps**

## 2.5 Customer journey for coupons – process flow at the point of interaction (POI)

The below diagram reflects the customer journey and the associated actual data flow for each component. The data flow shown here covers two key use cases:

1. Passing the actual coupon instances from the app server to the handset and then to the contactless terminal on the tap.
2. Passing only a customer identifier to the handset and then on to the contactless terminal. ePOS will be responsible for gathering actual coupon instance information based on the customer identifier passed.

| | Receive Offer | Accept Offer/ Get Coupon | Activate Coupon | Scan Items | Tap Phone to redeem coupon | Coupon success notification |
|---|---|---|---|---|---|---|
| **Customer journey** | • Customer receives an offer on their handset.<br>• The offer they receive could be from a retailer, FMCG, co-brander or other third party. The coupon can come directly into their MNO wallet application, or into their retailer/loyalty/other third-party app | • They like the offer, and accept it. | • *Option 1*: Customer activates the relevant coupon they wish to redeem in that particular shop.<br>• *Option 2*: Using GeoLocation and other parameters, relevant coupons are selected and activated. | • Inside the store, after collecting all items, they go to the checkout and scan these items. | • Customer tap their phone to the contactless terminal.<br>• Their relevant coupons and vouchers are now transferred to the terminal. | • Customer receives coupon success/failure notification on phone.<br>• Other offers can also be sent at this time to the customer's phone. |
| **Actual data flow** | • Offer is sent to a handset by any of the backend app servers (could be MNO server or other third-party server). | • Use Case 1: A coupon instance ID is generated and sent to the handset by the relevant app server<br>• Use Case 2: A coupon instance is created and sent to the Retailer backend system. | • Use Case 1: Activated coupons are sent to the SIM to be used on next tap<br>• Use Case 2: Customer identifier for the said retailer/ merchant is passed to the SIM. | N/A | • Use Case 1: Coupon instance ID and other relevant information is passed from SIM to the reader [encrypted].<br>• Use Case 2: Customer Identifier is passed from SIM to the reader.<br>• Reader decrypts the information and passes it to ePOS for validation. | • Customer is notified by their provider's backend via a push notification or update when they have network.access. . |

Figure 6**: Acquiring and redeeming coupons**

# 3 Solution architecture

## 3.1 Architecture

The high level solution architecture that supports the retail use cases is shown in Figure 7.

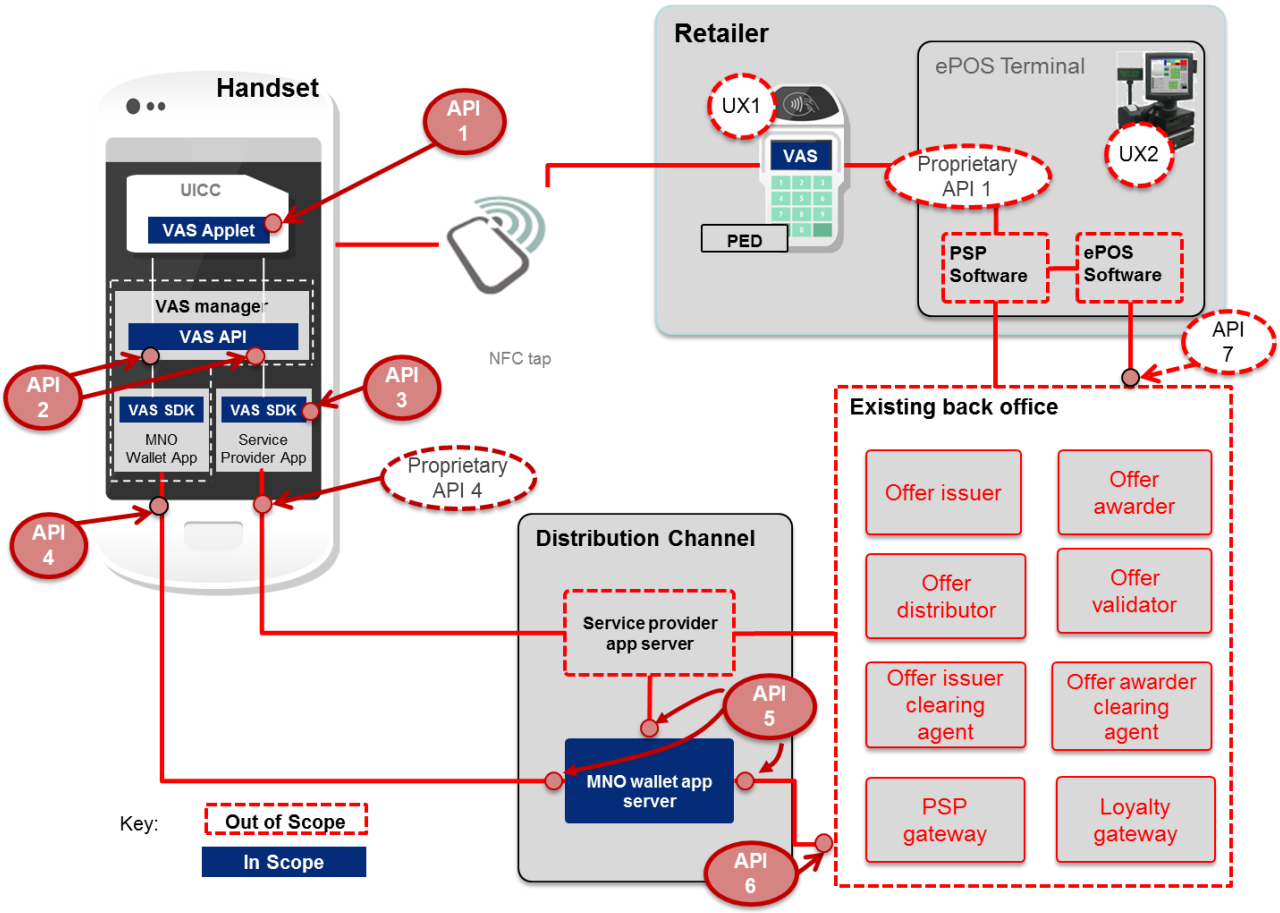


Figure 7**: High level solution architecture**

| API Name | Description | Section within the Document |
|---|---|---|
| API 1 | API used by the PED/Terminal to extract information from the value-added services applet. | Section 6.1 |
| API 2 | Application protocol data unit (APDU) Level API for access to the applet from within the handset. | Section 6.2 |
| API 3 | High Level API/software development kit (SDK) used by user interface (UI) apps to access the VAS applet. | Section 6.3 |
| API 4 | Backend app servers use this API to talk to their apps. | Section 6.4 |
| API 5 | Backend app server API used by apps and other back office processes. | Section 6.5 |
| API 6 | A common API to access various retailer (and wider ecosystem) back office from operator's app server. | Section 6.6 |
| Proprietary APIs | Proprietary APIs that can be used. | -- |

**Table 4: API descriptions**

The details of each component of this system, their role, ownership and responsibilities are defined further in Table 5.

| Component | Description | Ownership |
|---|---|---|
| Handset | Consumer mobile device supporting NFC card emulation capability via either a UICC or alternative secure element. | Consumer |
| UICC | A smart card that conforms to the specifications written and maintained by the ETSI Smart Card Platform project. | MNO |
| VAS applet | VAS applet hosted within a secure element environment and responsible for facilitating communication between user space VAS applications and PED (or other NFC readers) | MNO |
| VAS Manager (optional) | Service component hosted on the mobile device and responsible for presenting a simplified, functional, interface between the secure element VAS applet and user space applications that utilise the VAS applet. Capable of performing administration and co-ordination activities, local caching of data where UICC memory is limited and facilitating and arbitrating between multiple user space applications that could be contending for the VAS applet resources. This component is optional and it is to the discretion of individual MNOs to simplify the VAS applet access by implementing this service, or compel user space applications to access the UICC via operating system specific 7816-4 APDU (See [13]) commands.<br>The VAS Manager may be supplied as a standalone service or provided as part of an MNO wallet application at the discretion of the service providing MNO. | MNO |
| VAS SDK (API 3) | A simplified, functional interface presented by the VAS Manager. The API presented allows user space applications to access and manage VAS content hosted by MNO secure element applets. | MNO |
| MNO wallet app | A consumer facing wallet application published by individual MNOs. Contains functionality to support couponing, loyalty and other value added service capabilities defined throughout the remainder of this document. | MNO |
| Service provider app | A consumer facing mobile application published by service providers such as individual merchants or manufacturers. There may be many such applications installed on a consumer Mobile Device, each with its own merchant/manufacturer customised experience. All using the MNO VAS applet through the VAS SDK (API 3) or VAS API (API 2) to convey VAS data to a POS device. | Merchants, Manufacturers, Loyalty service |
| PED | PED supplied by various manufacturers, supporting ISO/IEC 14443 [11] based card emulation for UICC contactless VAS services. | Payment Service Provider |

| Component | Description | Ownership |
|---|---|---|
| ePOS terminal | Electronic POS terminal for facilitating transactions with the end customer. Typically interacts with a number of accessories including barcode scanners, PEDs, client facing displays etc. | Merchant and POS terminal vendor |
| PSP software module | Software component installed on the ePOS terminal by the payment service provider to facilitate payment service provider (PSP) payment and value-added services functionality. | PSP |
| ePOS software module | Implements the core functionality of the ePOS terminal. Typically supplied by the POS software vendor, this controls the workflows and accessory interaction, including interfacing to the PSP Software Module to facilitate payment and PSP enabled VAS. | POS terminal vendor |
| MNO wallet app server | A backend service implemented by the MNO supplying the mobile device wallet application. The wallet app server implements authentication, content management and interfacing with further backend systems, such as the loyalty service provider, to facilitate the consumer facing MNO wallet application experience. | MNO |
| Service provider app server | A backend service implemented by service provider entities to support the experiences implemented by consumer facing mobile applications. Services implement authentication, content management and interfacing with further backend systems, such as a loyalty service provider, to facilitate the consumer facing service provider application experience. | Manufacturer, Merchant |
| Offer issuer | Party issuing the coupons, bearing the commercial and financial responsibility for the coupons (as per GS1 digital coupon specification [4]). | Merchant, Manufacturer, Marketer, Loyalty service |
| Offer awarder | Party responsible for the redemption of the coupon (as per GS1 digital coupon specification [4]). | Merchant, Loyalty service |
| Offer distributor | Party responsible for the distribution of the coupons to the consumers on behalf of the Offer Issuer (as per GS1 digital coupon specification [4]). | MNO, Manufacturer, Merchant, Loyalty service |
| Offer validator | Party assisting the Offer Awarder in the validation of the coupons by offering a centralised service (as per GS1 digital coupon specification [4]). | POS Vendor, Merchant, PSP, Loyalty service |
| Offer issuer clearing agent | Party responsible for the financial clearance of redeemed coupons on behalf of the offer issuer (as per GS1 digital coupon specification [4]). | Manufacturer, Merchant, PSP, Loyalty Service, Other |
| Offer awarder clearing agent | Party responsible for the financial clearance of redeemed coupons on behalf of the offer awarder (as per GS1 digital coupon specification [4]). | Manufacturer, Merchant, PSP, Loyalty Service, Other |

| Component | Description | Ownership |
|---|---|---|
| PSP gateway | Payment service provider gateway to payment services. This may also act as a gateway to other services such as loyalty or anti-fraud services. | PSP |
| Loyalty gateway | Loyalty service provider gateway to loyalty service. | Loyalty service |
| UX 1 | Represents the user experience of phone/payment terminal interaction. Definition of this user experience is out of scope for this document. | MNO/Merchant |
| UX 2 | Represents the user experience provided by the POS terminal. Definition of this user experience is out of scope for this document. | POS Vendor, Merchant |
| API 1 | NFC value added service protocol for the bi-directional communication of VAS data between secure element applet and payment terminal.<br>Note: the protocol implemented by API 1 is built on the ISO 7816-4 APDU command set [13]. However, it should be possible to operate this protocol over various bearers; ISO 14443(1-4) [11], ISO 18092 (NFC-IP) [12], ISO 7816(1-3) [13], controlled by the user space applications (extracting stored data from the secure element) where appropriate. | 1. GSMA Standard<br>2. MNO (VAS applet)<br>3. Payment Terminal Vendors |
| API 2 | ISO 7816 APDU command interface [13] between consumer facing user space mobile application and the VAS applet operating in the secure element java card execution environment. | MNO |
| API 3 | API (and SDK) associated with the optional VAS manager. This may be a functional, Inter-process communication (IPC) or optionally dual interface with more abstract API than the 7816 APDU secure element services [13] that this abstracts.<br>The VAS manager may be either a simple integration library, for integration to MNO wallet applications, part of the MNO wallet application itself or a standalone operating system service. | MNO |
| API 4 | Interface to the MNO wallet for communication between the wallet and supporting MNO web service. | MNO |
| API 5 | Interface to the MNO wallet service for communication of VAS data from merchants or third-party retail/manufacturer/loyalty service provider services. This supports mechanisms to notify the wallet of updated data (issued coupons etc.) as well as other communication (updates, messaging, and account details). | MNO |
| API 6 | Interface to merchant/loyalty service provider systems that allow offers, coupons and loyalty scheme account administration. | Merchant/Loyalty service |

| Component | Description | Ownership |
|---|---|---|
| API 7 | Interface between the Retailer ePOS and the various backend systems to support online validation, coupons clearing and other processes. This document aims to briefly describe the necessary components to enable overall loyalty and couponing, the actual details and specifications will be proprietary to each implementation. This will be developed by the concerned parties, and not by the MNO. Details for API 7 are out of scope of this document. | Retailer ePOS, Backend system owners. |

**Table 5: System components**

## 3.2 Assumptions

### 3.2.1 UICC limitations

This proposal assumes that any system produced will be supported on current UICC technology. This section of this document describes some of the implications of limited UICC capabilities.

#### 3.2.1.1 Size limitations

UICC cards presently come in various memory sizes. Common among these are 64KB, 128 KB, 256 KB, or 512 KB. Most MNOs wishing to deploy a mobile wallet will choose a 256 KB UICC as a reasonable compromise between memory and cost.

A certain amount of space is normally reserved for an operator's network access application on the UICC and this applet could consume a significant amount of available memory.

Assuming that roughly 128 KB of a 256 KB UICC is used for the operator's network access application, about 128 KB of space will be left for the mobile wallet applets, which are expected to include both payment and VAS applets.

Typically, payment applets are around 50 to 60 KB in size. Assuming that only 100 KB of memory is reserved and that two payment applets (e.g. MasterCard and Visa) are installed on the UICC, this would leave around 28 KB for the VAS applet for loyalty and couponing (256 KB – 128 KB – 100 KB).

Assuming that a simple VAS applet only needs about 20 KB of code space, this would leave about 8 KB for loyalty IDs and coupon data.

The maximum APDU size is normally 255 bytes, although for device compatibility reasons, 240 bytes should be used. As long APDU support is not generally available, long data structures will need to be split into 240 byte chunks and there is protocol overhead for re-assembling the data structures.

#### 3.2.1.2 Speed limitations

Specified speeds for the NFC interface are 424 kbit/s, 212, kbit/s, or 106 kbit/s. This means that there are cards in use that will only support 106 kbit/s and this is seen as the default data transfer rate for EMV (Europay MasterCard Visa) transactions.

***Note***: Asymmetric cryptography (for encryption) would add a time overhead. These operations can range from 500 ms to 2000 ms to perform. As many service providers have a guideline of a 600 ms tap, the high end of the timing would not be feasible.

### 3.2.2 Proposed architecture limitations

Assuming that GS1 standards [4] are used for the coupon identifier, the standard GS1 coupon has 12 digits which needs 6 bytes in binary coded decimal (BCD) encoding. The GS1 and extended GS1-128 barcode together have 26 digits, requiring 13 bytes in BCD encoding. Assuming some additional encoding overhead and information requirement (but not anticipated here) it is assumed that 20 bytes are sufficient to store a coupon.

#### 3.2.2.1 Size assumptions

Given the assumption that 8 KB of storage are available for coupons and loyalty IDs and an assumed maximum size of 20 bytes per coupon or ID, 400 coupons/loyalty IDs/retail IDs can be stored simultaneously in the VAS applet.

#### 3.2.2.2 Speed assumptions

The POS protocol call flow for coupon redemption consists of the below requests/messages, where the `getData` APDU exchange could occur several times:

| Request/Message | Typical Time | Description |
|---|---|---|
| Polling | 30 ms | Looking for an NFC compatible handset |
| `select(VASAppAID)` | 20 ms | Selecting the VAS applet from the handset |
| `putData(MerchantID)` | 30 ms | Sending the Merchant ID to the VAS applet. This will actually be a custom command puVASData (See 6.1.4.3) |
| Security | 300-500 ms | Authenticating and securing the channel before data transmission |
| `getData` | 30 ms per APDU (in 106 kbit/s) | Transfer for coupon/loyalty information |

**Table 6: Typical timing of requests and messages**

Assuming three coupons/Loyalty IDs per `getData` APDU exchange, it is expected that up to thirty coupons/Loyalty IDs will be transferred within 300ms, excluding security.

### 3.2.3 Other assumptions

1. The system specified will interface with existing, physical loyalty card schemes.
2. Use of a trusted service manager (TSM) beyond the VAS applet provisioning is optional.
3. Mobile device data connectivity will not be necessary to redeem.
4. The system specified will allow any service provider an access to store coupons and loyalty information on the UICC.
5. Terminology used in the GS1 digital coupon management document [4] is recognised by the retail industry.
6. There is only one VAS applet on the UICC.
7. There may or may-not be a MNO wallet application on a given handset.
8. There will be no impact on existing payment system directives and licenses, e.g. PCI-DSS or terminal certification or eMoney License.
9. There may or may not be multiple service provider applications on a given handset.

### 3.2.4 System actors

We have based our work on the existing 'GS1 digital coupons standard specifications' [4], which specified interoperable messaging between various actors for successful validation and redemption of coupons, offers and other loyalty information. The various actors used throughout this document are described in Figure 8, with their potential role as per the GS1 specification.

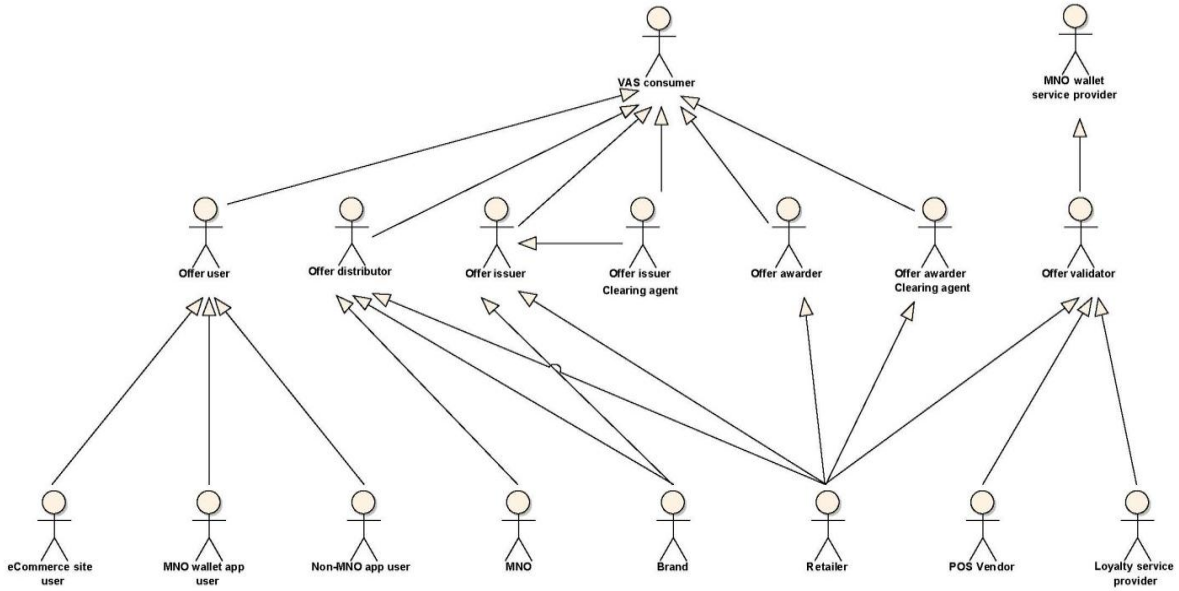


Figure 8**: Actors within the GS1 digital coupon specification [4]**

# 4 Data and control flows

In this section, we discuss the various data objects that can be used throughout the end-to-end solution. We start with a generic `ObjectInfo`, and extend it to `VASObjectInfo` and related objects and structures. Chapter 4 will also explore various control flows for the actual coupon/loyalty data to pass through the system.

Implementation of the APIs, data and control flows detailed in this proposal affect multiple execution environments (see the architecture components outlined previously in section 3 Solution architecture). Those implementing elements of this proposal on their respective architecture elements should take care to avoid namespace pollution.

## 4.1 VAS object complete data structure

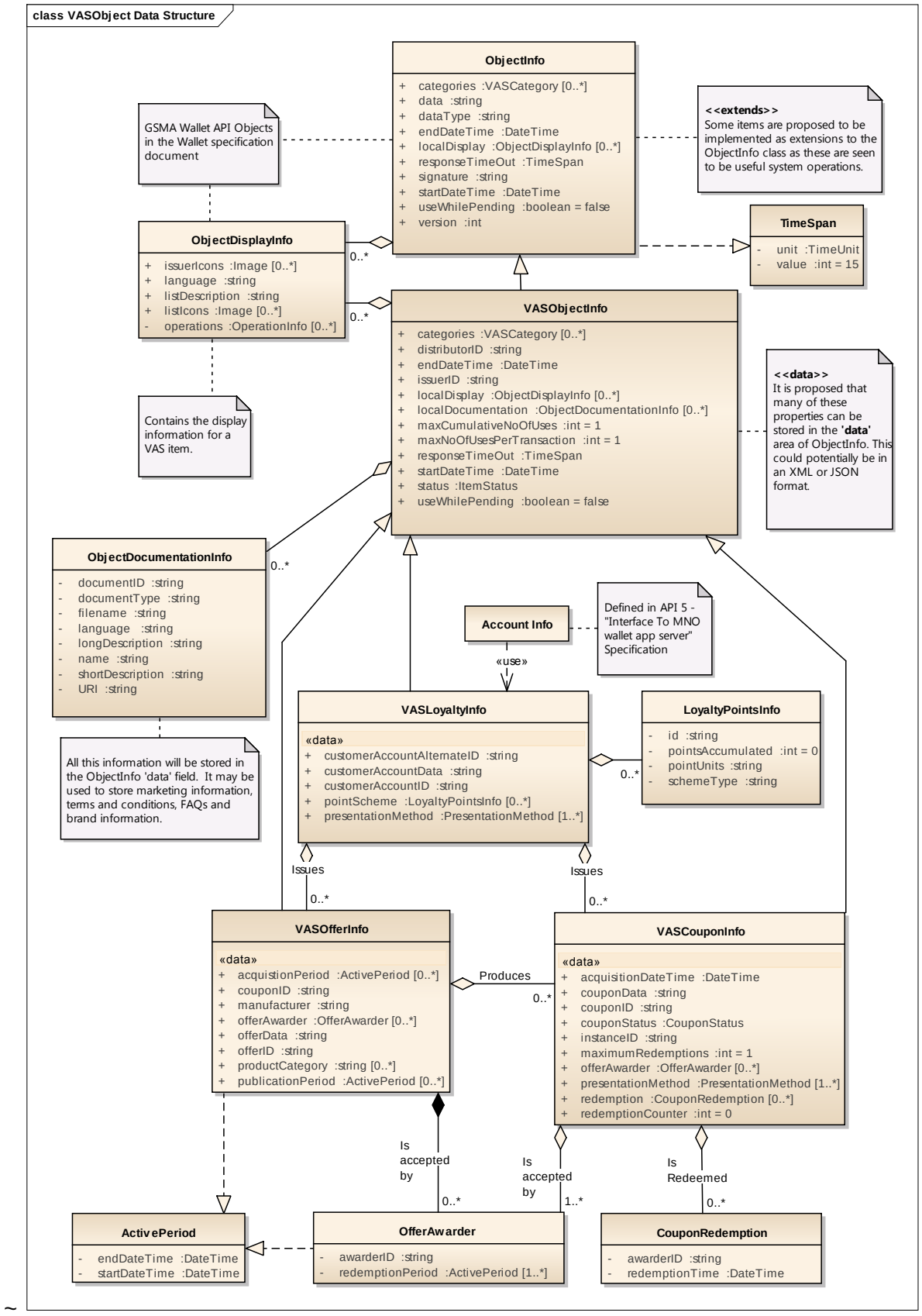


Figure 9**: VAS object data structure**

As seen in Figure 10, `ObjectInfo` class includes a reference to an `ObjectDisplayInfo` class that is used to manipulate visual information for a wallet item. Various value added services related classes and enumerations are derived from these two and available to use throughout the system.

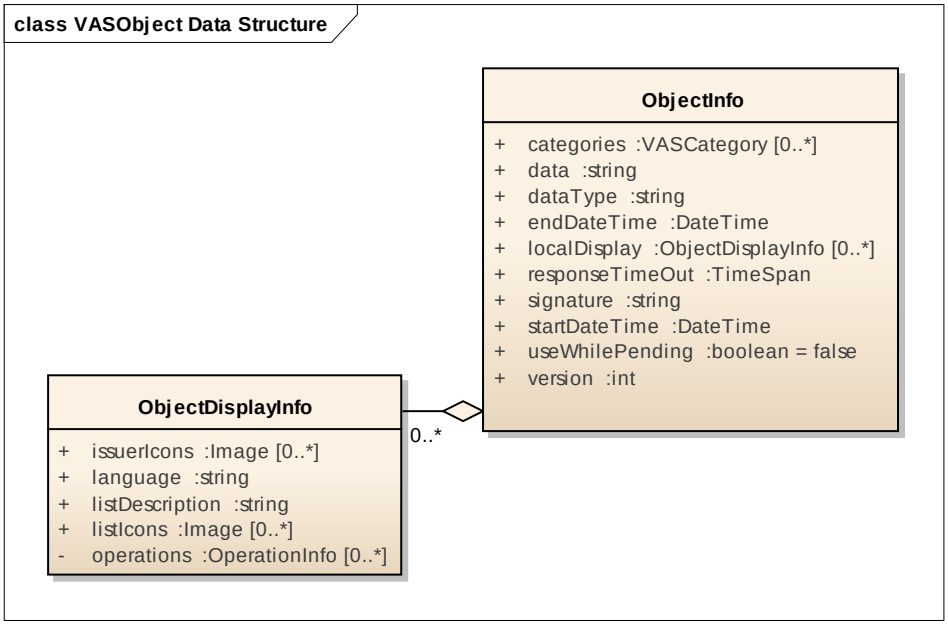


Figure 10**:** `ObjectInfo` **can contain a number of** `ObjectDisplayInfo` **objects**

The data structure for this proposal will assume an implementation of this wallet specification or similar. This architecture allows all the main data necessary for offers, coupons and loyalty to be passed through the system as a `data` property that is a primitive type of string. Individual parameters may then be defined within the data property using JavaScript Object Notation (JSON).

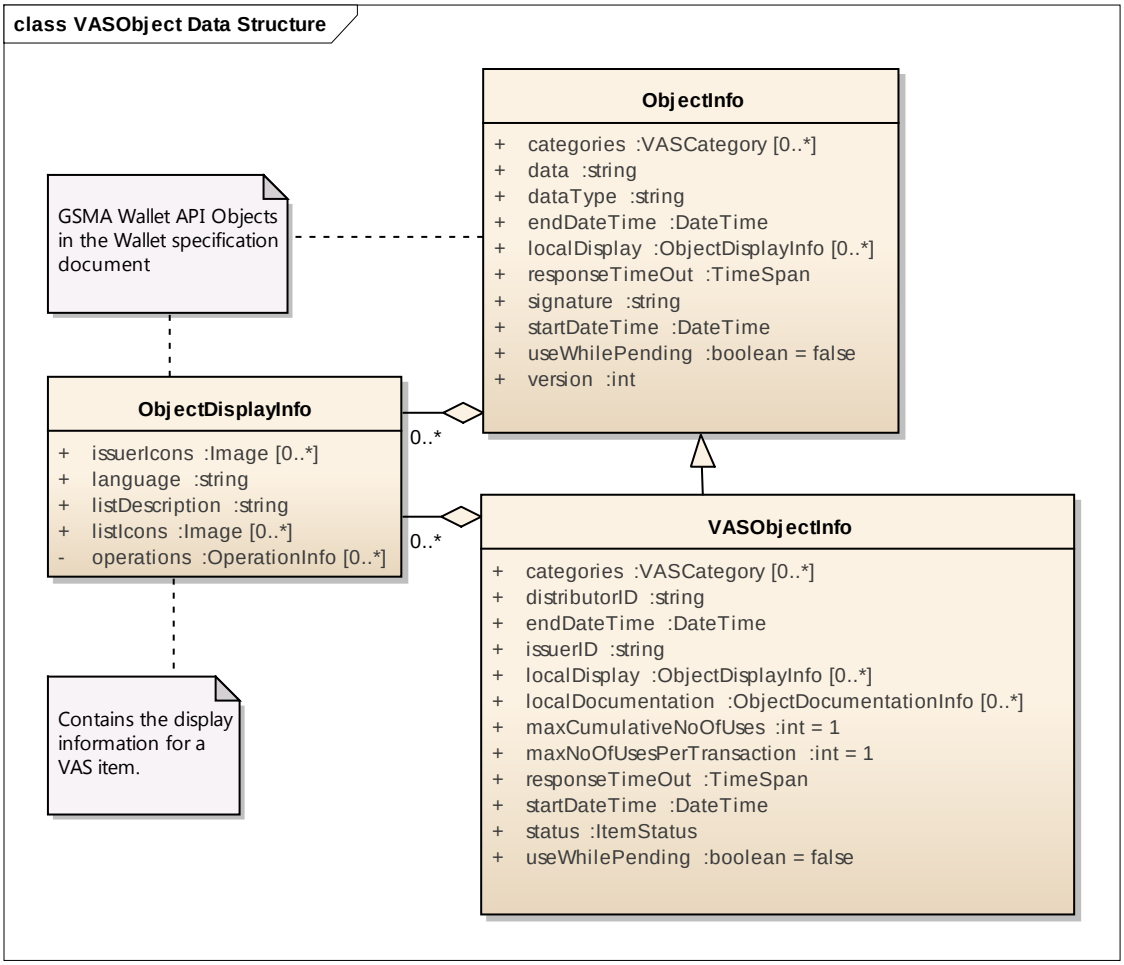


Figure 11: `VASObjectInfo` **is an instance of** `ObjectInfo`

One of the most important properties in this class is the `data` property as it will allow the wallet to be extended with key information for offers, coupons and loyalty.

`VASObjectInfo` is a generic VAS object, inherited from the wallet specifications generic item `ObjectInfo`. All public properties and functions available to the `ObjectInfo` class can also be in manipulated using the `VASObjectInfo` class.

`VASObjectInfo` is a common ancestor for the three VAS objects needed in the system, `VASOfferInfo`, `VASCouponInfo` and `VASLoyaltyInfo`.

This structure also considers:

- A loyalty scheme can issue offers or coupons
- Coupons can be produced from an offer
- A coupon or offer can be associated with a loyalty scheme

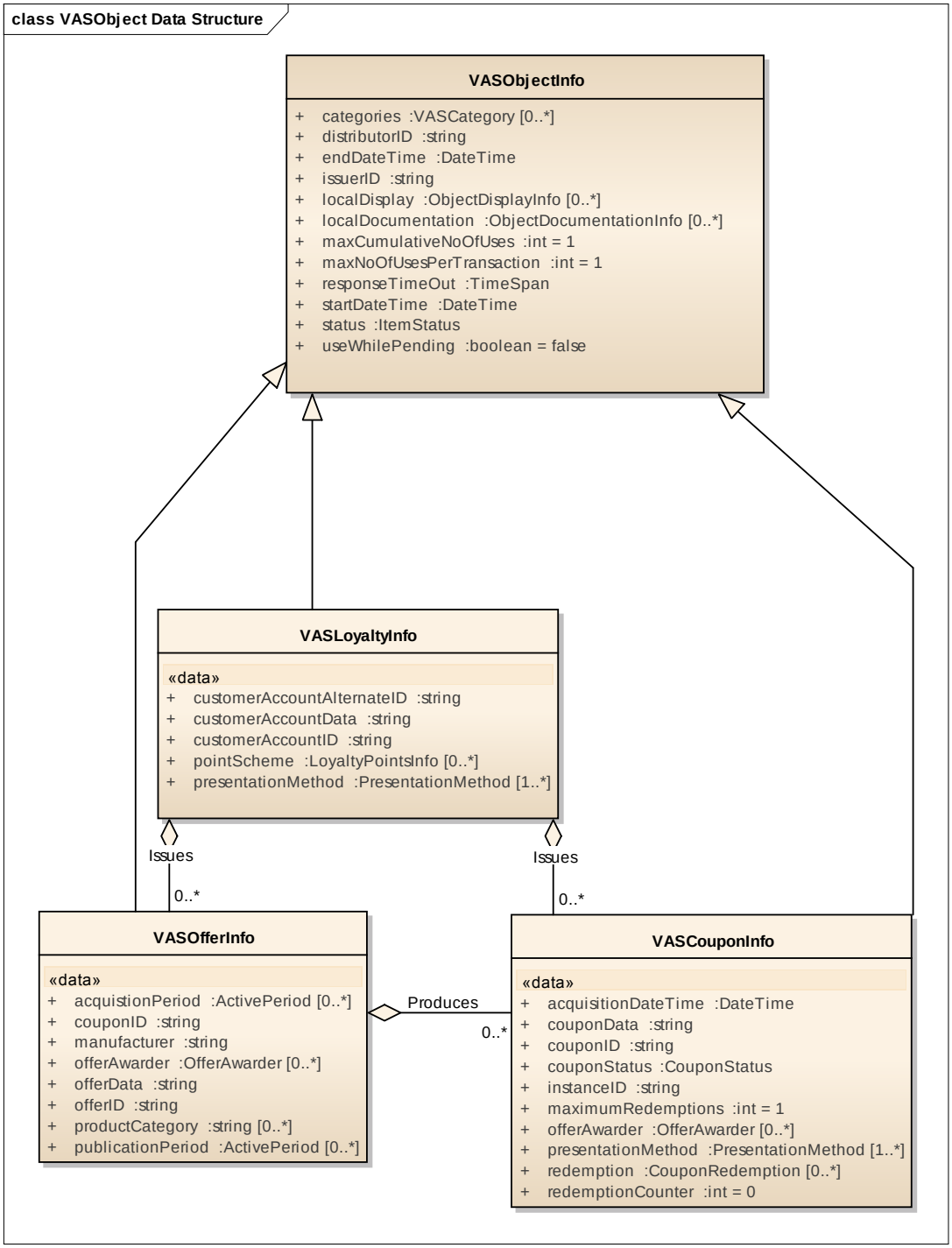


Figure 12**: The coupon, loyalty and offer objects are all instances of the VAS object**

The extended properties for the `VASObjectInfo` class will allow the wallet app to apply case-by-case rules for what to do once a wallet item is used and confirmation from the app server is not available.

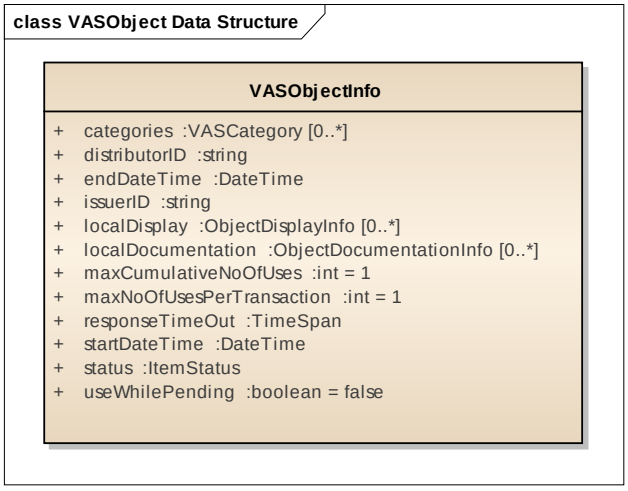


Figure 13**: The `VASObjectInfo` object**

The `responseTimeOut` value will allow an object to have a value applied for the time period that goes by without a "used confirmation", before it can be assumed that the wallet item was rejected and can be reclaimed. This is of type `TimeSpan`, a custom class with `TimeUnit` enumeration and a value. The `useWhilePending` flag allows an object to be set for re-use while waiting for a response or not.

`ObjectInfo`'s native `localDisplayObjectInfo` structure is fairly flexible for displaying a VAS object in an app. API 3 (section 6.3 of this document) includes an extra set of data for storing references to documentation that relates to a wallet item - `ObjectDocumentationInfo`.

`ObjectDocumentationInfo` allows extra pages of information to be associated with a wallet item. Instances of documentation may include 'terms and conditions' and 'manufacturer information'.

There are various enumerations used within the overall structure. These are shown in Figure 14.

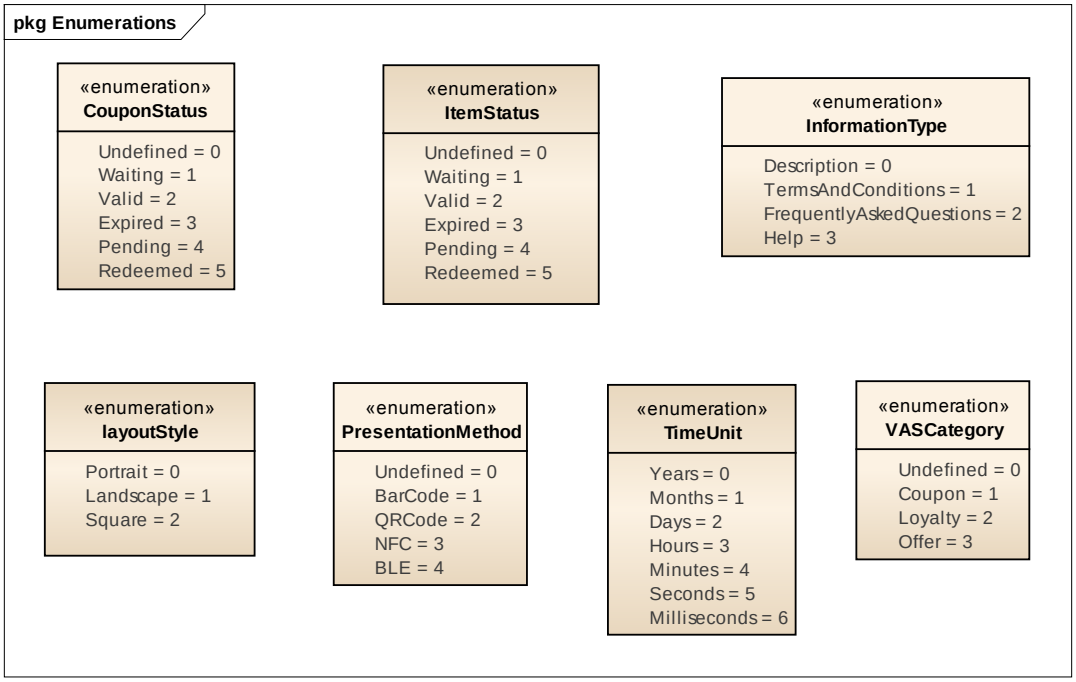


Figure 14**: Enumerations used within `VASObject` data structure**

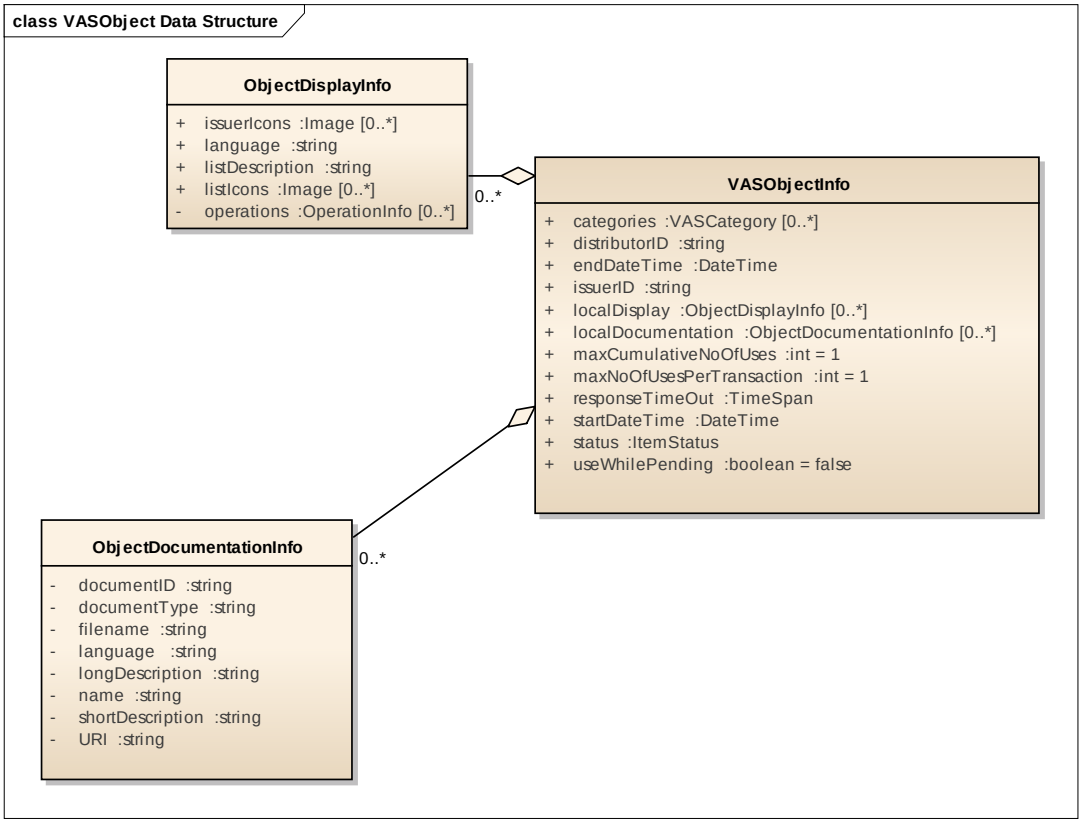


Figure 15**: `VASObjectInfo` may have a number of `localDisplay` objects and `localDocumentation` objects associated with it**

`VASOfferInfo` contains the data needed for offers in the wallet.

A `VASOfferInfo` can have a number of publication periods (`publicationPeriod`, representing periods when the offer can be seen) and acquisition periods (`acquisitionPeriod`, representing periods when an offer can be turned into a coupon that can later be redeemed). `publicationPeriod`, `acquisitionPeriod` and `redemptionPeriod` are all derived from an `ActivePeriod` data type that consists of a start time and an end time. The time periods in this proposal are based upon the coupon redemption model defined in the GS1 Digital Coupon Management specification [4].

A `VASOfferInfo` may also have a number of offer awarders (`offerAwarder`, representing organisations who will accept a coupon produced by the offer).

Coupons that can be produced by the offer are then associated with the relevant offer awarders.

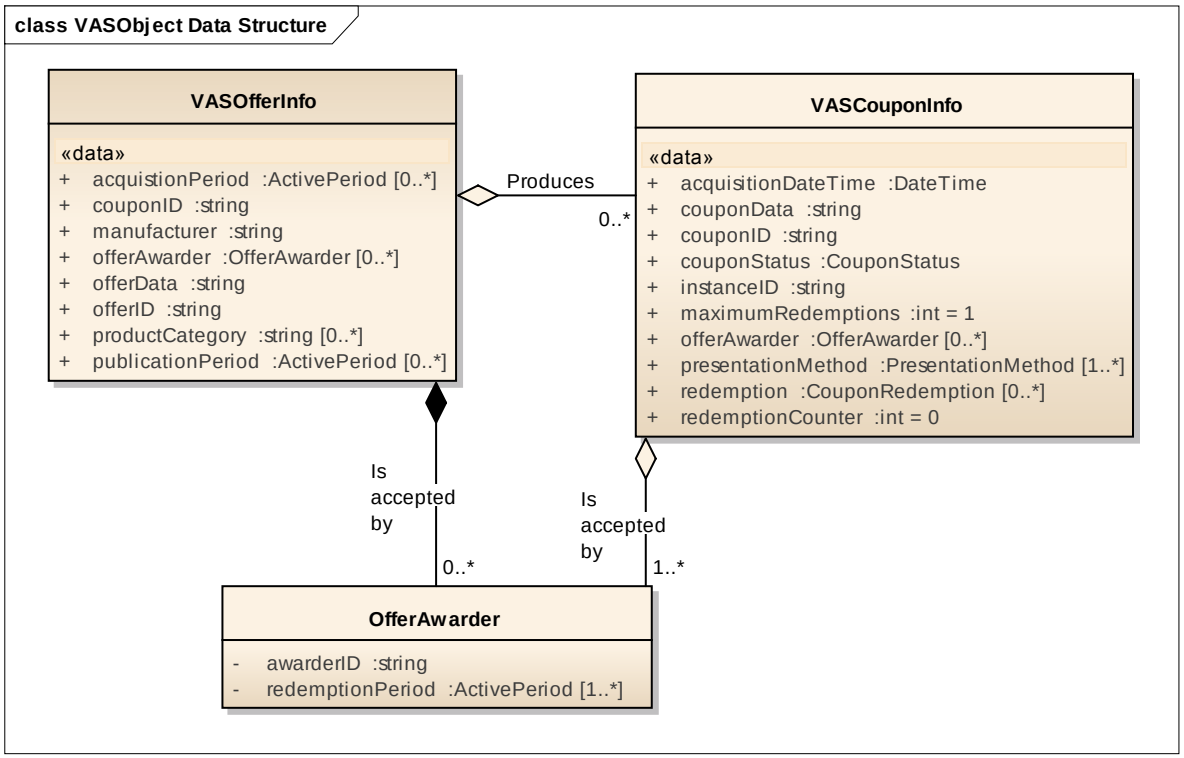


Figure 16**: The VAS offer may produce a number of coupons that can be redeemed at a number of offer awarders.**

`VASCouponInfo` controls the data for a coupon.

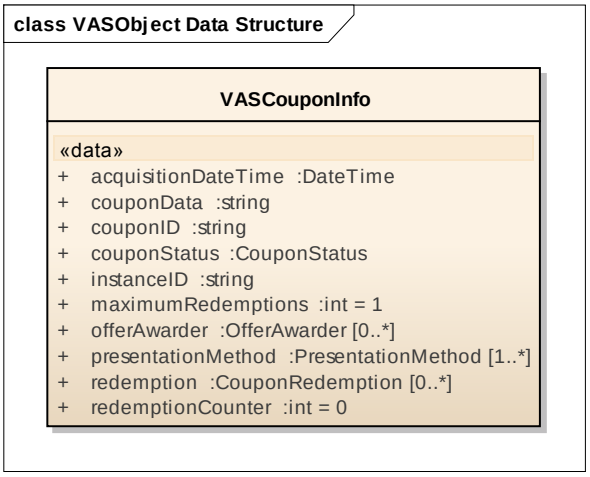


Figure 17**: The VAS coupon information object**

A coupon is associated with one or more offer awarders (organisations that will accept the coupon in exchange for goods, services or reward).

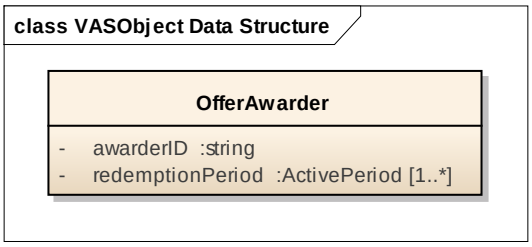


Figure 18**:** `OfferAwarder` **object**

Each offer awarder has a number of periods that the coupon is available for.

Once a coupon is redeemed, the awarder ID and redemption time should be stored as an instance of `CouponRedemption`.

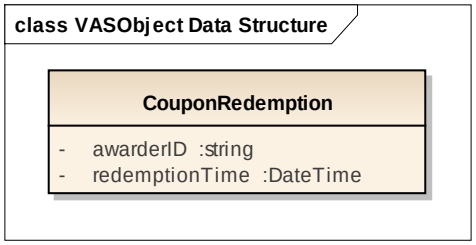


Figure 19**: A coupon redemption object that is used to store information about who accepted a coupon and time of acceptance**

`VASCouponInfo` includes a redemption counter that will allow the coupon to be used a number of times before the maximum number of redemptions is reached.

Keeping a redemption count on the mobile device allows for multiple uses without the need for feedback from the backend system.

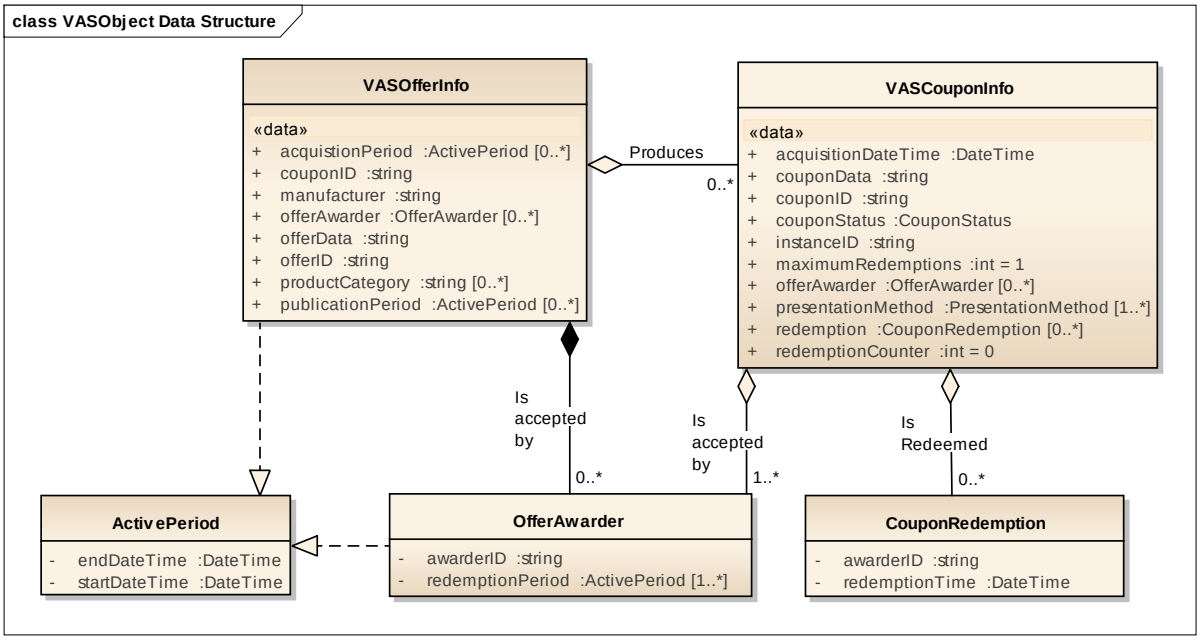


Figure 20**: Relationships between offers, coupons, awarders and coupon redemption**

`VASLoyaltyInfo` contains the information about a loyalty scheme

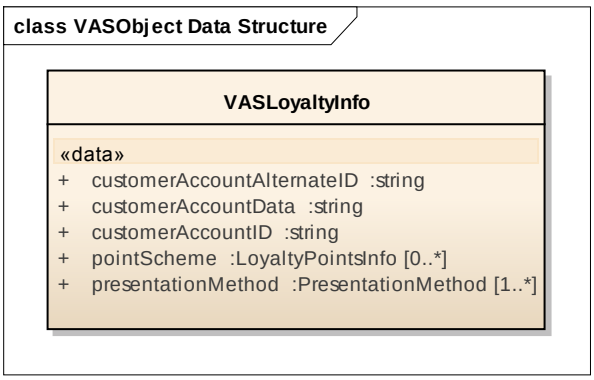


Figure 21**: VAS loyalty information object**

The key part of `VASLoyaltyInfo` is a customer account ID, but `VASLoyaltyInfo` also allows multiple point schemes to be associated with it.

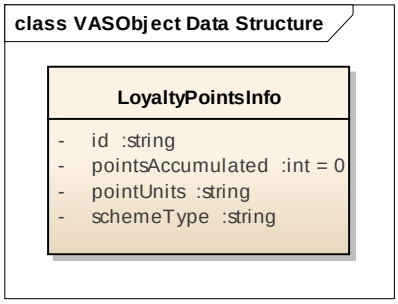


Figure 22**: VAS loyalty points information object**

The `LoyaltyPointsInfo` class includes the offline view of how many points are available, a unit for the number (such as points, stars and stamps), an ID used for the point scheme, and a scheme type (implementation specific).

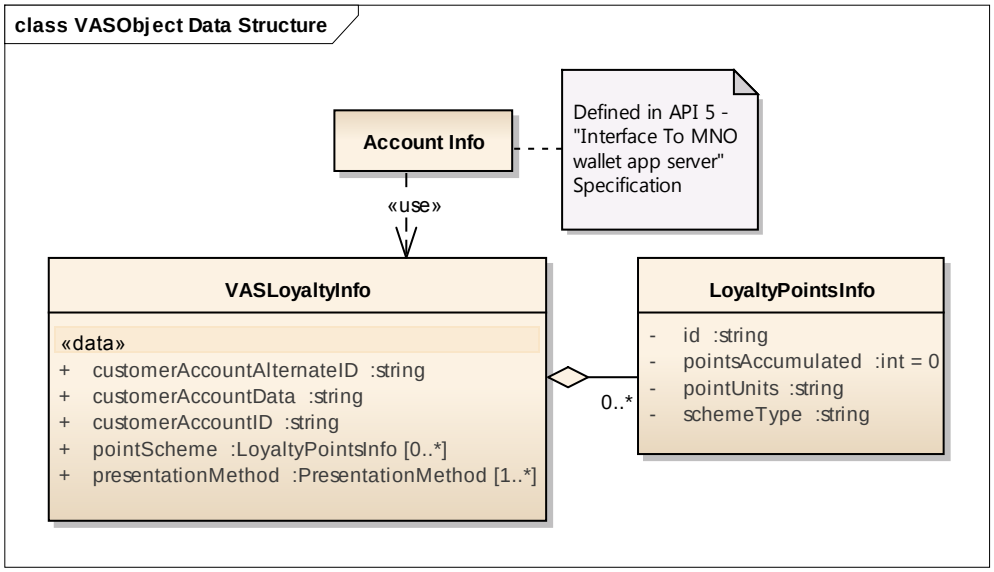


Figure 23**: Relationships between loyalty information, accounts and point schemes**

## 4.2 Data attributes

The tables used in this section show attributes, their type, the number of instances, a description, the default value and which of the application programming interfaces (APIs) will have a view of the attribute.

### 4.2.1 Common properties between `ObjectInfo` and `VASObjectInfo`

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `categories` | `VASCategory` | 0..* | List of categories that the VAS item is associated with. It will be setup at the time of object creation and is not expected to be changed through the object's lifetime. See Enumerations (Figure 14) for more detail | – | - | y | y | y | - | - |
| `dataType` | `string` | 1 | `"JSON"`, `"XML"`, … | `"JSON"` | y | y | y | y | - | - |
| `startDateTime` | `DateTime` | 1 | The date that the item will start to be active. If this is blank, then it will be assumed that the item is valid immediately. The date format should follow ISO 8601 [10]. | `01-01-1970 00:00:00` | y | y | y | y | - | - |
| `endDateTime` | `DateTime` | 1 | The date that the item will expire. If this is blank, then it will be assumed that the item will not expire. The date format should follow ISO 8601 [10]. | `01-01-1970 00:00:00` | y | y | y | y | - | - |

| Attribute | Type | Cardi | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| localDisplay | ObjectDisplayInfo | 0..* | Reference to one or more DisplayInfo objects that are used to display the offer in a list. | - | y | y | y | y | - | - |
| responseTimeOut | TimeSpan | 1 | The count of how long to wait until a response or it will be assumed that a response will not come back. | – | y | y | y | y | - | - |
| useWhilePending | boolean | 1 | This property indicates that the VAS item can still be used during the time period between a previous transaction and confirmation of that transaction. | false | y | y | y | y | - | - |
| Data | string | 1 | The data that is used to define the individual parameters required for the VAS items. This will consist of mandatory fields and any platform specific data. | "" | y | y | y | y | | |

**Table 7: The common properties that belong to each VAS information object (loyalty, offer and coupon)**

### 4.2.2 Attributes contained within the data property for `VASObjectInfo`

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `distributorID` | `string` | 1 | Contains reference of the organisation that distributed the VAS item. (GS1 digital coupon management specification actor [4]). The MNO wallet app server must ensure this ID is unique to each distributor across the platform. | "" | - | y | y | y | - | - |
| `issuerID` | `string` | 1 | Contains reference of the organisation that issued the VAS item. (GS1 digital coupon management specification actor [4]). The MNO wallet app server must ensure this ID is unique to each issuer across the platform. | "" | y | y | y | y | - | - |
| `localDocumentation` | `objectDocumentationInfo` | 0..* | This is a list of all the instances of documentation that are associated with the VAS item. These could be images, text documents, or references to web pages. | - | y | y | y | y | - | - |

| Attribute | Type | Cardi | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| `maxCumulativeNoOfUses` | `int` | 1 | This is the maximum number of uses that the VAS items can be applied to. | `1` | y | y | y | y | - | - |
| `maxNoOfUsesPerTransaction` | `int` | 1 | This is the maximum number of times that the VAS item can be applied to a single transaction. | `1` | y | y | y | y | - | - |
| `Status` | `ItemStatus` | 1 | This is the state of the item within the system. See Enumerations (Figure 14). | `Undefined = 0` | y | y | y | y | - | - |

**Table 8: Common attributes contained within the `data` property of all VAS objects (loyalty, offers and coupons)**

### 4.2.3 Properties of `VASLoyaltyInfo`

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `customerAccountAlternateID` | `string` | 1 | Any alternate ID that a Service Provider may wish to use. | "" | y | y | y | y | - | - |
| `customerAccountData` | `string` | 1 | Any other customer account information deemed useful by the service provider. | "" | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| customerAccountID | string | 1 | An identifier for the loyalty account belonging to the user. The MNO wallet app server must ensure this ID is unique to each loyalty card product across the platform. | "" | y | y | y | y | y | y |
| pointScheme | LoyaltyPointsInfo | 0..* | Optional service provider input to specify the exact loyalty type. | - | y | y | y | y | - | - |
| presentationMethod | PresentationMethod | 1..* | The presentation method list defines all the ways that a loyalty identity can be presented to the user.<br>See Enumerations (Figure 14) for more detail | - | y | y | y | y | - | - |

**Table 9: VAS loyalty information**

### 4.2.4 Properties of LoyaltyPointsInfo

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| Id | string | 1 | An identifier to be used to identify the points tier in the system. This is expected to be proprietary for each scheme. | "" | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `pointsAccumulated` | `int` | 1 | The actual count for the number of points, stars, stamps etc. that have been awarded under the tier of the scheme. | `0` | y | y | y | y | - | - |
| `pointUnits` | `string` | 1 | A string value that will be displayed to the user to represent the value of the points awarded. | "" | y | y | y | y | - | - |
| `schemeType` | `string` | 1 | This is a proprietary setting that can be used to distinguish between different types of point tiers within an application. | "" | y | y | y | y | - | - |

**Table 10: VAS loyalty points information**

### 4.2.5 Properties of `VASOfferInfo`

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `acquistionPeriod` | `ActivePeriod` | 0..* | This is a list of time periods during which the offer can be converted into a coupon. | - | y | y | y | y | - | - |
| `manufacturer` | `string` | 1 | This attribute can be used to identify a manufacturer that is associated with the offer. | "" | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| couponID | string | 1 | A generic code that is used as a root to a specific code for a coupon. A separate unique identifier may be produced when a coupon is produced from the offer. | "" | y | y | y | y | - | - |
| offerAwarder | OfferAwarder | 0..* | This attribute references a list of all the offer awarders that will accept coupons generated from the offer. | - | y | y | y | y | - | - |
| offerData | string | 1 | This can be used as a channel to send proprietary information through the system. It is recommended that this field is populated with a string of variables and values in JSON format. | "" | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `offerID` | `string` | 1 | This is used to identify the offer through the system. It is recommended that GS1 references [4] are used for new systems that do not already have proprietary rules for ID formats.<br>The MNO wallet app server must ensure this ID is unique to each offer across the platform. | "" | y | y | y | y | - | - |
| `productCategory` | `string` | 0..* | This is a list of product categories that the offer may be associated with. | - | y | y | y | y | - | - |
| `publicationPeriod` | `ActivePeriod` | 0..* | This is a list of time periods during which the offer can be displayed to a user. | - | y | y | y | y | - | - |

**Table 11: VAS offer information**

### 4.2.6 Properties of `OfferAwarder`

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| awarderID | string | 1 | This attribute identifies that merchant or service provider who will accept the coupon in exchange for goods, services or reward.<br>The MNO wallet app server must ensure this ID is unique to each offer awarder across the platform. | "" | y | y | y | y | - | - |
| redemptionPeriod | ActivePeriod | 1..* | This is a list of time periods when the coupon can be redeemed at the associated awarder specified in the awarderID attribute. | "" | y | y | y | y | - | - |

**Table 12: Offer awarder – The organisation who will accept an offer during specified periods in time that the offer can be redeemed**

### 4.2.7 Properties of ActivePeriod

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| endDateTime | DateTime | 1 | This is the date and time for the end of the time period.<br>The format for this entry should follow ISO 8601 [10]. | 01/01/1970 00:00:00 | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `startDateTime` | `DateTime` | 1 | This is the date and time for the start of the time period.<br>The format for this entry should follow ISO 8601 [10] | `01/01/1970 00:00:00` | y | y | y | y | - | - |

**Table 13: The start date and time, and the end date and time for a specific period**

### 4.2.8 Properties of `VASCouponInfo`

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `acquisitionDateTime` | `DateTime` | 1 | This will be used to hold the date and time that the coupon was created from an offer. The format for this entry should follow ISO 8601 [10]. | `01/01/1970 00:00:00` | y | y | y | y | - | - |
| `couponData` | `string` | 1 | This will hold any proprietary coupon information required by a platform. | `""` | y | y | y | y | y | y |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| couponID | string | 1 | The coupon ID is a reference that identifies the coupon in a backend system. It is recommended that this value uses GS1 notation [4] if a format for this number is not already defined in the backend system for the coupon.<br>The MNO wallet app server must ensure this ID is unique to each coupon across the platform (though not necessarily each instance). | "" | y | y | y | y | y | y |
| couponStatus | CouponStatus | 1 | An enumerated type that defines a state of the coupon. This can be used to display the coupon differently to the user during the lifecycle of the coupon.<br>See Enumerations (Figure 14) for more detail | Undefined = 0 | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| instanceID | string | 1 | A unique value that is assigned to the instance of the coupon. This can be used to track the coupon that has been assigned to an individual. If multiple instances of a coupon are assigned to an individual, then that individual will receive multiple coupons with different instance ID values.<br>If an instance coupon can be used multiple times (maximum redemptions > 1), then the instance ID will appear in the backend system a number of times.<br>The MNO wallet app server must ensure this ID is unique to each coupon instance across the platform. | "" | y | y | - | - | - | - |
| maximum Redemptions | int | 1 | The maximum redemptions attribute defines how many times an instance of a coupon can be used for transactions. | 1 | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| offerAwarder | OfferAwarder | 0..* | The offer awarder references a list of organisations that will exchange a coupon generated by the offer for goods services or reward, during a specific period of time (or a collection of time periods). | - | y | y | y | y | - | - |
| presentationMethod | PresentationMethod | 1..* | The presentation method list defines all the ways that a coupon can be presented to the user.<br>See Enumerations (Figure 14) for more detail | - | y | y | y | y | - | - |
| redemption | Coupon Redemption | 0..* | This attribute references instances of a coupon being redeemed for goods, services or reward. The coupon redemption data structure includes a merchant identifier and a date time stamp. | - | y | y | y | y | - | - |

| Attribute | Type | Cardinality | Description | Default | Available to API | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 6 | 5 | 4 | 3 | 2 | 1 |
| `redemptionCounter` | `int` | 1 | The redemption counter allows for an offline count of how many times the coupon has been used. This will allow a coupon to be used a number of times while waiting for confirmation that each redemption has been successful or not. | `0` | y | y | y | y | - | - |

**Table 14: VAS coupon information.**

## 4.3 Coupon redemption data flow (consumer acquires and redeems coupon)

The coupon messaging journey through the architecture is described in Figure 24.

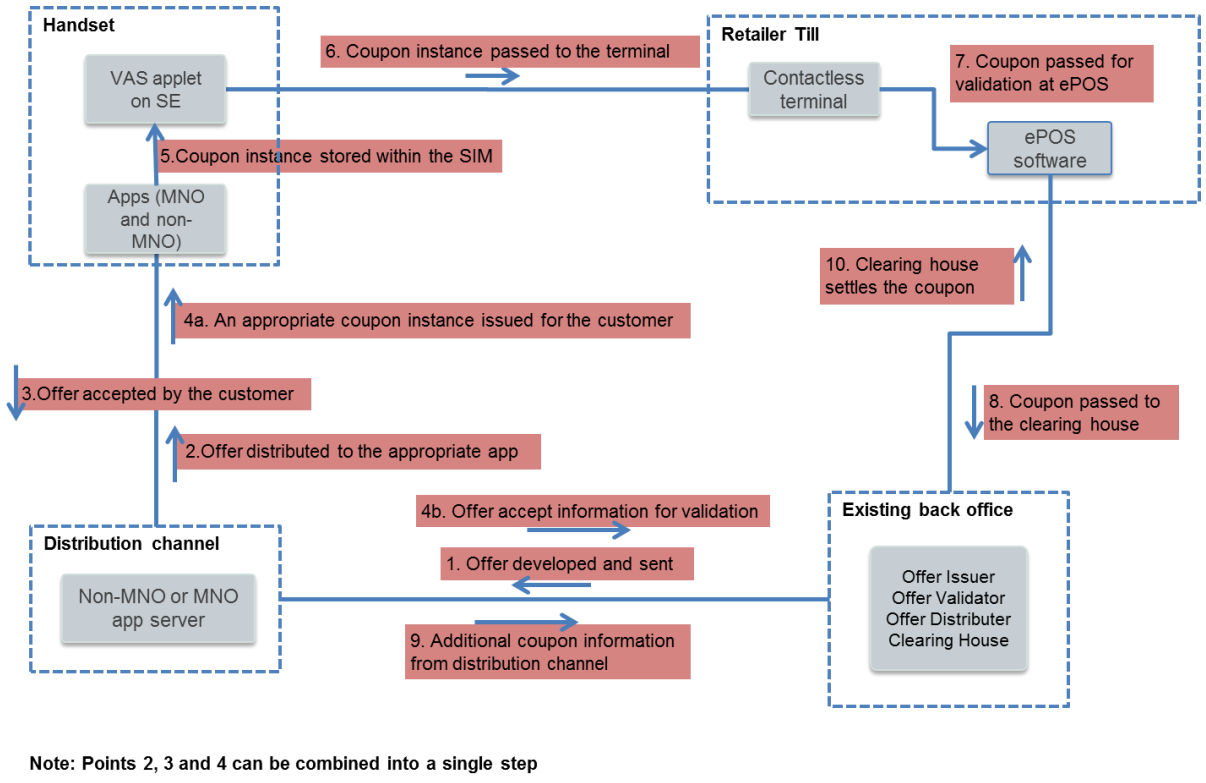


Figure 24**: Coupons messaging journey flow**

This consists of the following elements:

10. Offer sent to server: Offer awarder and offer distributor prepare the offer, relevant marketing content and identify the target customer base and send this offer to the application servers.
11. Offer distributed to apps: App servers then send the offers to the relevant customers via the installed apps on customer mobile device.
12. Offer accepted by user: User likes the offer and accepts it.
13. Coupon issued: Relevant unique coupon is issued to the customer:
    a) The issued coupon instance is passed to the relevant app on the customer handset
       Or
    b) The offer accept notification is sent to the relevant backend services for validation.
14. Coupon activated: The coupon is stored within the VAS Applet for redemption when the customer taps next.
15. Coupon data passed to POS: The coupon instance is passed to the contactless reader at the POS.
16. Validation at the ePOS: The coupon is redeemed and customer gets the offer with the goods/service.
17. Coupon clearing: Merchant passes the coupon redemption information to the clearing agency for a claim.
18. Coupon audit: Clearing agency may deploy additional software to get customer information from the app servers so as to audit the coupon redemption information.
19. Coupon cleared: Clearing agency confirms the coupon redemption.

In case of passing the Customer Identifier through the system, Step 4b is used instead of step 4a. Also, the distribution channels could be as simple as a smart poster, or a QR code from where the customer can download the offer on their handsets. Lastly, steps 2, 3, and 4 can be combined from a user experience point of view.

### 4.3.1 Coupon validation data flow

Coupon validation can happen in two ways: Online and offline. For this document, online validation is specified to conform to the GS1 standard [4]. Support for offline/batch processing is expected to be released in a later addendum.

1. **Online validation**: The ePOS connects to an online database of offers hosted either by the offer issuer or the clearing house. The flow presented in Figure 24 has been modified to incorporate online validation, and is shown in Figure 25.

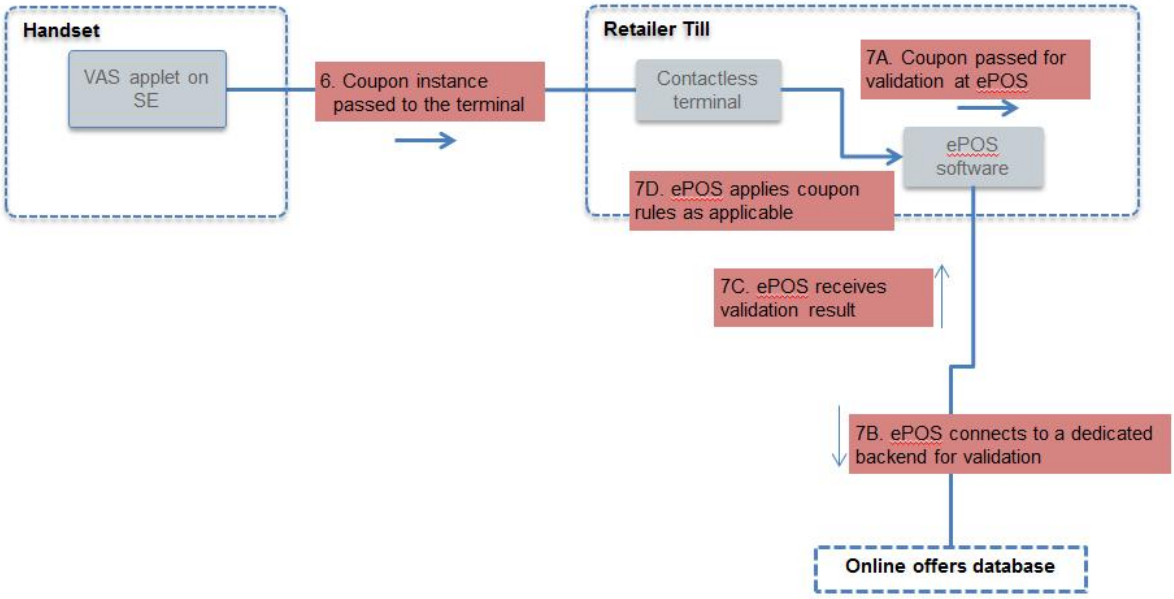


Figure 25**: Online validation**

2. **Offline Validation**: There are two options for Offline Validation –

   a) Download an Offer Validation File: ePOS downloads an 'offer validation file' at a set interval (possibly daily or weekly) and uses this file to validate a coupon.

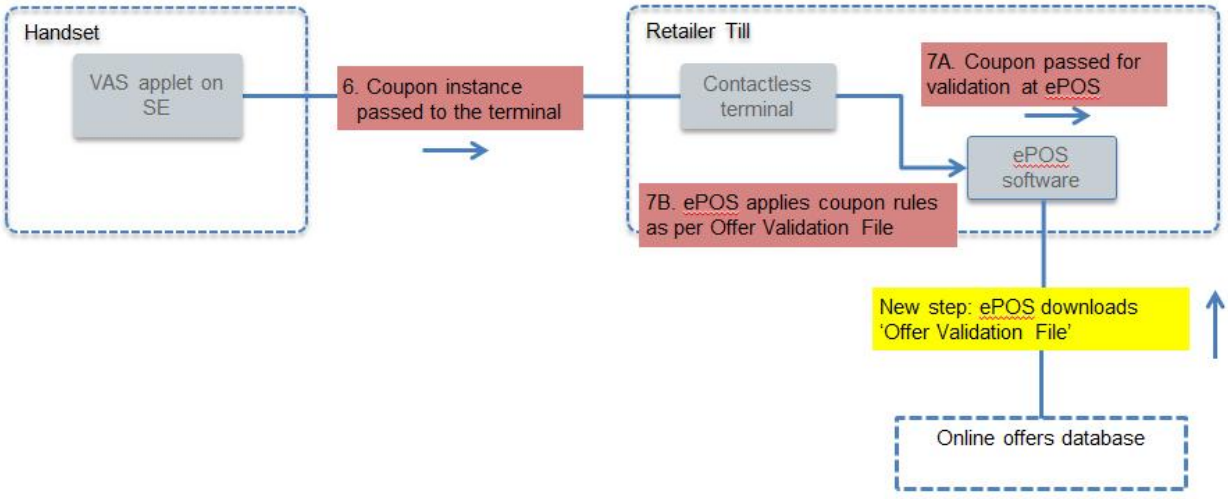


Figure 26**: Offline validation example (download offer file)**

b) Coupon Instance ID contains validation: The coupon ID itself contains self-validation data and the ePOS can act on this. This is the current scenario with paper coupons.

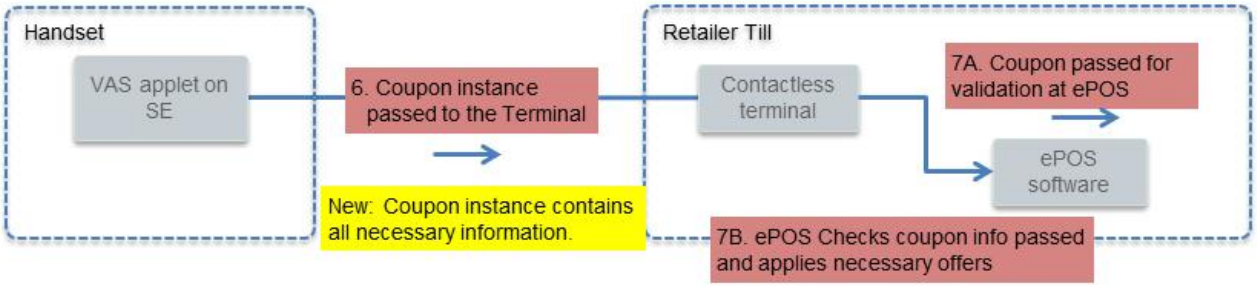


Figure 27**: Offline validation example (Coupon instance contains validation)**

# 5 Ecosystem considerations

This document is intended to be a high level proposal of the complete architecture, rather than a fully defined set of specifications (which could be part of a future release). This section will review specific guidelines for each critical actor within the wider ecosystem.

## 5.1 Merchant, loyalty scheme, manufacturers

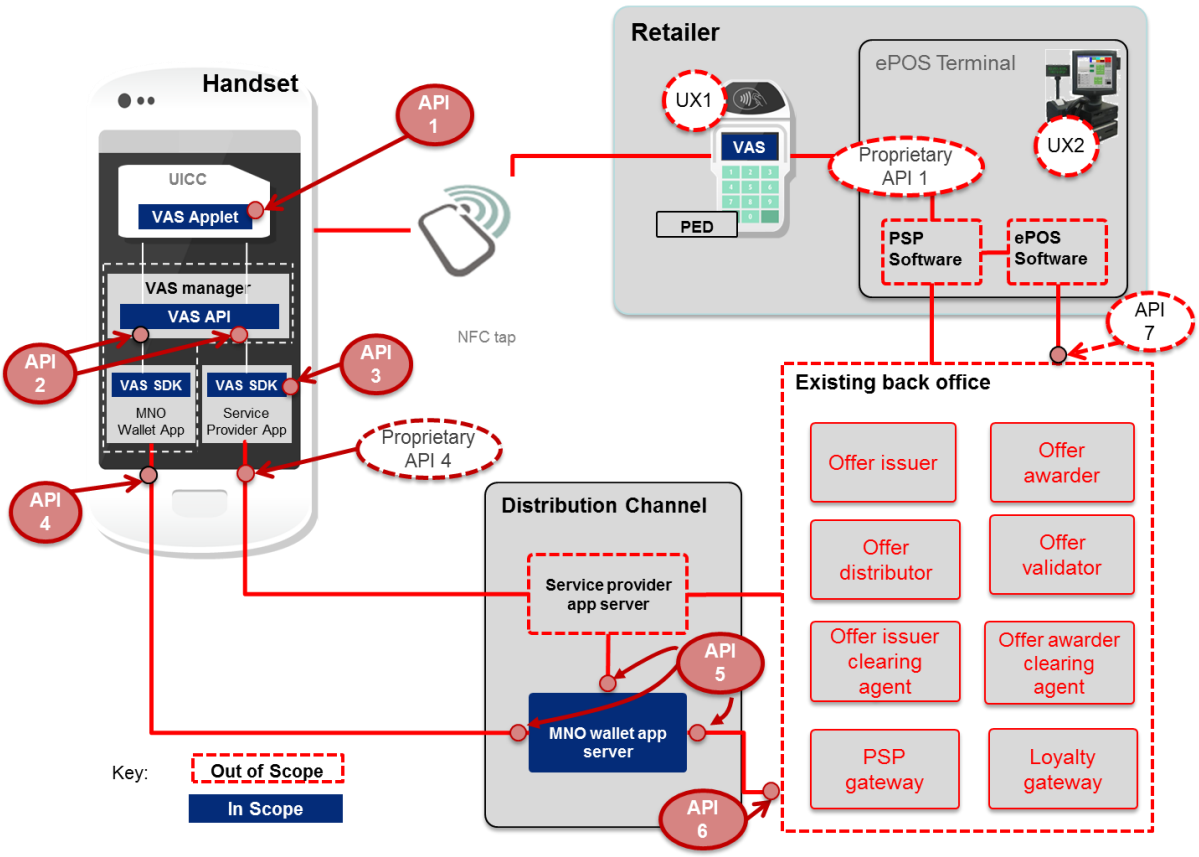


Figure 28**: Proposed APIs in the existing ecosystem**

A guiding principle for this proposal is to minimise the necessary effort for merchants, loyalty schemes and other coupon partners to be part of this ecosystem. The following represents the minimum necessary effort for each actor in order to support the overall architecture.

### 5.1.1 Merchant

20. Update the PED (PIN Entry Device) so it can look for coupons and loyalty that comes via the handset.
21. Update the ePOS software to validate the coupons and loyalty ID that come in via the PED. In case there is no connection between the PED and the ePOS (via proprietary API 1), another mechanism for transferring the necessary data needs to be in place. For example, QR codes that are displayed on the mobile device and read directly by the ePOS.

22. Develop a digital way of informing the offer awarder clearing agent of a coupon that has been redeemed. This is essential for payment to the merchant, and will replace the paper coupon that is presently sent by the merchant to the clearing agency. This could be a potential API 7, which is out of scope for this document.
23. Use API 3: implement a way to store the necessary VAS information on the actual VAS applet (on the secure element) – this is only relevant if the merchants are using their own applications. In the event that they use MNO wallet apps, they do not need to develop this.

### 5.1.2 Loyalty schemes

Loyalty service providers can provide digital loyalty and digital coupons to their users via this method. In some cases, they will collaborate with the merchant or they might be the merchant themselves. The following steps illustrate how loyalty schemes can be part of this ecosystem:

24. Service provider app server: This is an optional item and provides a distribution channel for the relevant information to go into specific apps (proprietary API 4). In case this is not used, they can still use the MNO provided app server to distribute relevant content (via API 5).
25. Connections of the app servers to their existing backend (via API 6).
26. Use API 3: Again optional, if the loyalty service providers develop their own apps, they will need to use a high level API 3 to connect to the VAS applet (either via VAS SDK, VAS manager or directly – whichever is made available by the relevant MNO).

### 5.1.3 Manufacturers

Various fast moving consumer goods (FMCG) manufacturers will have a unique distribution channel to reach their customers and provide them with relevant offers and loyalty. They will need to work with the offer awarders, distributors and clearing houses in order to achieve this. The following steps illustrate how manufacturers can be part of this ecosystem:

27. Working with the offer awarders, manufacturers will issue unique digital coupons.
28. Manufacturers can develop their own in-house apps and related app servers to distribute these offers and coupons to their users. Optionally, they can leverage other app servers (like the MNO wallet app server) to complete this.
29. Digital validation mechanism in partnership with the clearing houses: Manufacturers will need to develop a new mechanism for digital validation of the coupons. This will be more secure, prevent fraud and provide capabilities to enhance the data.

# 6 Application programming interface (API)

## 6.1 API 1 – Application protocol data unit (APDU) interface from the contactless reader to the VAS applet

### 6.1.1 Scope

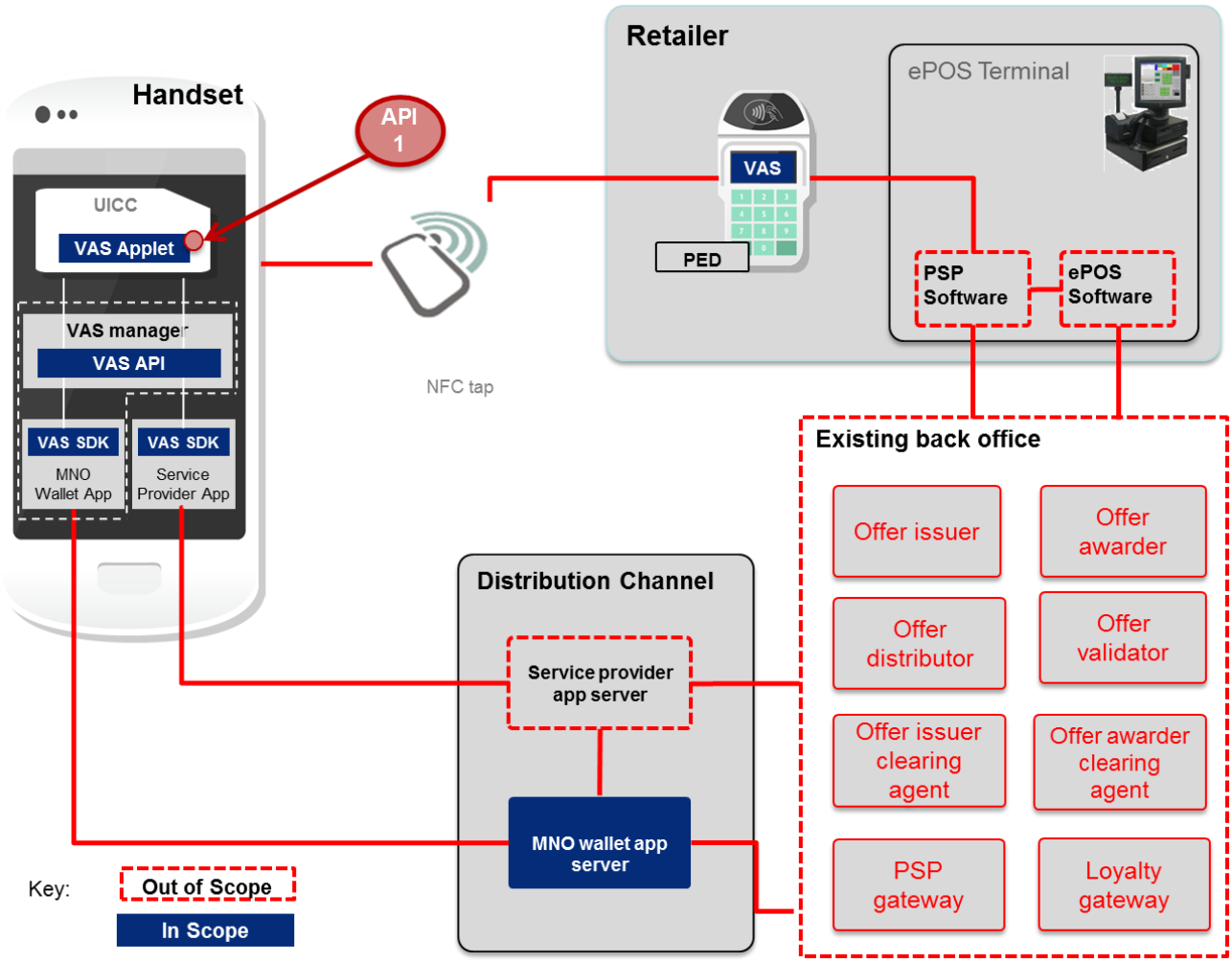


Figure 29**: Location of API 1 within the proposed architecture**

API 1 defines the interface between the VAS applet and the contactless reader/terminal. The proposal is based upon the GSMA NFC Wallet-POS Proposal (Version 1.0) [1]. The intention is for that specification to be enhanced to incorporate the requirements identified during this work. The API defined here only supports couponing and loyalty and does not include payment functionality or payment initiation within the VAS interaction.

### 6.1.2 Actors

| Actor | Description |
|---|---|
| Offer Awarder | The offer awarder is the merchant in the use cases associated with this API and is only responsible for the initiation of the transaction. |

| Actor | Description |
|---|---|
| ePOS | The ePOS is the merchant POS infrastructure in the use cases associated with this API. It is only responsible for the initiation of the transaction to the contactless reader. |
| Contactless Reader | The contactless reader is the merchant device that interacts with the consumer's phone and VAS applet to perform the transaction e.g. the PED |
| VAS Applet | The VAS applet resides within the UICC on the consumer's mobile device and interacts with the contactless reader to perform a transaction. |
| Offer User | The offer user in the use cases associated with this API is only responsible for tapping the mobile device on the contactless reader. |

**Table 15: Actors for API 1**

### 6.1.3 Use cases

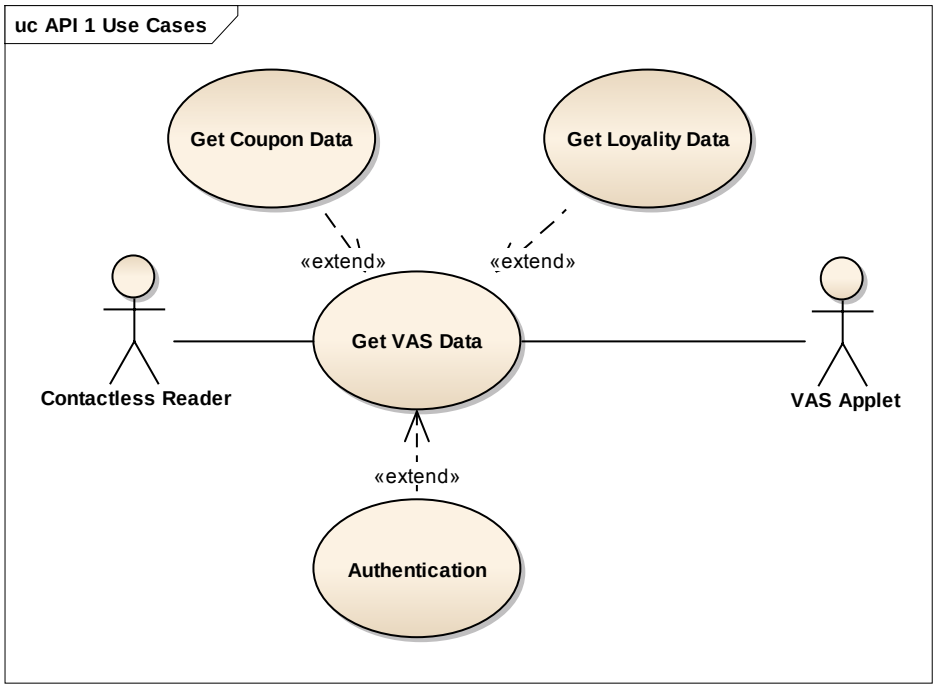


Figure 30**: API 1 Use cases**

| Use Case | Description |
|---|---|
| Get VAS data | **Source:** Contactless reader<br>**Target:** VAS applet<br>**Description:** Retrieval of the data from the VAS applet. Includes the capability to retrieve coupon data or loyalty data. |
| Get coupon data / Get loyalty ID | **Source:** Contactless reader<br>**Target:** VAS applet<br>**Description:** Retrieval of coupon/loyalty data from the VAS applet. |

| Use Case | Description |
|---|---|
| Authentication | **Source:** Contactless reader<br>**Target:** VAS applet<br>**Description:** Enables the contactless reader to deliver data to the VAS applet to enable authentication/security handshake. |

**Table 16: API 1 Use cases**

### 6.1.4 VAS Applet APDU commands

The VAS Applet is a Java Card applet running on the UICC, communicating with the POS system via NFC card emulation mode. This has previously been defined in NFC core wallet requirements [2]

#### 6.1.4.1 Selection of VAS application

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

The contactless reader will send an API 1 `Select(…)` command with `VASAppAID` as a parameter. This will be the partial AID (Registered application provider identifier – RID – part only, first 7 bytes, assuming the application code to be `0x00:0x01`) reserved for value added services across all PED's globally. In the case where a single matching AID is present within the UICC, it will be returned to the contactless reader and a secure channel will be established. When more than one matching AID is present, this will be indicated by the returned `status` object. In this case, a proximity payment system environment (PPSE)-like mechanism can be followed by first interrogating the applet directory. MNOs have an implementation option here: use a single AID return call, or in-case of provisioning multiple applets, use a PPSE-like return as a response to `Select`.

30. Single AID: MNOs understand that there will only ever be a need for a single VAS applet on the UICC for their market, and hence provide that back directly to the contactless reader.
31. PPSE-like response: There could be a possibility of multiple VAS applets on a single UICC. These could be by the same operator, multiple operators or even third-party service providers (as agreed between the operators and the service providers). To maintain interoperability with the previous option, and also the save on communication bandwidth, the Operator applet must now indicate this with the `status` message in the return call. Following this, the applet should implement `VASAppletDirectory()`, that passes the applet directory structure on response to contactless reader's `GetVASAppletDirectory()`. This is akin to PPSE-like response that returns a directory list of all the AID's available to the contactless reader. The contactless reader should be configured to select a single AID from this response, which it wishes to communicate with.

Also to consider here are security mechanisms that might be needed to authenticate the communication between the contactless reader and the applet. We recognise the need for such mechanisms, but also understand that varying business requirements and existing

technology architectures make it difficult to recommend a certain single solution suitable for all implementations. Hence, the requirement has been provided for a security set up, but actual implementation details will be followed in future revisions after consultation with the wider industry.

The `VASAppAID` is the application identifier of the MNO's VAS Applet. The encoding of the `VASAppAID` is defined according to Table 17:

| Meaning | AID-Coding |
|---|---|
| *RID*<br>This RID is internationally unique and reserved for applications according to this proposal | `0xA0:0x00:0x00:0x05:0x59` |
| *Application Code* | `0x00:0x01` |
| *Mobile Country Code*<br>Three digit code (as per ITU-T E.212 [15]) as a BCD, right aligned, padded with F, e.g. 0xF2:0x62 Germany | `0xFX:0xXX` |
| *Mobile Network Code*<br>Two or three digit code (as per ITU-T E.212 [15]) as a BCD, right aligned, padded with F, e.g. T-Mobile Deutschland GmbH 0xFF:0x01 | `0xFX:0xXX` or `0xFF:0xXX` |
| *Application Provider Field (optional, length 0-5)*<br>Additional provider specific data | `0xXX:0xXX:0xXX:0xXX:0xXX` |

**Table 17: VAS Applet AID**

| Field name | Type | Length | Value | Description |
|---|---|---|---|---|
| `CLA` | `Byte` | `1` | `0x00` | Instruction class: No secure messaging or no secure messaging indication |
| `INS` | `Byte` | `1` | `0xA4` | Select File |
| `P1` | `Byte` | `1` | `0x04` | Direct selection by dedicated file name (Data field = dedicated file name) |
| `P2` | `Byte` | `1` | `0x00` | First record, Return FCI, optional template |
| `Lc` | `UInt8` | `1` | `0x07` | Length of the AID |
| `Data` | | `Lc` | `0xA0:0x00:0x00:0x05:0x59 :0x00:0x01` | VAS Applet AID |
| `Le` | `Byte` | `1` | `0x00` | No maximum length |

**Table 18: VAS Applet `Select File` Command**

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

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Status Code | Short | 2 | 0xXX:0xXX | SW1 SW2 (defined in Section Table 23) |
| | | e.g. | 0x90:0x00 | Operation successfully completed |
| | | e.g. | 0x6A:0x82 | File not found |
| | | e.g. | 0x61:0xXX | More data available (more than one VAS applet present, GetVASAppletDirectory call required |

**Table 19: VAS Applet `Select File` Response**

#### 6.1.4.2 `GetVASData` command

With the command `GetVASData`, data objects can be retrieved from the VAS applet. The requested objects can be specified by using a filter criterion (unique or a combination of `MerchantID`, `LoyaltyIssuerID`, and `CouponIssuerID`) and 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 can decide very flexibly which data objects it wants to read.

Each individual implementation will have the liberty to enforce their proprietary business rules in respect of the security, authentication and data cardinality considerations. This flexibility is provided as there are multiple proprietary retailer ePOS and loyalty systems in use worldwide, each using varying sets of data exchange. However, by encapsulating the flexible data message in a generic `GetVASData` command, a common reference is achieved for communication between the VAS applet and the contactless reader, while preserving the proprietary nature of the data.

In particular, it is important that each implementation should implement sufficient security to safeguard business processes e.g. one merchant's PED should not be able to access another merchant's coupons from the VAS applet via the `GetVASData` command (or any other API1 interfaces) without explicit authorisation having been previously given. See section 7 for guidance on how to safeguard this process with an overview of specific security measures than can support this requirement. More in depth analysis of security approaches will feature in future documentation.

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| CLA | Byte | 1 | 0x00 | Instruction class: No secure messaging no secure messaging indication |
| INS | Byte | 1 | 0xCB | Custom instruction: GetVASData |
| P1 | Byte | 1 | 0x00 | |
| P2 | Byte | 1 | 0x00 | |
| Lc | UInt8 | 1 | 0xXX | Length of Data field |

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Data | TagList | Var. | | List of Tags required. |
| Le | Byte | 1 | 0x00 | No maximum length |

**Table 20: VAS Applet `GetVASData` Command**

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Data | | 0..255 | | |
| Status Code | Short | 2 | 0xXX:0xXX | SW1 SW2 (defined in Section Table 23) |
| | | e.g. | 0x90:0x00 | Operation successfully completed |
| | | e.g. | 0x67:0x00 | Wrong data length |
| | | e.g. | 0x6A:0x80 | Incorrect parameters in the Data field |

**Table 21: VAS Applet `GetVASData` Response**

#### 6.1.4.3 `PutVASData` command

With the command `PutVASData`, 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 `PutVASData` command, the Contactless Reader has the possibility to update multiple tokens in one step flexibly by simply adding them to the data field. This is as per ISO 7816-4 [13]

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| CLA | Byte | 1 | 0x00 | Instruction class: No secure messaging or no secure messaging indication |
| INS | Byte | 1 | 0xDB | Custom instruction: `PutVASData` |
| P1 | Byte | 1 | 0x00 | |
| P2 | Byte | 1 | 0x00 | |
| Lc | Byte | 1 | 0xXX | Length of `Data` field |
| Data | | 0..255 | | Token(s) with their updated status. |

**Table 22: VAS Applet `PutVASData` Command**

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Status Code | Short | 2 | 0xXX:0xXX | SW1 SW2 (defined in Section Table 23) |
| e.g. | | | 0x90:0x00 | Operation successfully completed |
| | | | 0x9C:0x06 | Access denied |

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| | | | 0x9C:0x01 | Insufficient memory onto the UICC to complete the operation |

**Table 23: VAS Applet `PutVASData` Response**

#### 6.1.4.4 VAS applet response codes

These codes are based on ISO 7816 [13] (standard codes are marked as 'ISO').

| 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 $L_e$ and $L_c$). (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 available within the UICC to complete the operation. |
| 9C | 02 | Unsuccessful authentication. |
| 9C | 03 | Operation not allowed because of the internal state of the VAS applet. |
| 9C | 05 | The requested feature is not supported either by the UICC or by the VAS applet. |
| 9C | 06 | Unauthorised. (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 24: VAS applet response status codes**

### 6.1.5 Message flows

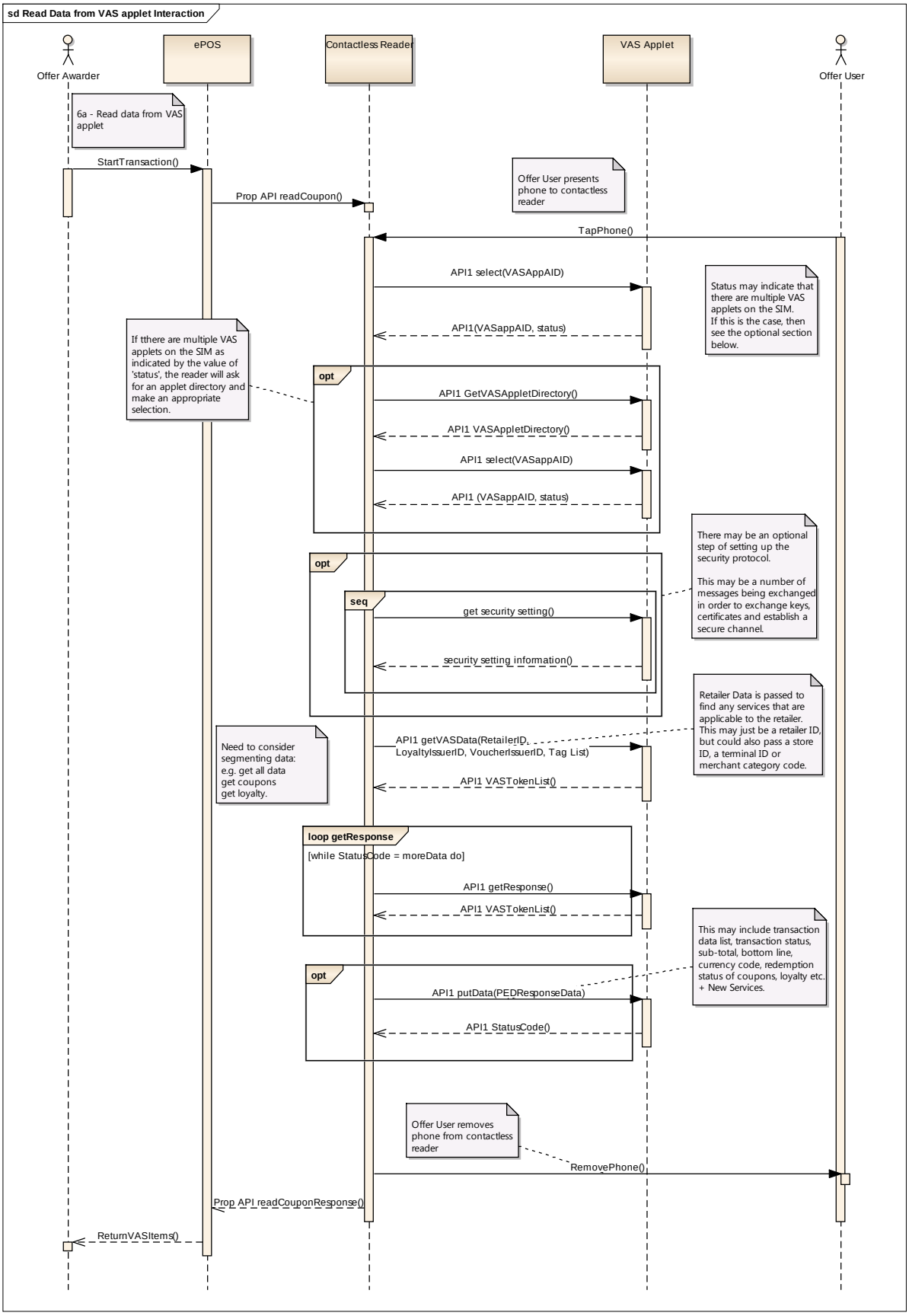


Figure 31**: Read data from VAS applet**

## 6.2 API 2 – Application protocol data unit (APDU) command interface from the VAS manager to the VAS applet

### 6.2.1 Scope

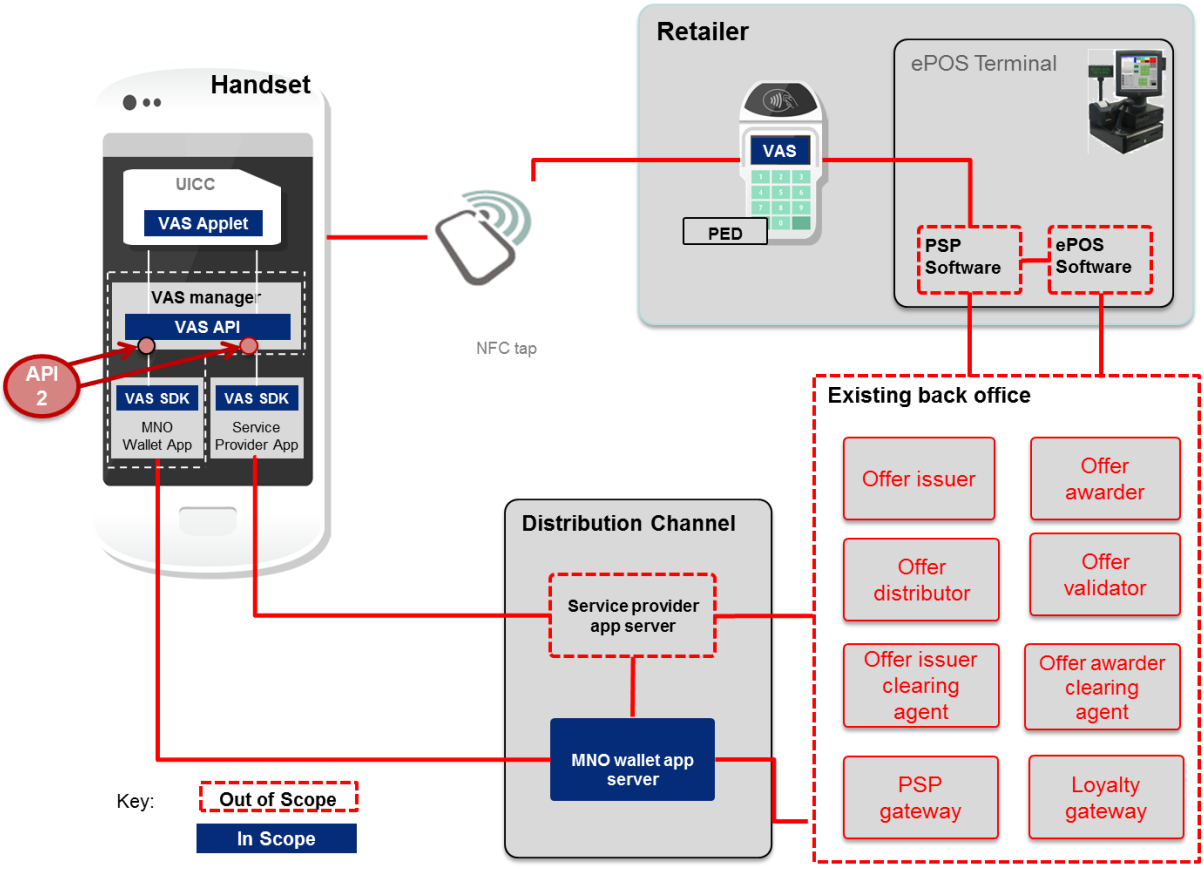


Figure 32**: Location of API 2 within the proposed architecture**

API 2 defines the interface between the VAS applet and the VAS manager and it can be seen that this is almost identical to API 1; just the flows are different. The proposal is based upon the GSMA NFC Wallet-POS Proposal (Version 1.0) [1]. The intention is for that specification to be enhanced to incorporate the requirements identified during this work. The API defined here only supports couponing and loyalty and does not include payment functionality or payment initiation within the VAS interaction.

### 6.2.2 Actors

| Actor | Description |
|---|---|
| VAS SDK/VAS manager | Background component to manage access between wallet apps and the VAS applet on the UICC. |
| VAS applet | The VAS applet resides within the UICC and interacts with the VAS manager to provision coupons and loyalty IDs for later presentation to the contactless reader within a transaction. |

### 6.2.3 Use cases

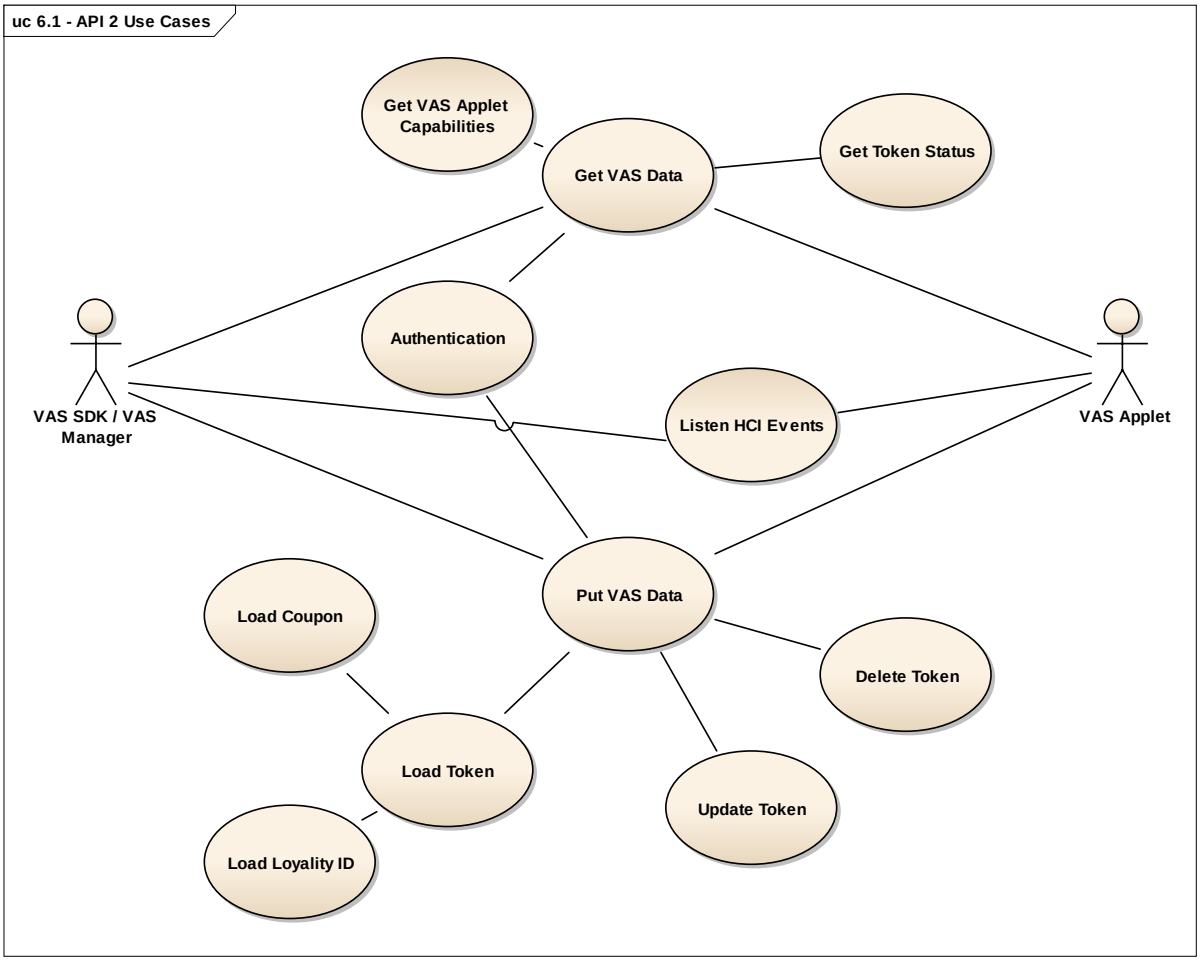


Figure 33**: API 2 use cases**

| Use Case | Description |
|---|---|
| Get VAS Data | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Retrieval of the data from the VAS applet. Includes the capabilities of the VAS applet and the status of existing coupons |

| Use Case | Description |
|---|---|
| Get VAS Applet Capabilities | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Retrieval of the VAS applet capabilities from the VAS applet. |
| Get Token Status | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Retrieval of the status of tokens (coupons or loyalty IDs) from the VAS applet. |
| Authentication | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Enables the VAS SDK or VAS manager to deliver data to the VAS applet to enable authentication/security. |
| Put VAS Data | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Enables the VAS SDK or VAS manager to load, update or delete tokens (coupons or loyalty IDs) in the VAS applet and to enable authentication/security around the tokens. |
| Load Token (Load Coupon / Load Loyalty ID) | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Enables the VAS SDK or VAS manager to load a token into the VAS applet. Token is any form of a data that is needed to be stored on the applet. In this instance, it can be a coupon or a loyalty ID. |
| Update Token | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Enables the VAS SDK or VAS manager to update a token in the VAS applet. (Redeem, update). |
| Delete Token | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Enables the VAS SDK or VAS manager to delete a token in the VAS applet. |
| Authentication | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description:** Enables the VAS SDK or VAS manager to deliver data to the VAS applet to enable authentication/security. |
| Listen HCI Events | **Source:** VAS SDK/VAS manager<br>**Target:** VAS applet<br>**Description**: Enables the VAS SDK or VAS manager to listen to HCI event EVT_TRANSACTION that will be issued by the applet. |

**Table 25: Use case description for API 2**

### 6.2.4 VAS Applet APDU commands

The VAS Applet is a java card applet running on the UICC, communicating with the VAS SDK/VAS manager using ISO 7816 APDUs [13].

#### 6.2.4.1 Selection of VAS application

Like other UICC applets, the VAS Applet is identified and selected by its AID. The standard selection command is called by POS system to make the VAS Applet selected and ready for further APDU processing.

The `VASAppAID` is the application identifier of the MNO´s VAS Applet.

The encoding of `VASAppAID` is defined according to the following table:

| Meaning | AID-Coding |
|---|---|
| *RID*<br>This RID is internationally unique and reserved for applications according to this proposal | `0xA0:0x00:0x00:0x05:0x59` |
| *Application Code* | `0x00:0x01` |
| *Mobile Country Code*<br>Three digit code (as per ITU-T E.212 [15]) as a BCD, right aligned, padded with F, e.g. 0xF2:0x62 Germany | `0xFX:0xXX` |
| *Mobile Network Code*<br>Two or three digit code (as per ITU-T E.212 [15]) as a BCD, right aligned, padded with F, e.g. T-Mobile Deutschland GmbH 0xFF:0x01 | `0xFX:0xXX` or `0xFF:0xXX` |
| *Application Provider Field (optional, length 0-5)*<br>Additional provider specific data | `0xXX:0xXX:0xXX:0xXX:0xXX` |

**Table 26: VAS Applet AID**

| Field name | Type | Length | Value | Description |
|---|---|---|---|---|
| `CLA` | `Byte` | `1` | `0x00` | Instruction class: No secure messaging or no secure messaging indication |
| `INS` | `Byte` | `1` | `0xA4` | Select File |
| `P1` | `Byte` | `1` | `0x04` | Direct selection by dedicated file name (Data field = dedicated file name) |
| `P2` | `Byte` | `1` | `0x00` | First record, Return FCI, optional template |
| `Lc` | `UInt8` | `1` | `0xXX` | Length of the AID |
| `Data` | | `11-16` | `0xA0:0x00:0x00:0x05:0x59:0x00:0x01:0xFX:0xXX:0xFX:0xXX(:0xXX:0xXX:0xXX:0xXX:0xXX)` | VAS Applet AID (as defined in Table 26) |
| `Le` | `Byte` | `1` | `0x00` | No maximum length |

**Table 27: VAS Applet `Select File` Command**

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Data | FCI template | 0..255 | | File control information (optional) |
| Status code | Short | 2 | 0xXX:0xXX | SW1 SW2 (defined in Section 6.2.4.4) |
| | | e.g. | 0x90:0x00 | Operation successfully completed |
| | | e.g. | 0x6A:0x82 | File not found |
| | | | 0x61:0xXX | More data available (more than one VAS applet present, GetVASAppletDirectory call required |

**Table 28: VAS Applet Select File Response**

### 6.2.4.2 GetVASData command

With the command GetVASData, data objects can be retrieved from the VAS applet. 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 | Instruction class: No secure messaging no secure messaging indication |
| INS | Byte | 1 | 0xCB | Custom instruction: GetVASData |
| P1 | Byte | 1 | 0x00 | |
| P2 | Byte | 1 | 0x00 | |
| Lc | UInt8 | 1 | 0xXX | Length of Data field |
| Data | TagList | Var. | | List of Tags required |
| Le | Byte | 1 | 0x00 | No maximum length |

**Table 29: VAS Applet GetVASData Command**

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Data | | 0..255 | | |
| Status code | Short | 2 | 0xXX:0xXX | SW1 SW2 (defined in Section 6.2.4.4) |
| | | e.g. | 0x90:0x00 | Operation successfully completed |
| | | e.g. | 0x67:0x00 | Wrong data length |
| | | e.g. | 0x6A:0x80 | Incorrect parameters in the data field |

**Table 30: VAS Applet `GetVASData` Response**

#### 6.2.4.3 `PutVASData` command

With the command `PutVASData`, data objects can be sent to the VAS applet, for example the `MerchantID` and tokens. With the `PutVASData` command, the VAS SDK or VAS manager has the possibility to transport multiple data objects and update multiple tokens in one step flexibly, by simply adding them to the `Data` field. .

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| CLA | Byte | 1 | 0x00 | Instruction class: No secure messaging or no secure messaging indication |
| INS | Byte | 1 | 0xDB | Custom instruction: PutVASData |
| P1 | Byte | 1 | 0x00 | |
| P2 | Byte | 1 | 0x00 | |
| Lc | Byte | 1 | 0xXX | Length of Data field |
| Data | | 0..255 | | Token(s) with their updated status |

**Table 31: VAS Applet `PutVASData` Command**

| Field | Type | Length | Value | Description |
|---|---|---|---|---|
| Status Code | Short | 2 | 0xXX:0xXX | SW1 SW2 (defined in Section 6.2.4.4) |
| | | | 0x90:0x00 | Operation successfully completed |
| | | | 0x9C:0x01 | Insufficient memory available to the UICC to complete the operation |
| | | | 0x9C:0x06 | Access denied |

**Table 32: VAS Applet `PutVASData` Response**

#### 6.2.4.4 VAS applet response codes

These codes are based on ISO 7816 [13] (standard codes are marked as 'ISO')

| 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 $L_e$ and $L_c$) (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 available to the UICC to complete the operation. |
| 9C | 02 | Unsuccessful authentication. |
| 9C | 03 | Operation not allowed because of the internal state of the VAS applet |
| 9C | 05 | The requested feature is not supported either by the UICC or by the VAS applet |
| 9C | 06 | Unauthorised (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. |

### 6.2.5 Message flows

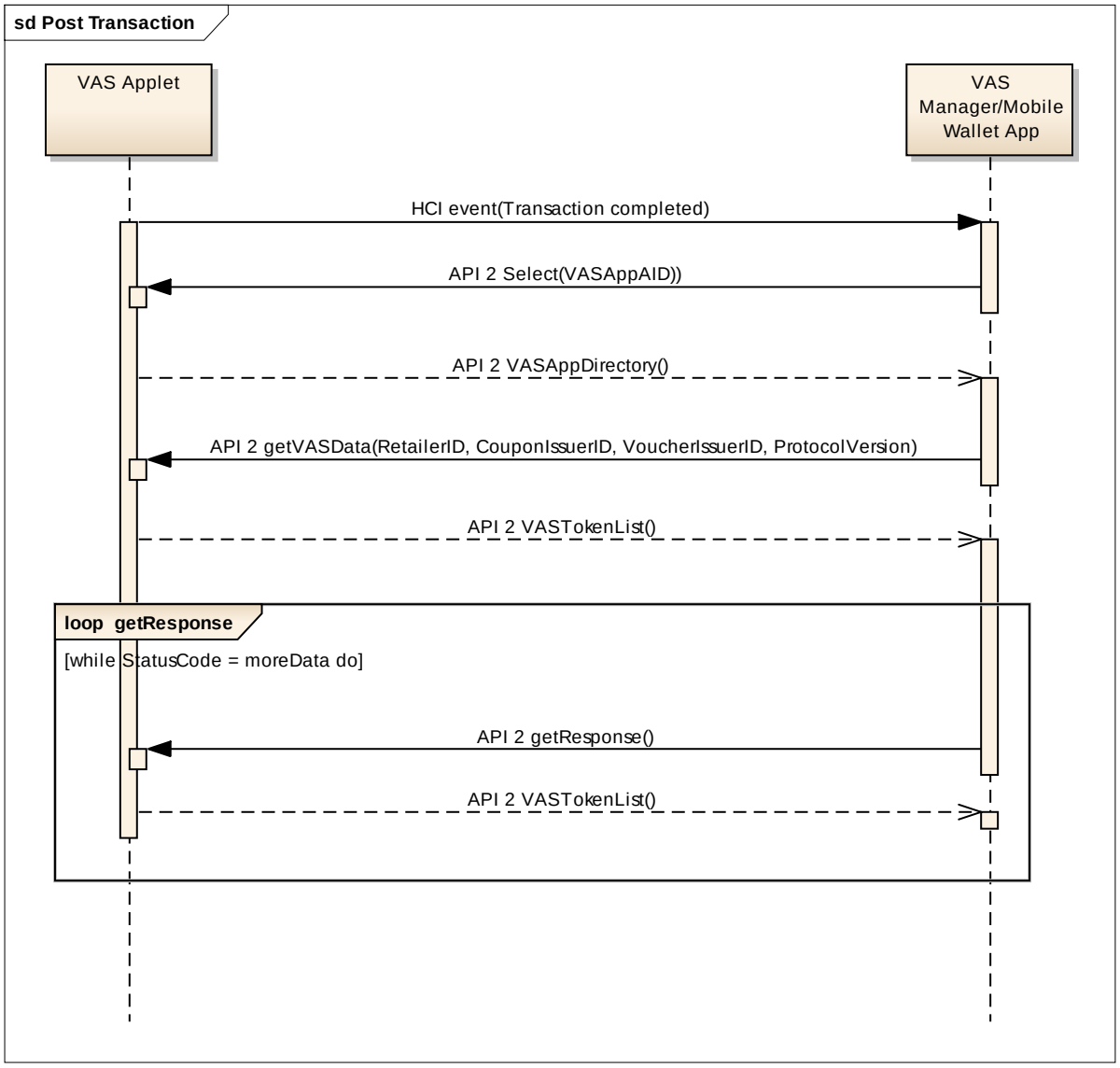


Figure 34**: Message flow for API 2**

## 6.3 API 3 – VAS manager API

### 6.3.1 Scope

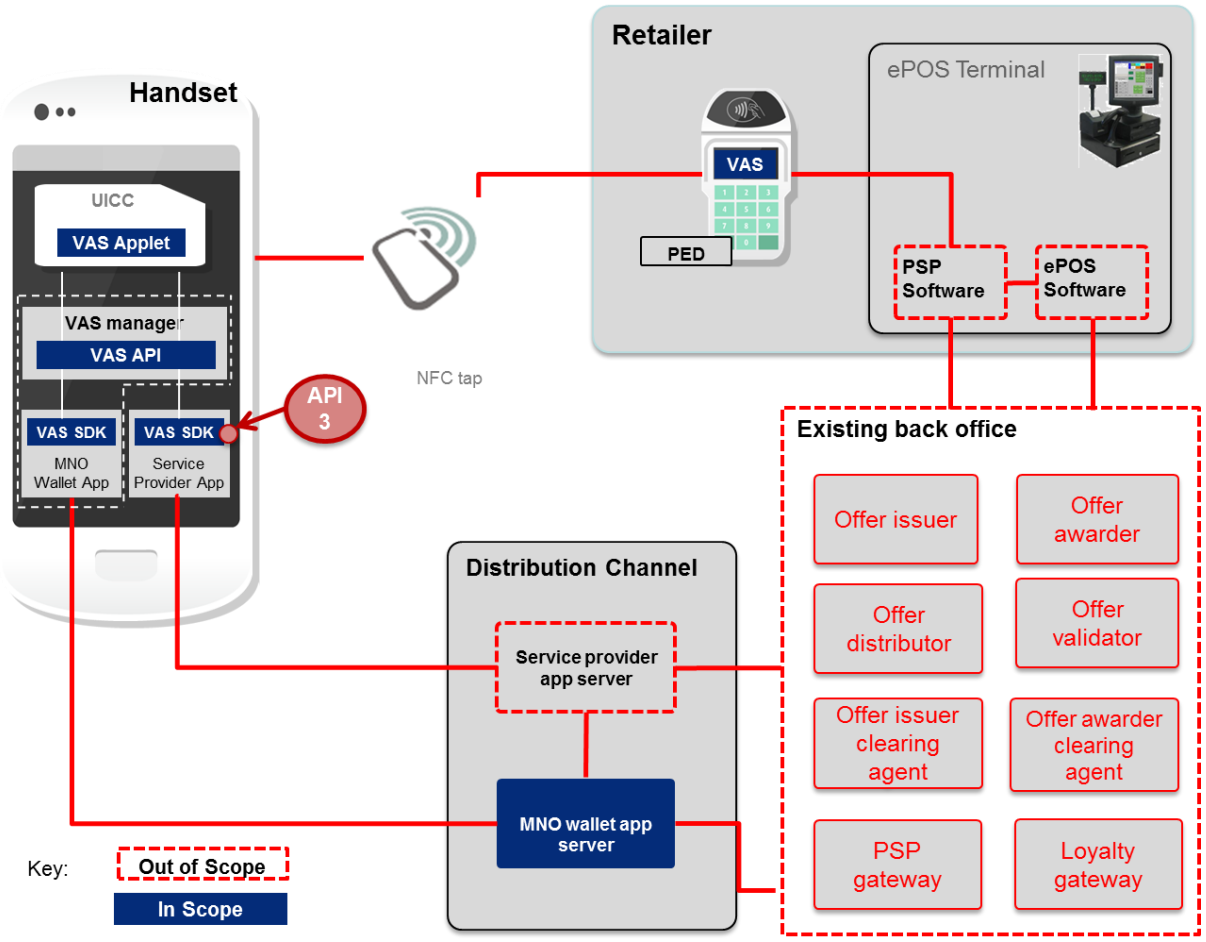


Figure 35**: Location of API 3 within the proposed architecture**

This section describes the on-device MNO wallet app (also referred to as core wallet) APIs – APIs offered by the MNO wallet app or the VAS manager. These APIs will be available to various service provider applications that want to interact with the VAS manager.

Any backend processes via the Internet or the internal subsystems of any stakeholders are explicitly out of scope for this document. Additionally, it does not address functions on the UICC or other secure elements on the handset which might be needed to implement specific core wallet API functions.

Any references to such processes or functions are only informative in nature and purely provided to aid understanding.

The APIs described in this section are only a subset of all APIs that a full-featured core wallet (GSMA NFC Core Wallet Requirements [2]) would offer any service provider app on the handset that wishes to interface to the MNO wallet app (also referred to as service provider man-machine interfaces, or SP-MMIs). They have been reduced to those APIs needed for the coupon and loyalty proposition described in this document.

At a later stage, this API 3 chapter will be extended into a specification describing the complete set of core wallet APIs covering use cases beyond couponing and loyalty. When the core wallet API document is available this chapter will be replaced by a reference to that external specification document.

### 6.3.2 Architectural considerations

API 3 can be constructed in two ways. Option 1 is a simplified API and option 2 is a complete feature-rich API:

Option 1: Basic intermediate API 3: A high-level abstraction for APDU commands, making it easier for external developers to use the VAS applet. Also, it provides basic access management (such as which apps have which rights). There will be no in-depth coupon management system and apps will have to handle that accordingly. VAS applet uses simple and straightforward logic.

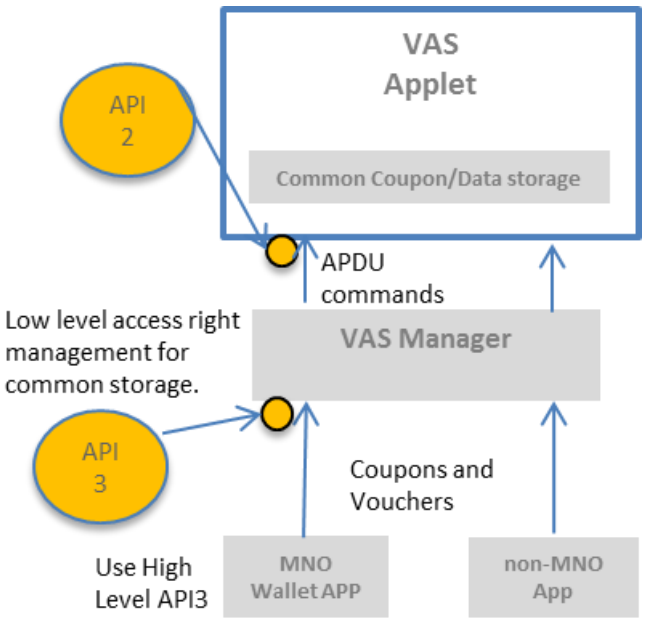


Figure 36**: Basic VAS manager interaction**

Option 2: Full featured API 3: A full featured API providing a high-level abstraction for apps to put coupons and other data on the VAS applet, as well as intelligently controlling the coupon management within the VAS applet via a bucket system.

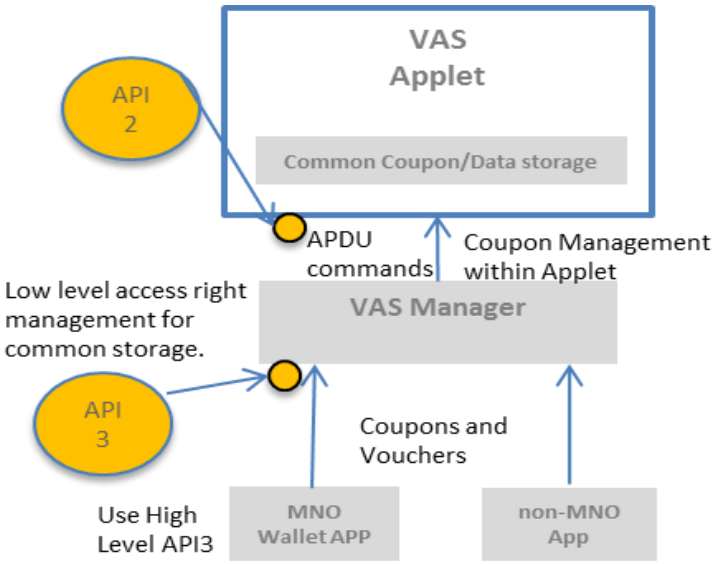


Figure 37**: A full featured VAS manager**

One way intelligent controlling of coupons on the applet could work is described below. This is an implementation guideline only. The technical details needed for this specific implementation are out of scope, as this document details only the high-level overall storage available on the UICC:

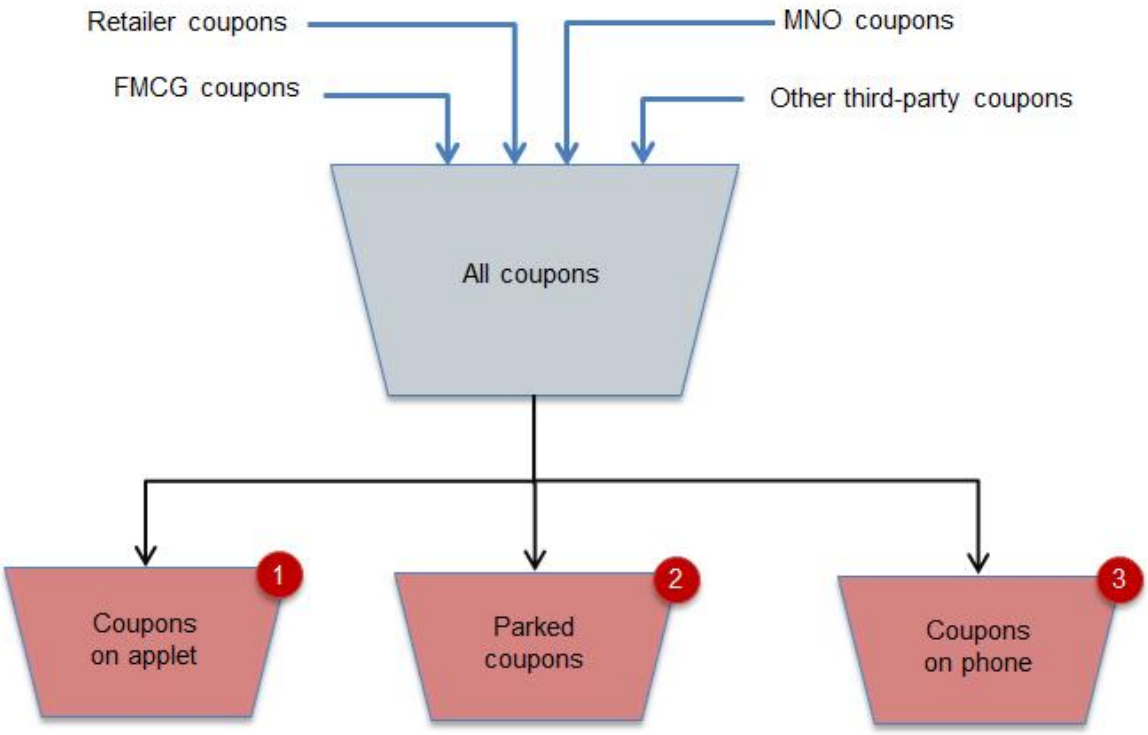


Figure 38**: Potential scenario to implement buckets**

The API would employ the following decisions to put coupons on relevant buckets.

As far as space permits, any coupon being sent to the API to be placed on the applet will reach bucket 1 “coupons on applet” via the call-back method and the API will inform the app of the success/failure of this operation.

When the applet’s available memory for coupons space is full, it will employ an intelligent First In, First Out (FIFO) approach. The merchant trying to push the next coupon will have all of their coupons put onto the applet. Any coupons that cannot be redeemed at this retailer taken from the applet using the FIFO approach, and placed in bucket 2 “parked coupons”. Bucket 2 sits within the mobile device memory rather than the UICC memory.

Once space is available on the applet for more coupons, bucket 2 coupons will be placed into bucket 1 by the FIFO approach.

Bucket 3 will contain the remaining coupons available to the user but have not been asked to be placed on the applet by a specific app or by the user.

If the user explicitly asks for a set of coupons to be placed on the applet, this will take priority against any other, and if needed other coupons will be removed and placed in bucket 2. This removal will employ the FIFO approach.

In the rare event of one retailer using all available memory for coupons on the applet, and subsequently attempting to place more coupons into the applet, its coupons will be prioritised based on FIFO.

This document describes option 2, the full-featured API. Management of secure element memory allocations for coupon and loyalty objects in the value-added services applet will be automatically handled by the VAS manager.

The VAS manager extends the core wallet with couponing and loyalty specific functionality. Therefore, the MNO wallet app and the VAS manager implement an extended wallet as described in the GSMA Mobile Wallet Whitepaper [7].

### 6.3.3 Definition of terms

| Term | Description |
|---|---|
| Core Wallet | A subset of features, protocols, interactions models and interfaces different mobile operators’ mobile wallets should support, as defined in Core Wallet Requirements [2]<br>Equivalent to MNO wallet app for the purposes of this proposal |
| SP-MMI | Any app (Service Provider Man-Machine Interface) residing on the handset that wishes to interface to the wallet (Service Provider Man-Machine Interfaces)<br>Equivalent to service provider app for the purposes of this proposal |
| VAS manager | Service running on the handset implementing the core wallet API – may be part of the MNO wallet app (also referred to as “wallet” in this chapter) |

**Table 33: Definitions of terms in API 3**

### 6.3.4 Use cases

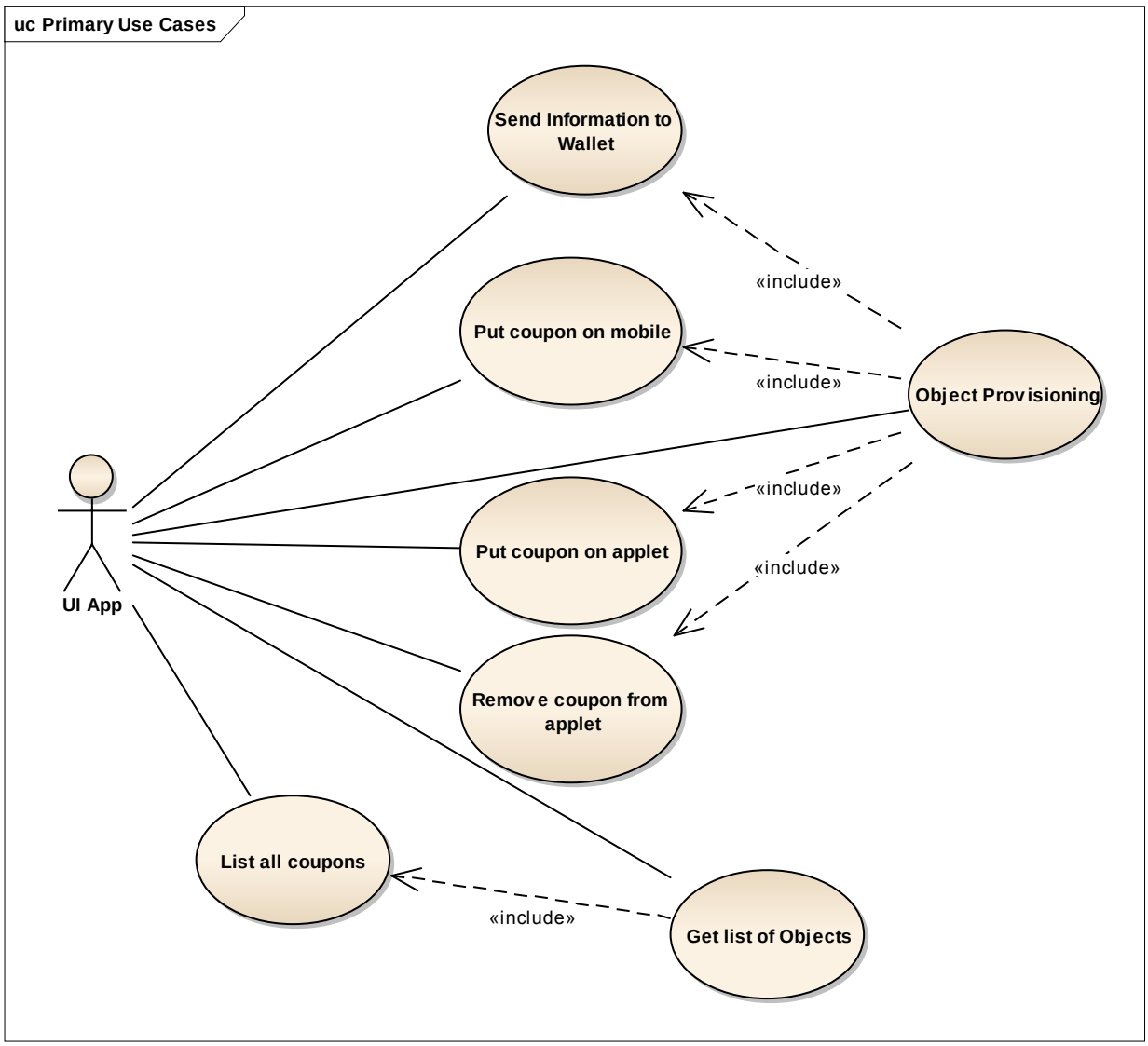


Figure 39**: Coupon handling use case diagram**

Figure 39 is a high-level use case of how a SP-MMI interacts with the core wallet to manage coupons on the value-added service applet. It allows adding and removing coupons from this applet and also retrieval of coupons stored on it for in the SP-MMI.

| Name | Description | Dependencies |
|---|---|---|
| Put coupon on applet | Put a coupon into the value-added services applet on the UICC. | <<include>> Object Provisioning |
| Remove coupon from applet | Remove a coupon from the value-added services applet on the UICC. | <<include>> Object Provisioning |
| List all coupons | List all coupons in the VAS Manager | <<include>> Get List of Objects |

| Name | Description | Dependencies |
|---|---|---|
| Put coupon on mobile | Send the coupon to the mobile phone memory (non-UICC) | `<<include>>` Object Provisioning |
| Object provisioning | Makes light-weight objects provided by the service provider or third-party accessible in the wallet. Makes basic operations on these objects possible without invoking the service provider or third-party app user interface (for example, deletion, activation, redemption). This mechanism should be configurable, within limits, by the service provider or third-party app. Provisioning of coupons and loyalty cards is described in this document, but the mechanism could also apply to other types of objects, for example tickets. | -- |
| Get list of objects | Request a list of objects from the wallet | -- |
| Send information to wallet | Send information messages relevant to the service to the wallet for display and possibly storage in a history list (For example, changes of terms and conditions, special offers, planned service outages). | `<<include>>` Object Provisioning |

**Table 34: Use case descriptions**

### 6.3.5 Application programming interfaces

This section specifies the actual function calls and their payload messages that are exchanged between the wallet and SP-MMIs. These calls are described generically and at a high-level to allow a mapping to different handset operating system-specific inter-app communication mechanisms. Their descriptions aim to be sufficiently detailed so that it is possible to derive mappings from them given a set of rules for a specific platform.

This section is structured as follows: First the API calls are introduced. Then the payload messages are explained in detail with all their attributes.

Note on internationalisation: The API messages detailed below do not contain provisions for transferring alternative internationalised strings or images — this would increase the message sizes too much. It is the responsibility of the communicating apps (wallet, SP-MMIs) to respect the global language settings of the mobile device and send appropriate messages in the correct language only. The `ObjectInfo` data structure, however, contains a way to send multiple language elements to the wallet. Instead, the sending SP-MMI may update existing objects stored in the wallet by issuing update calls (`addObjects` messages with updated language content).

#### 6.3.5.1 API calls

Figure 40 gives an overview of the wallet API calls to be implemented by the wallet app and the user interface API calls to be implemented by the SP-MMIs.

These will be described in detail in the following sections.

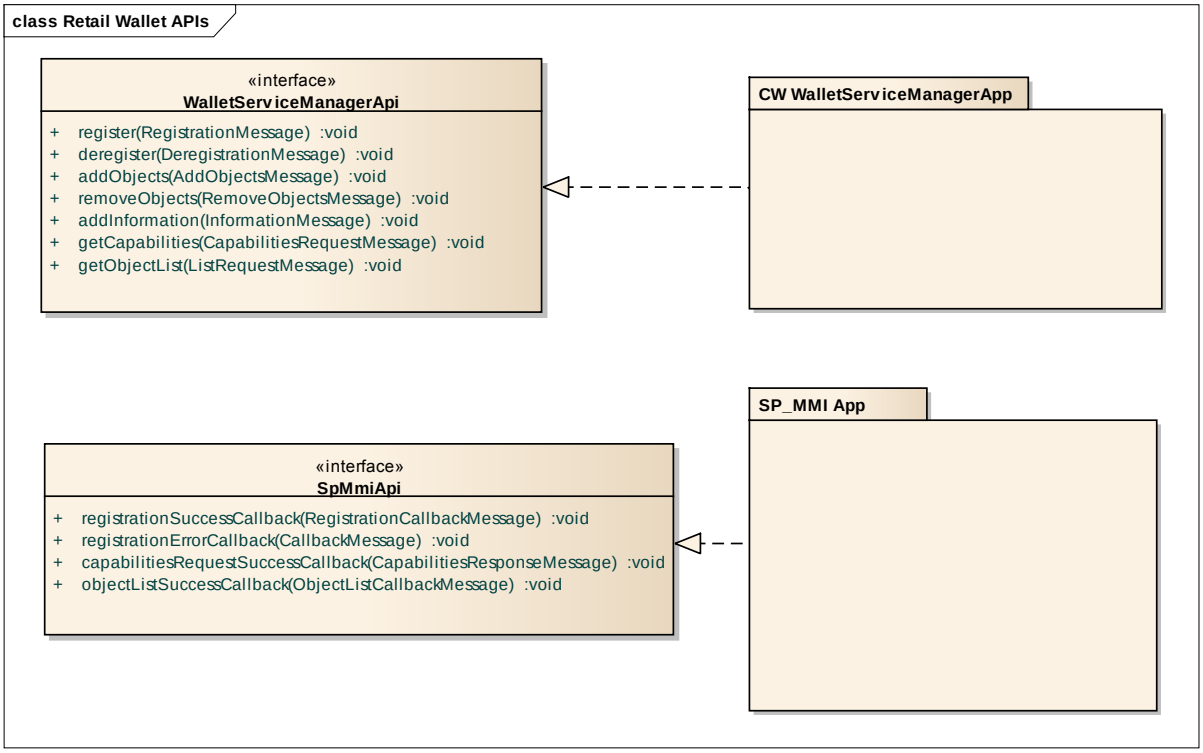


Figure 40**: Application programming interfaces**

##### 6.3.5.1.1 Core wallet (CW) API

Core wallet (CW) API calls are calls from the SP-MMI to the wallet on the handset. The specified calls are listed and described in Table 35.

| Call signature | Description |
|---|---|
| `register(RegistrationMessage):void` | This call is used by the service provider or third-party app to make itself known to the Wallet. The registration call might involve some user interaction on the side of the wallet. The wallet and the service providers or third-party app should cache this connection, so that registration is only necessary once.<br>Registration is optional. Some wallet implementations may process the following API calls without any prior registration. This is not recommended for security reasons, however. |

| Call signature | Description |
|---|---|
| `deregister(DeregistrationMessage):void` | This call is used by the service provider or third-party app to deregister from the Wallet. It might involve some user interaction on the side of the wallet. The wallet and the service providers or third-party app should clear relevant caches of previous connections.<br>Deregistration (as per registration) is optional, though apps which register should also deregister when required. |
| `addObjects(AddObjectsMessage):void` | This call adds lightweight objects to the wallet. The wallet is supposed to display the added objects in their respective categories and allow for user interaction with them if any operations are sent alongside the objects. |
| `removeObjects(RemoveObjectsMessage):void` | This call removes a list of named objects or all of the objects added by the SP-MMI from the wallet. The wallet should not offer the objects to the user any further after they have been removed. |
| `addInformation(InformationMessage):void` | This call instructs the wallet to display specific information in a uniform manner to the user. The `InformationMessage` may be stored by the wallet for future reference by the user. |
| `getCapabilities(CapabilitiesRequestMessage):void` | Asks the wallet for a list of capabilities of the API and the secure element storages maintained by the wallet. |
| `getObjectList(ListRequestMessage):void` | Asks the wallet for a list of objects stored in the wallet (or on a secure element maintained by the wallet). Only those objects that the calling app has access rights to are returned. Access is granted to objects that the particular app put into the wallet or to objects that are not issuer-specific (for example, manufacturer coupons with no particular offer awarder(s) specified). |

**Table 35: Core wallet API calls**

#### 6.3.5.1.2 CW to SP-MMI API

This section describes calls from the wallet to SP-MMIs.

| Call signature | Description |
|---|---|
| `registrationSuccessCallback(RegistrationCallbackMessage):void` | Call-back after successful registration call |
| `registrationErrorCallback(CallbackMessage):void` | Call-back after unsuccessful registration call |
| `capabilitiesRequestSuccessCallback(CapabilitiesResponseMessage): void` | This call returns information about capabilities of the wallet to the calling app. |
| `objectListSuccessCallback(ObjectListCallbackMessage): void` | This call returns the list of objects requested by the calling app. |

**Table 36: SP-MMI API calls**

### 6.3.5.2 API messages

API messages are the payloads of the API calls previously described, and will be explained in more detail in the following sections.

API messages always include the suffix “Message”. Also, there are other auxiliary objects with different endings that are transmitted as parts of messages.

Note that a class notation with the concept of inheritance and aggregation is used in this section to describe the relationship between messages and other payload objects. Inheritance is a mechanism for code reuse, and is when objects of the same class can inherit data. Aggregation means that some objects are named parts of other objects. An additional concept is the array: the familiar array notation “[ ]” means that several or no objects of a specific type are aggregated into another object.

The basic data types in Table 37 are used in the message proposal.

| Call signature | Description |
|---|---|
| `string` | Sequence of characters in UTF-8 encoding |
| `Key` | Representation of a cryptographic key that also be used as a numerical identifier |
| `Image` | Representation of a bitmap image |
| `URI` | Representation of uniform resource identifier (typically a string) |
| `int` | Whole integer number, greater than or equal to zero |
| `DateTime` | Date and time information with second resolution and time zone information |

**Table 37: Basic data types**

The platform-specific mappings of these basic data types may be specified in future handset operating system -platform mapping documents.

#### 6.3.5.2.1 Registration and object API messages

Figure 41 shows an overview of the registration messages.

Registration messages deal with the initial setup of the communication between SP-MMIs and the wallet. The registration process covers three main aspects of wallet and SP-MMI communication:

- It makes SP-MMIs known to the wallet and – most importantly – to the user who should have full control over any apps wanting to communicate with or use functions of the wallet. This interchange might optionally be secured using cryptographic or other security mechanisms.
- It allows SP-MMIs to give additional information that may be used by the wallet to display details of the service to the user. Note that there might be other mechanisms (such as metadata loaded from a wallet backend server) that also supply this detailed information. It is up to the wallet implementation on how to deal with possible conflicting service information from the registration call.
- Additionally, the registration message may convey an initial set of objects with some specific operations to be displayed in the wallet. In the absence of service detail information, these objects might be the only visible artefact of the service in the wallet. Note that objects may be added and removed dynamically at a later time.

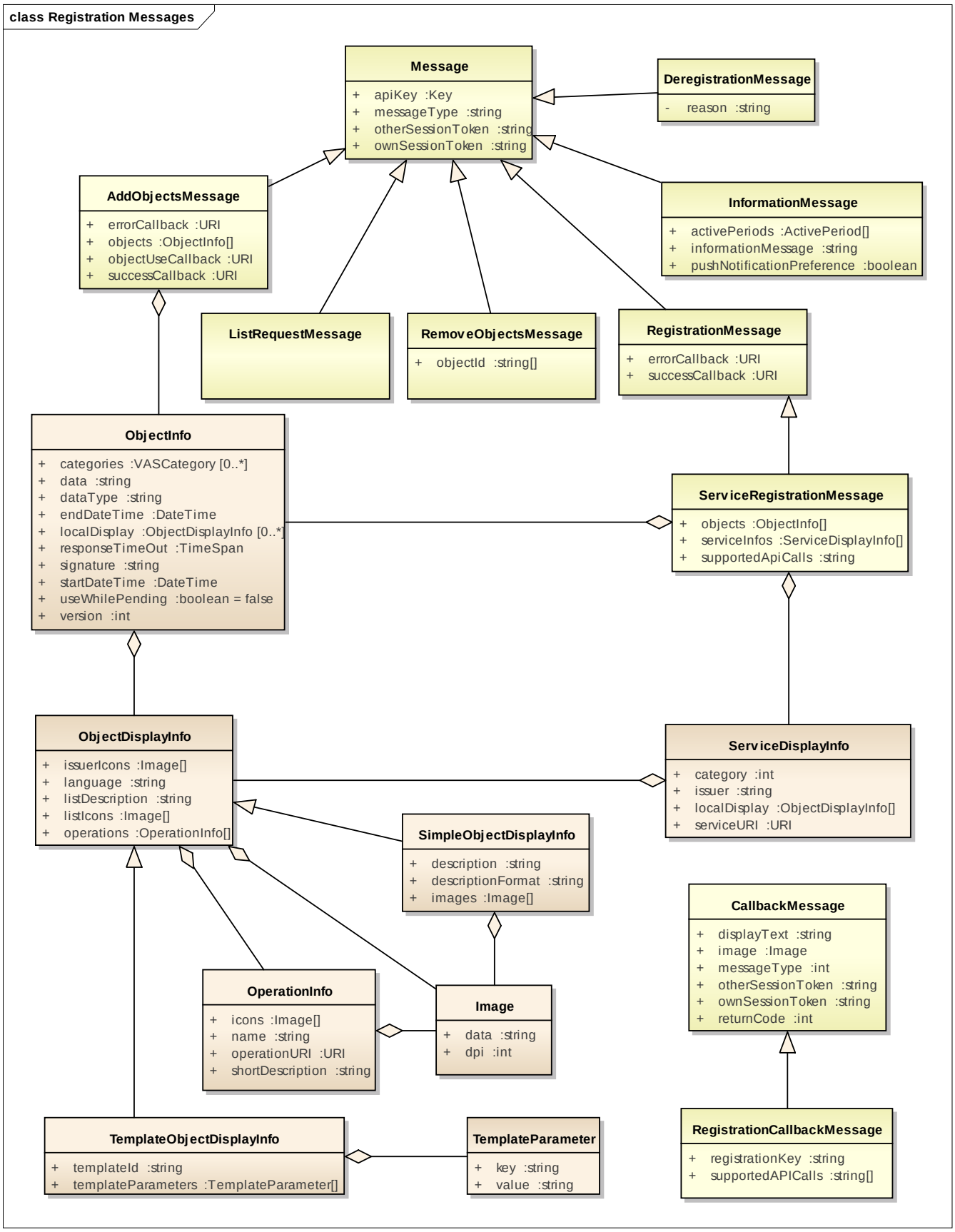


Figure 41**: Registration and object API messages**

#### 6.3.5.2.1.1 `Message` class

`Message` is the base type of all API messages, the data elements with the suffix “Message”.

| Attribute | Description |
|---|---|
| `messageType:string` | String denoting the type of this message. This must be the message name from this proposal, for example `“AddObjectsMessage”`. |
| `apiKey:Key` | This field may be used by the calling service provider or third-party app to transport identifying or authenticating information about the app to the wallet. The wallet might check this information based on business and/or security rules (for example, checking against a white list). |
| `ownSessionToken:string` | This field may be used by the calling app to convey session information to the called app. The called app should copy this information into the `otherSessionToken` field when doing a call-back or some other communication related to the current session. |
| `otherSessionToken:string` | The `otherSessionToken` is the `ownSessionToken` received by the communication partner in a previous API call in the same session. Note: The mechanisms of `ownSessionToken` and `otherSessionToken` may be used by communicating apps to avoid user authentication requirements (such as re-entry of PINs or other login credentials) when switching between apps in the same session. Apps may store timestamps alongside these tokens, to implement session timeouts. |

**Table 38:** `Message` **attributes**

#### 6.3.5.2.1.2 `DeregistrationMessage` class

The `DeregistrationMessage` is used by a service to break its connection to the wallet. It undoes all effects of the `RegistrationMessage`: service and object data structures are deleted, API or registration keys are deleted, and session information such as `ownSessionToken` or `otherSessionToken` is discarded.

| Attribute | Description |
|---|---|
| `reason:string` | This attribute contains the reason for deregistering. It should be displayed, or recorded for later review by the wallet. |

**Table 39:** `DeregistrationMessage` **attributes**

#### 6.3.5.2.1.3 `AddObjectsMessage` class

With the `AddObjectsMessage` a service can dynamically add objects to the wallet for display to the user. These objects augment the objects communicated in the `RegistrationMessage`.

Objects of the same ID are replaced in the wallet, for example, the `AddObjectsMessage` can have the semantics of updating objects already present in the wallet.

| Attribute | Description |
|---|---|
| `objects:ObjectInfo[]` | ObjectInfo data structures for the newly added objects. |
| `successCallback:URI` | Callback URI is invoked by the wallet in case of successful addition of objects. |
| `errorCallback:URI` | Callback URI is invoked by the wallet in case of error while attempting addition of objects. |
| `objectUseCallback:URI` | Callback URI is invoked by the wallet in case of object use e.g. coupon presented to PED |

**Table 40:** `AddObjectsMessage` **attributes**

#### 6.3.5.2.1.4 `RemoveObjectsMessage` class

With `RemoveObjectsMessage`, the service app can remove previously added objects from the wallet. The wallet must delete all internal data related to these objects and stop displaying them to the user. The `RemoveObjectsMessage` can also remove objects added in a `RegistrationMessage`.

| Attribute | Description |
|---|---|
| `objectId:string[]` | List of IDs of the objects to remove. If this list is empty, all objects added by the calling SP-MMI are removed from the wallet. |

**Table 41:** `RemoveObjectsMessage` **attributes**

#### 6.3.5.2.1.5 `RegistrationMessage` class

The `RegistrationMessage` was added as a pure virtual base class of `ServiceRegistrationMessage`. It is the parameter of the `register()` core wallet API call.

#### 6.3.5.2.1.6 `ServiceRegistrationMessage` class

The `ServiceRegistrationMessage` is meant for apps that want to integrate with the wallet. These apps might be service apps that come along with an applet on the UICC, but also could be apps that want to add their objects to the wallet in a dynamic manner, such as couponing apps.

In the case of service apps, there could be an overlap of the information in the `ServiceDisplayInfo` fields and metadata pulled by the wallet on discovery of new applets on the UICC or metadata pushed to the wallet by a backend service. It is up to the wallet implementation to reconcile this possible conflicting information. If the wallet is supposed to load service-specific metadata from a backend system, service apps should still complete the `serviceInfos` fields to allow the wallet to display information to the user even if the backend communication fails or is delayed.

| Attribute | Description |
|---|---|
| `serviceInfos:ServiceDisplayInfo[]` | This attribute contains information about the services added through this registration. Since a specific app might implement many different services all in one, this can be more than one entry. If there is no `ServiceDisplayInfo` entry, the wallet must either rely on some backend system to provide this info, or the service is only displayed by means of the objects it controls (see below). |
| `objects:ObjectInfo[]` | This is list of objects to be initially added to the wallet. This list might be augmented or reduced by subsequent `addObjects()` and `removeObjects()` calls. This list can be empty if no additional objects are to be displayed in the wallet. |
| `successCallback:URI` | Callback URI is invoked by the wallet in case of successful registration. |
| `errorCallback:URI` | Callback URI is invoked by the wallet in case of registration error. |
| `supportedApiCalls:string[]` | List of the API calls supported by the service on the CW to SP-MMI interface. |

**Table 42:** `ServiceRegistrationMessage` **attributes**

#### 6.3.5.2.1.7 `CapabilitiesRequestMessage` class

The `CapabilitiesRequestMessage` can be used by SP-MMI to inquire about capabilities of the Wallet and its associated secure elements.

| Attribute | Description |
|---|---|
| `apiVersion:string` | Version of the API supported by the SP-MMI. |
| `successCallback:URI` | Call-back URI is invoked by the wallet to return the capabilities information. |

**Table 43:** `CapabilitiesRequestMessage` **attributes**

#### 6.3.5.2.1.8 `ListRequestMessage` class

The `ListRequestMessage` can be used by SP-MMI to request a list of objects from the wallet. Only those objects will be returned that have been put by the SP-MMI into the wallet (that is, no objects controlled by other SP-MMIs can be retrieved that way). An exception to this rule is objects that are meant to be shared between SP-MMIs (for example, manufacturer coupons with no particular offer awarder(s) specified).

#### 6.3.5.2.1.9 `InformationMessage` class

The `InformationMessage` data structure describes information messages relevant to the service that the SP-MMI may send to the wallet for display to the user e.g. notification of changes of terms and conditions, special offers or planned service outages. The wallet may store these messages in a history. `AddObjects` must be used to update relevant objects with specific changes in addition to providing alerts advising the user via `addInformationMessage`, where applicable.

User preferences or specific wallet implementations may result in `informationMessage` being displayed to the user as a push notification or the message may be made available to the user e.g. upon initial access of the wallet UI, via a messages wallet UI view or via a wallet UI view specific to the SP-MMI. `pushNotificationPreference` informs the wallet of the SP-MMI preference, but does not guarantee that preference is upheld.

An `ActivePeriod` is implemented to ensure the message is only delivered and/or available while relevant. SP-MMIs should ensure that the `informationMessage` is appropriate to the localisation of the wallet and SP-MMI. The user may dismiss any `informationMessage` within the wallet UI, such that it will no longer be visible to the user.

| Attribute | Description |
|---|---|
| `informationMessage:string` | Message for display to the user in appropriate localised format. |

| Attribute | Description |
|---|---|
| `pushNotificationPreference:boolean` | Where `true`, the wallet is advised that the SP-MMI preference is for the notification to be pushed to the user. Depending on user preferences and wallet implementations, push notifications may not be delivered to the user. Where there are one or more `activePeriods`, the push notifications would be delivered at each `startDateTime` |
| `activePeriods:ActivePeriod[]` | Active periods for which the message should be displayed to the user. Where delivered by push notification, the message is delivered to the user at each `startDateTime`. The message is available to view in the wallet between `startDateTime` and `endDateTime`. Where omitted, the message is considered permanently available and where delivered by push notification, delivered immediately. The wallet UI may still archive the message, hiding it from the user, after some undefined period of time, according to either wallet implementation or user preferences. |

**Table 44:** `InformationMessage` **attributes**

#### 6.3.5.2.1.10 `ObjectInfo` class

The `ObjectInfo` data structure describes objects to be added to the wallet. This is the same as the data structure first described in Figure 9. Each object can have a list of operations associated with it that the user might invoke from within the wallet.

| Attribute | Description |
|---|---|
| `version:int` | Version of this `ObjectInfo` structure. The version number shall be `0` in all structures conforming to this proposal. |
| `localDisplay:ObjectDisplayInfo[]` | This field contains all information needed by the wallet to display a meaningful representation of the object to the user in different local languages. |
| `categories:VASCategory[]` | Categories of the object. See Enumerations (Figure 14) for more detail Multi-purpose objects may belong to several categories. |
| `endDateTime:DateTime` | The last time that the object is valid and usable (for example, expiry of a coupon offers). It is the responsibility of the wallet to handle this condition: the wallet might silently delete the object from its internal memory or mark the object as expired. |

| Attribute | Description |
|---|---|
| `data:string` | Specific objects might be handled directly by the extended wallet. Depending on the object type the wallet might make some objects directly available via NFC.<br>In this case the data is a byte string with an object-type specific meaning (for example, `VASCouponInfo` or `VASLoyaltyInfo` etc.).<br>For objects not to be handled by the wallet, this attribute can be empty. |
| `dataType:string` | Type of the data attribute of the `ObjectInfo`. This field shall contain a mime type, if available. |
| `signature:string` | If the `ObjectInfo` structure is used in stand-alone mode (outside of the context of `AddObjectsMessage`) a signature may be included to authenticate the object for the wallet. |
| `useWhilePending:boolean` | This property indicates that the VAS item can still be used during the time period between a previous transaction and confirmation of that transaction. |
| `responseTimeOut:TimeSpan` | The count of how long to wait until a response or it will be assumed that a response will not come back. |
| `startDateTime:DateTime` | The date that the item will start to be active. |

**Table 45: `ObjectInfo` attributes**

#### 6.3.5.2.1.11 `OperationInfo` class

The `OperationInfo` describes an operation on an object transmitted to the wallet. Operations on objects should be presented by the wallet in a way that is suitable for the mobile handset platform (for example, a drop-down menu) and the wallet app design style.

| Attribute | Description |
|---|---|
| `name:string` | Operation name used to invoke a wallet request. |
| `icons:Image[]` | Icons to display alongside the `shortDescription` (may be omitted if no icon is available). Several icons in different resolutions may be included to target terminal devices with different display resolution or pixels per inch (PPI). |
| `shortDescription:string` | Short description of the operation to be shown to the user, such as in a menu. The description should be formatted as UTF-8. |
| `operationURI:string` | URI to be invoked once the operation is selected by the user. |

**Table 46: OperationInfo attributes**

#### 6.3.5.2.1.12 `ObjectDisplayInfo` class

The `ObjectDisplayInfo` structure contains all the information necessary for a user-friendly presentation of objects in the wallet, including the internal handling of objects.

| Attribute | Description |
|---|---|
| `language:string` | Language code for the language used in this `ObjectDisplayInfo` data structure for all texts and graphics presented to the user. The language code should follow ISO 639-1 [14]. |
| `listIcons:Image[]` | Icons to display alongside the `listDescription` (may be omitted if no icon is available). Several icons in different resolutions may be included to target terminal devices with different PPI. |
| `issuerIcons:Image[]` | Additional icons to display alongside the `listDescription` (may be omitted if no icon is available). Several icons in different resolutions may be included to target terminal devices with different PPI. These icons should be used to visualise the issuer of the object to the user. |
| `listDescription:string` | Short description of the object to be shown to the user in the context of a list of objects, for example in a menu. The format shall be UTF-8. |
| `operations:OperationInfo[]` | List of operations associated with the object. If this is empty the object can only be viewed by the user in the wallet. This might still make sense if the object contains information such as a one or two dimensional barcode. |

**Table 47:** `ObjectDisplayInfo` **attributes**

#### 6.3.5.2.1.13 `SimpleObjectDisplayInfo` class

The `SimpleObjectDisplayInfo` structure describes a self-contained simple object that can be displayed by every wallet without additional context information. The `SimpleObjectDisplayInfo` structure contains all attributes of the `ObjectDisplayInfo`.

| Attribute | Description |
|---|---|
| `images:Image[]` | Large-scale images to be shown alongside the `longDescription` text to visualise the object to the user. Different resolutions may be included for different device displays. |
| `description:string` | Description of the object that explains the object to the user. It is up the wallet implementation how to display this information if needed by the user. |
| `descriptionFormat:string` | Format of the description string as a mime type (for example, `"text/html"`). |

**Table 48:** `SimpleObjectDisplayInfo` **attributes**

#### 6.3.5.2.1.14 `TemplateObjectDisplayInfo` class

The `TemplateObjectDisplayInfo` structure is meant to describe pre-defined, structured objects for domain-specific applications. Placeholders in the template – the "keys" – are instantiated with the values to render a specific object. Several templates for specific application domains will be defined in a separate document at a later stage.

| Attribute | Description |
|---|---|
| `templateId:string` | String-formatted identifier of the template to use for this object. |
| `templateParameters:TemplateParameter[]` | List of key/value to be used to instantiate a pre-defined template for rendering by the wallet. |

**Table 49:** `TemplateObjectDisplayInfo` **attributes**

#### 6.3.5.2.1.15 `TemplateParameter` class

The `TemplateParameter` structure is an auxiliary structure for the `TemplateObjectDisplayInfo` structure that defines a key/value pair.

| Attribute | Description |
|---|---|
| `key:string` | Key of the `TemplateParameter`. |
| `value:string` | Value of the `TemplateParameter`. |

**Table 50:** `TemplateParameter` **attributes**

#### 6.3.5.2.1.16 `Image` class

The `Image` structure contains an image along with information on the targeted screen resolution in pixels (pixels per inch). The wallet is expected to evaluate this information and choose a suitable resolution image for a given screen real estate and screen physical resolution. The exact method of this selection process is up to the individual wallet implementation.

| Attribute | Description |
|---|---|
| `dpi:int` | Targeted screen resolution of the image in dots per inch (dpi). |
| `data:string` | Image data. The image data format must contain information about the image size in pixels. |

**Table 51:** `Image` **attributes**

#### 6.3.5.2.1.17 `ServiceDisplayInfo` class

The `ServiceDisplayInfo` is used to characterise a service in the wallet. It re-uses the `ObjectDisplayInfo` data structure to carry localised service descriptions to be presented to the user.

| Attribute | Description |
|---|---|
| `category:int` | Type of the service. These service types are pre-defined:<br>• `0`: Undefined<br>• `1`: Payment<br>• `2`: Couponing<br>• `3`: Loyalty |
| `issuer:string` | Name of the issuer. This information might be presented to the user by the wallet. |
| `serviceURI:URI` | URI to identify the service in detail. The URI format was chosen so that service names can be unique without needing a central service name registry. The URI should resolve a web page detailing specifics of the service when invoked in a browser. |
| `localDisplayInfo:ObjectDisplayInfo[]` | Localised service descriptions to be presented to the user |

**Table 52:** `ServiceDisplayInfo` **attributes**

#### 6.3.5.2.2 Callback messages

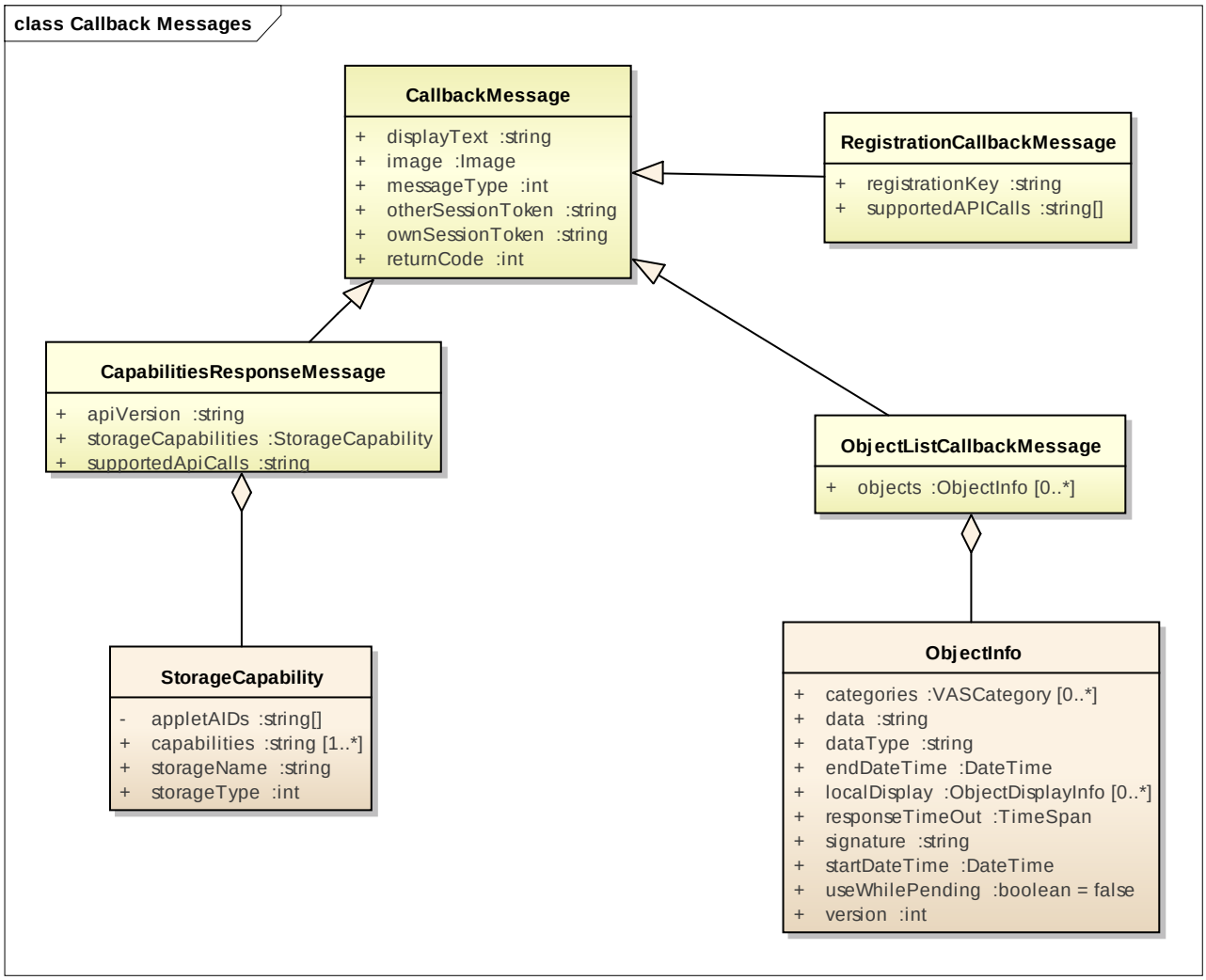


Figure 42: **Callback messages**

Callback messages are messages with references to call-back URLs that require the UI control flow go back to the calling SP-MMI, and transport important information back to the calling app.

##### 6.3.5.2.2.1 `CallbackMessage` class

The `CallbackMessage` is the base type for all other call-back messages described in this section.

| Attribute | Description |
|---|---|
| `messageType:int` | The codes for the `messageType`. |
| `returnCode:int` | Return code of the call-back. This return code is defined:<br>• `0`: no error |
| `displayText:string` | Text to display to the user when receiving a call-back. |
| `image:Image` | Image to present to the user when receiving a call-back. |

| Attribute | Description |
|---|---|
| `ownSessionToken:string` | This field may be used by the calling app to convey session information to the called app. The called app should copy this information into the `otherSessionToken` field when doing a call-back or some other communication related to the current session. |
| `otherSessionToken:string` | The `otherSessionToken` is the `ownSessionToken` received by the communication partner in a previous API call in the same session. Note: The mechanism of `ownSessionToken` and `otherSessionToken` may be used by communicating apps to avoid user authentication requirements (such as re-entry of PINs or other login credentials) when switching between apps in the same session. Apps may store timestamps alongside these tokens to implement session timeouts. |

**Table 53:** `CallbackMessage` **attributes**

#### 6.3.5.2.2.2 `RegistrationCallbackMessage` class

The `RegistrationCallbackMessage` carries additional information needed for a successful return from a `register()` call.

| Attribute | Description |
|---|---|
| `registrationKey:string` | This might be a cryptographic key for subsequent communication with the wallet. This parameter is optional and dependent on the security capabilities of the handset operating system and the specific mapping of the core wallet API for the given platform. |
| `supportedApiCalls:string[]` | This is a list of all the core wallet API calls supported by the platform. The list must at least contain the entry "`register`" – otherwise there can be no `registrationSuccessCallback()` |

**Table 54:** `RegistrationCallbackMessage` **attributes**

Note that the calling SP-MMI must support all call-back calls specified.

#### 6.3.5.2.2.3 `CapabilitiesResponseMessage` class

The `CapabilitiesResponseMessage` returns information about the wallet's capabilities back to the SP-MMI.

| Attribute | Description |
|---|---|
| `supportedApiCalls:string[]` | Returns a list of API calls supported by the Wallet. |
| `apiVersion:string` | Version number of the API |
| `storageCapabilities:StorageCapability[]` | List of the different secure element storage capabilities available. |

**Table 55:** `CapabilitiesResponseMessage` **attributes**

Note that the calling SP-MMI must support all call-back calls specified.

#### 6.3.5.2.2.4 `StorageCapability` class

The `StorageCapability` object contains secure element storage capability details.

| Attribute | Description |
|---|---|
| `storageName:string` | Name of the storage |
| `storageType:int` | Type of storage (to be defined later) |
| `capabilities:string[]` | List of storage capabilities (to be defined later) |
| `appletAIDs:string[]` | List of AIDs (hexadecimal strings without spaces or other special characters) of applets installed in the secure element. |

**Table 56:** `StorageCapability` **attributes**

#### 6.3.5.2.2.5 `ObjectListCallbackMessage` class

The `ObjectListCallbackMessage` returns the requested objects in response to a `ListRequestMessage`.

| Attribute | Description |
|---|---|
| `objects:ObjectInfo[]` | List of objects that match the selection criteria in the `ListRequestMessage` |

**Table 57:** `ObjectListCallbackMessage` **attributes**

## 6.4 API 4 – MNO services to the MNO wallet app

### 6.4.1 Scope

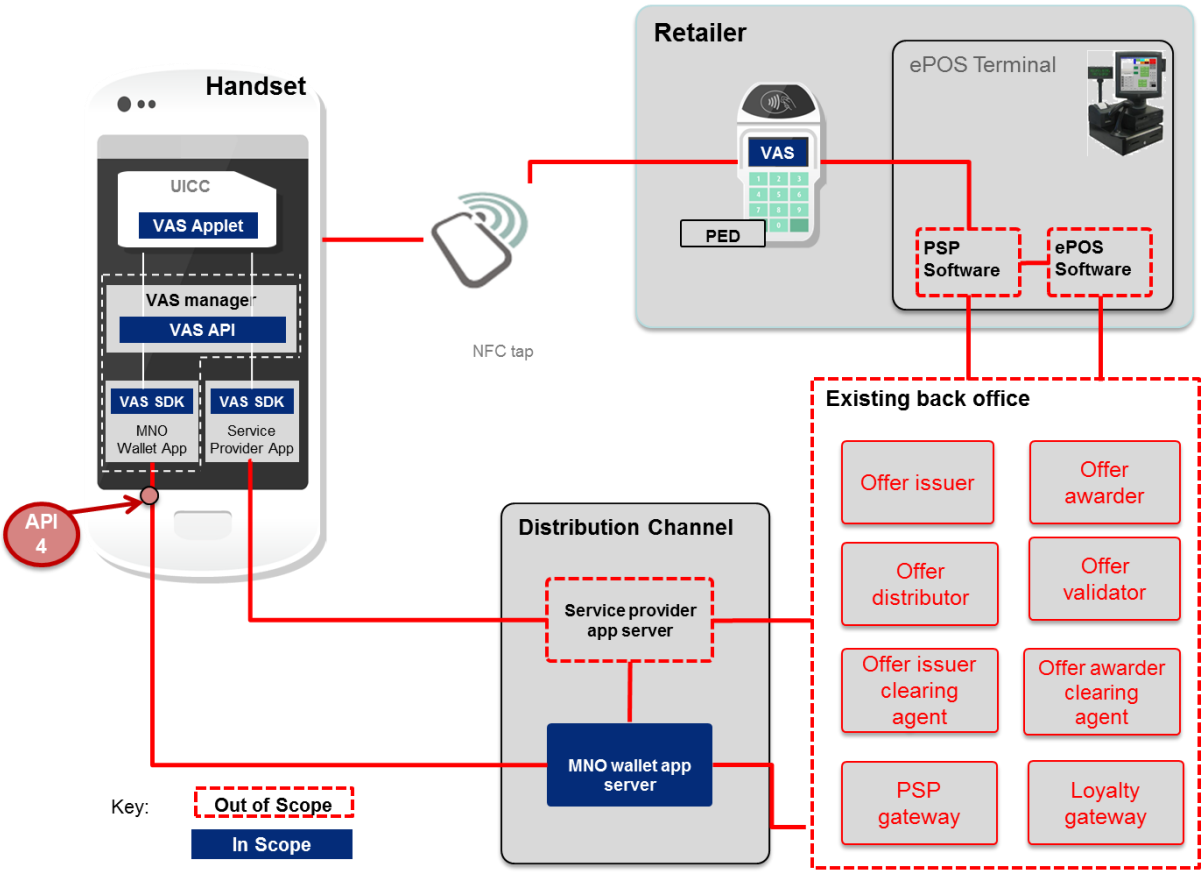


Figure 43**: Location of API 4 within the proposed architecture**

API 4 defines an interface for the MNO wallet app server to integrate with the MNO wallet app. From an MNO wallet app server perspective, the intention is that merchant coupons, FMCG coupons and loyalty scheme data can be pushed to a consumer's handset seamlessly in the background. The MNO wallet app server will have direct access to the MNO wallet app to update the state as a result of interaction between the UICC and the NFC PED.

In addition, there will also be provision for updating application access records. The maintenance of app access records (access control information for service provider apps access to the VAS manager – not related to VAS applet permissions) and related objects is facilitated by the MNO wallet app server.

Depending on the implementation of this API, it may work in a push/pull fashion, as opposed to strictly push. This means that the MNO wallet app server may simply notify the wallet app of the events such as new coupons and updated loyalty balance, which will in turn trigger a call to API 5 to request the remaining data.

Considering the nature and proposed function of this API it could be viewed as an optional component. If not implemented, the MNO wallet app would be responsible for synchronising its data store with the MNO wallet app server. Manual synchronisation would mean polling API 5 for updates.

### 6.4.2 Use

This API can be implemented as a background service or incorporated into the MNO wallet app and triggered via the mobile operating system push notification system. Running as a service allows for real time management of wallet data by the MNO wallet app server. The advantage of real time updates is that upon wallet start up there will be significantly less processing between the MNO wallet app server and MNO wallet app, allowing for a better user experience.

### 6.4.3 Actors

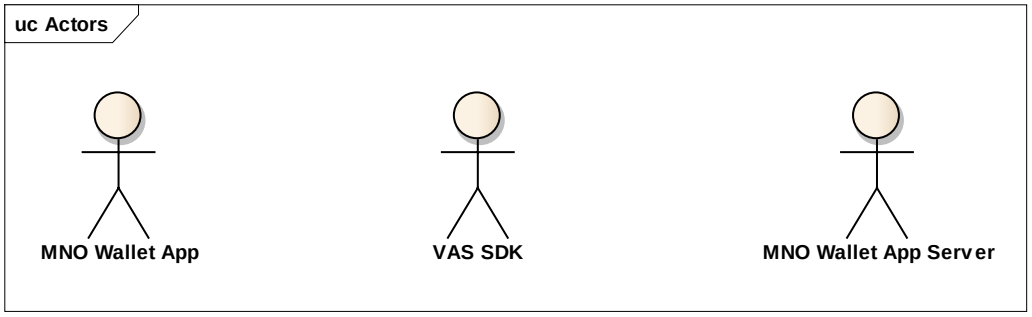


Figure 44**: API 4 actors**

| Actor | Description |
|---|---|
| MNO Wallet App | Consumer facing mobile application with coupon and loyalty presentation ability. |
| MNO Wallet App Server | MNO-based management suite offering coupon, loyalty, account management etc. |
| VAS SDK | Background component to manage access between apps and VAS applet on MNO UICC. |

**Table 58: Actors and descriptions**

### 6.4.4 Use cases

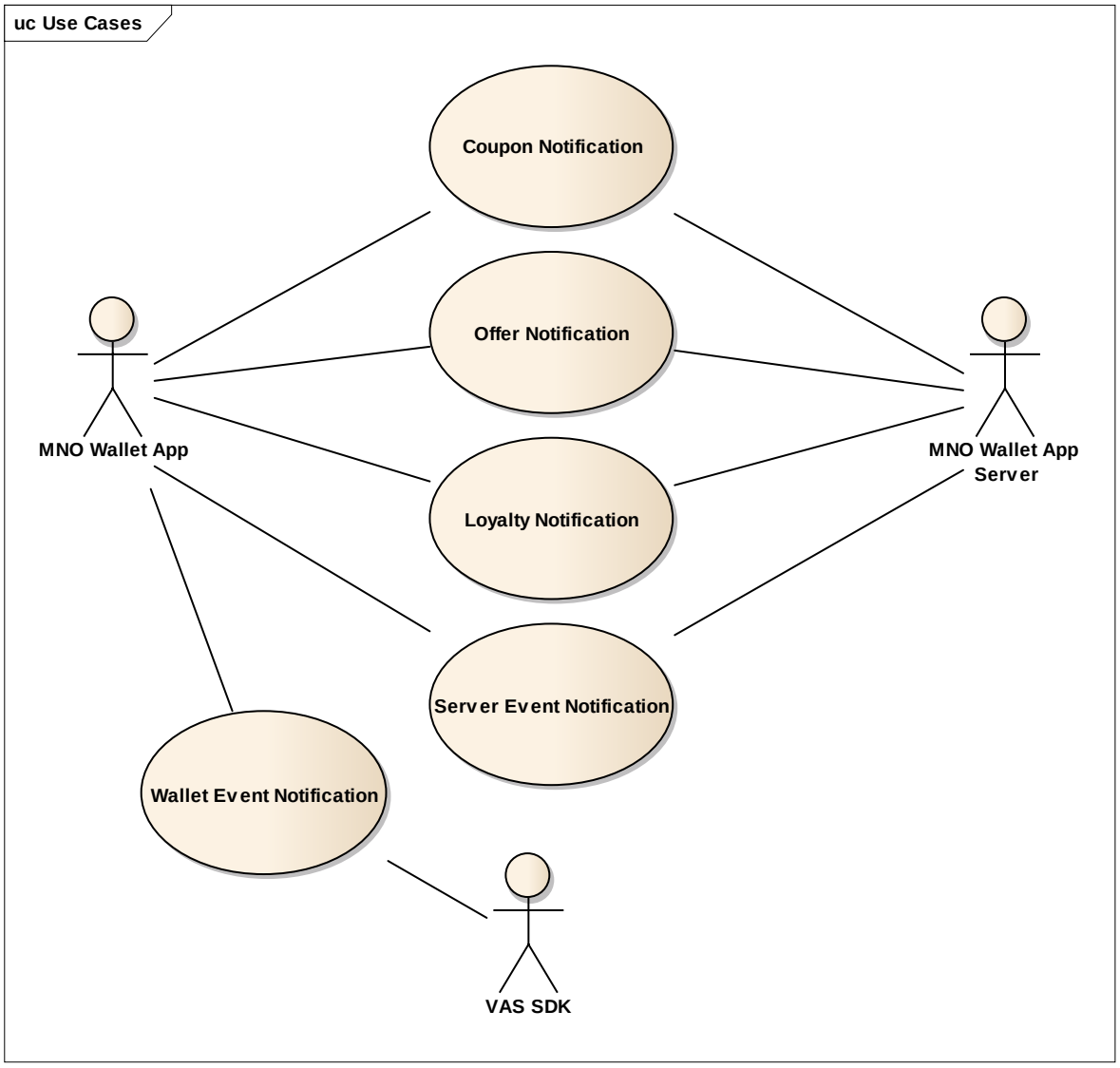


Figure 45**: Use cases for API 4**

| Use case | Description |
|---|---|
| Coupon Notification | Notifications to MNO wallet app of coupon created, updated and deleted events |
| Offer Notification | Notifications to MNO wallet app of offer created, updated and deleted events |
| Loyalty Notification | Notifications to MNO wallet app of loyalty card created, deleted and updated/deleted events |
| Wallet Event Notification | Notification of events completed in wallet at applet/value added services SDK level. For example, VAS SDK can notify MNO wallet of coupon deleted from applet or coupon presented to pin-entry device. |
| Server Event Notification | Notification of events completed on the server side, other than coupon, offer or loyalty notifications. |

**Table 59: Use cases and descriptions**

#### 6.4.4.1 Use case analysis

| Use case | Description |
|---|---|
| Notification of offer created/updated | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Update wallet app that one or many offers have been created and/or updated. The MNO wallet app should then query API 5, if necessary. Persistence of offer at app level once received. |
| Notification of offer deleted | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** MNO wallet app server to update mobile app that specified offer is to be deleted from local store. At this stage it is likely the coupon has been retracted/ deleted at MNO/service provider level and is no longer valid. |
| Notification of loyalty card created/updated | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Instruction to MNO wallet app that one or many loyalty cards have been updated and/or created. MNO wallet app should then query API 5, if necessary. Persistence of data at app level once received. |
| Notification of loyalty card deleted | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Instruction from MNO wallet app server that the MNO wallet app is to delete specified loyalty cards. |
| Notification of coupon created/updated | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Instruction to MNO wallet app that one or many coupons are new or updated and are ready. MNO wallet app should then query API 5, if necessary. |
| Notification of coupon deleted | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Instruction from MNO wallet app server that the MNO wallet app is to delete specified coupon. |
| Wallet action completed | **Source:** VAS SDK<br>**Target:** MNO wallet app<br>**Method:** IPC<br>**Description:** As a result of a request made to the VAS SDK, the MNO wallet app is to be notified that an action has been completed. Examples include coupon deleted from applet, coupon added to the applet and coupon presented to PIN entry device. |

| Use case | Description |
|---|---|
| Notification of application access record created | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Update wallet app that one or many app access records have been created and/or updated. The MNO wallet app should then query API 5, if necessary. Persistence of data at app level once received. |
| Notification of application access record deleted | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** Instruction from MNO wallet app server that MNO wallet app is to delete specified application access records. |
| Server action completed | **Source:** MNO wallet app server<br>**Target:** MNO wallet app<br>**Method:** PUSH<br>**Description:** As a result of activity on the wallet app server, the MNO wallet app might need notification of these events to preserve consistency. This case covers notifications that do not align with the previous notification messages mentioned. |

**Table 60: Use case analysis**

### 6.4.5 Data structure

| Name | Description |
|---|---|
| `ApplicationAccessRecord` | Represent access control information for service provider apps access to the VAS manager. These permissions are not related to VAS applet permissions. |
| `PushMessageType` | Enumerable type specifying the type of message sent to API 4:<br>• `Sync` - Only ID sent. Must query API 5 for remaining data.<br>• `Message` - Complete object sent, all necessary data contained in message. No need to query API 5. |

**Table 61: Data structure**

### 6.4.6 Messages and parameters

#### 6.4.6.1 Push

Table 62 describes the calls available to the MNO wallet app server via push notifications or background service (web socket).

| Method | Parameters | Description |
|---|---|---|
| `itemsCreated` | `P0: itemType:VASCategory`<br>`P1: type:PushMessageType`<br>`P2: itemIdList:string[]`<br>`P3: itemList:ObjectInfo[]` | Notification of new items. `P0` specifies item type as in section 4.2.1 (`categories`). `P1` specifies the message type, sync (`Sync`) or complete object (`Message`). `P2` contains a list of item IDs used for sync operations with `P3` containing a list of complete objects. `itemList` and/or `itemIDList` contents must reflect `VASCategory` |
| `itemsDeleted` | `P0: itemType:VASCategory`<br>`P1: itemIdList:string[]` | Notification of items deleted. `P0` specifies item type as in section 4.2.1 (`categories`). `P1` specifies list of item IDs that have been deleted. `itemIDList` contents must reflect `VASCategory` |
| `eventNotification` | `P0: event:Event` | Notification that an event has occurred at the MNO wallet app server side. Provides an opportunity for the MNO wallet app to react. `P0` represents the event that has occurred. This document does not define a structural definition for this event object. |

**Table 62: Push message**

#### 6.4.6.2 IPC – Inter-process communication

This section covers the calls available to the MNO wallet app server via the IPC interface. These functions are “callback” methods called from the MNO wallet app server once registered using the methods described in section 6.3.5.1.1 core wallet API.

| Method | Description |
|---|---|
| `registrationSuccessCallback(RegistrationCallbackMessage):void` | Call-back after successful registration call. |
| `registrationErrorCallback(CallbackMessage):void` | Call-back after unsuccessful registration call. |
| `capabilitiesRequestSuccessCallback(CapabilitiesResponseMessage): void` | This call returns information about capabilities of the wallet to the calling app. |

| Method | Description |
|---|---|
| `objectListSuccessCallback(ObjectListCallbackMessage): void` | This call returns the list of objects requested by the calling app. |

**Table 63: Inter-process communication**

## 6.5 API 5 – Interface to MNO wallet app server

### 6.5.1 Scope

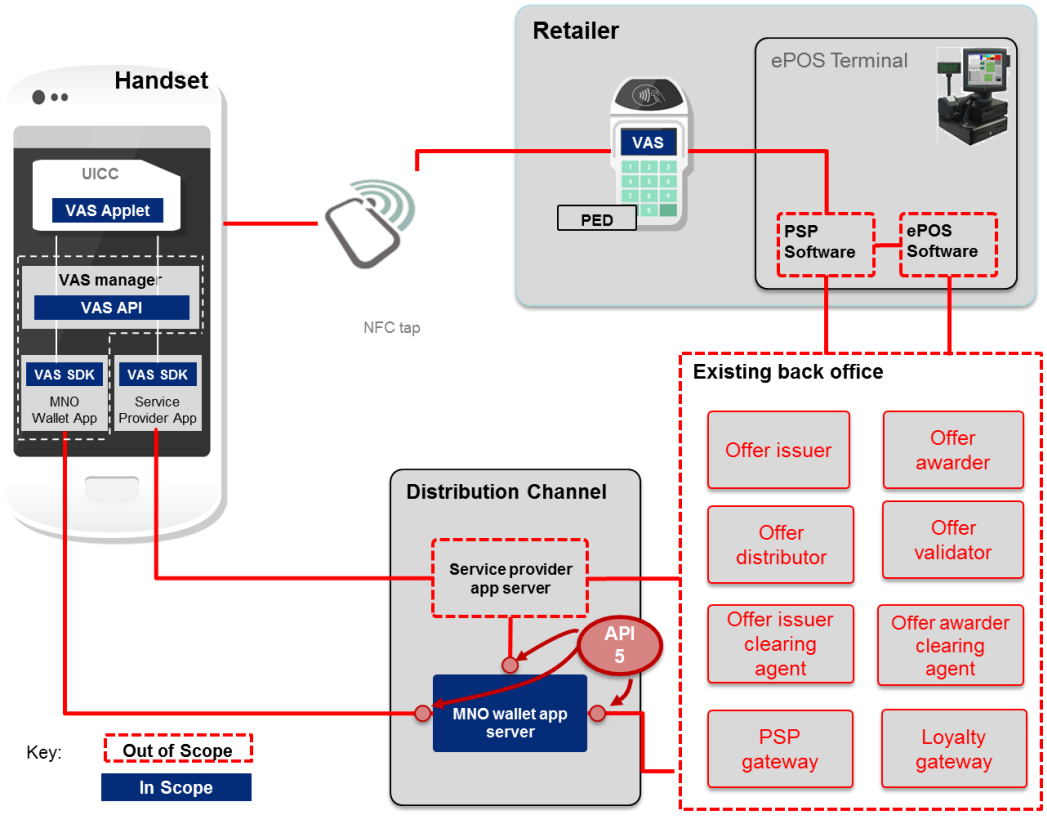


Figure 46**: Location of API 5 within the proposed architecture**

API 5 defines the interface of the MNO wallet app server, which implements the workflows and business logic necessary for third-party services to access and interact with the MNO wallet app. Interaction through API 5 enables merchants, manufacturers, FMCG organisations and third-party application developers expose compatible loyalty mechanics through the MNO wallet app.

The scope of API 5 supports interaction between the MNO wallet app server and external services that wish to access the MNO wallet on the mobile device. These services may include, but are not limited to, service provider app servers, merchant systems, loyalty service provider systems and other associated services that may need to access the MNO wallet app server data.

The MNO wallet app server acts as a gateway to the mobile device application, ensuring appropriate security/access control and that only MNO approved content reaches the mobile application. The interface to the application is specified in API 4.

API 5 defines the interface presented by the MNO wallet app server. The interfaces of third-party services (service provider app servers or loyalty service providers) are out of scope and subject to proprietary third-party services.

Note: The MNO wallet app server will need to support integration with potentially multiple third-party services and support of their unique APIs.

The API documented in this chapter defines the generic interface capabilities of the MNO wallet app server. In practical deployments, this interface will be further partitioned and each outward facing interface subject to specialised implementation requirements; for example, isolating the mobile facing interface to support appropriate authentication and perimeter network security.

### 6.5.2 Actors

Individual actors can hold both source and target roles in individual use cases, with relationship direction defined more clearly on a case-by-case basis. Subsequent use cases represent the set of actors either:

- **VAS provider:** represents the system components that supply coupons or loyalty service functionality to the solution. This could, for example, be an offer distributor, a stand-alone loyalty service provider or part of a payment service provider.
- **VAS consumer:** represents the system components that consume the functionality of the MNO wallet app server. These components would typically be offer issuer and awarders as well as the MNO wallet app itself.

Table 64 identifies the relevant API 5 actors drawn from the larger, global set of system actors defined in Figure 8.

| Generalised Actor | Example Specialisation | Description |
|---|---|---|
| **VAS consumer** | MNO wallet app | Consumer facing mobile application that is used for display of offers, elicitation of coupons, rendering of rich media content and interaction with the applet via API 3. |
| | Service provider app server | Third-party wallet service wishing to integrate with the MNO wallet app server. |
| | Offer Issuer | Merchant integration with MNO wallet app server. |
| **VAS provider** | Offer Validator | Provider of promotion engine and loyalty logic to facilitate coupon and loyalty mechanics. |

**Table 64: Key actor descriptions**

### 6.5.3 Use cases

The MNO wallet app server interface supports four high level classes of operations. These are defined by the use cases illustrated in Figure 47:

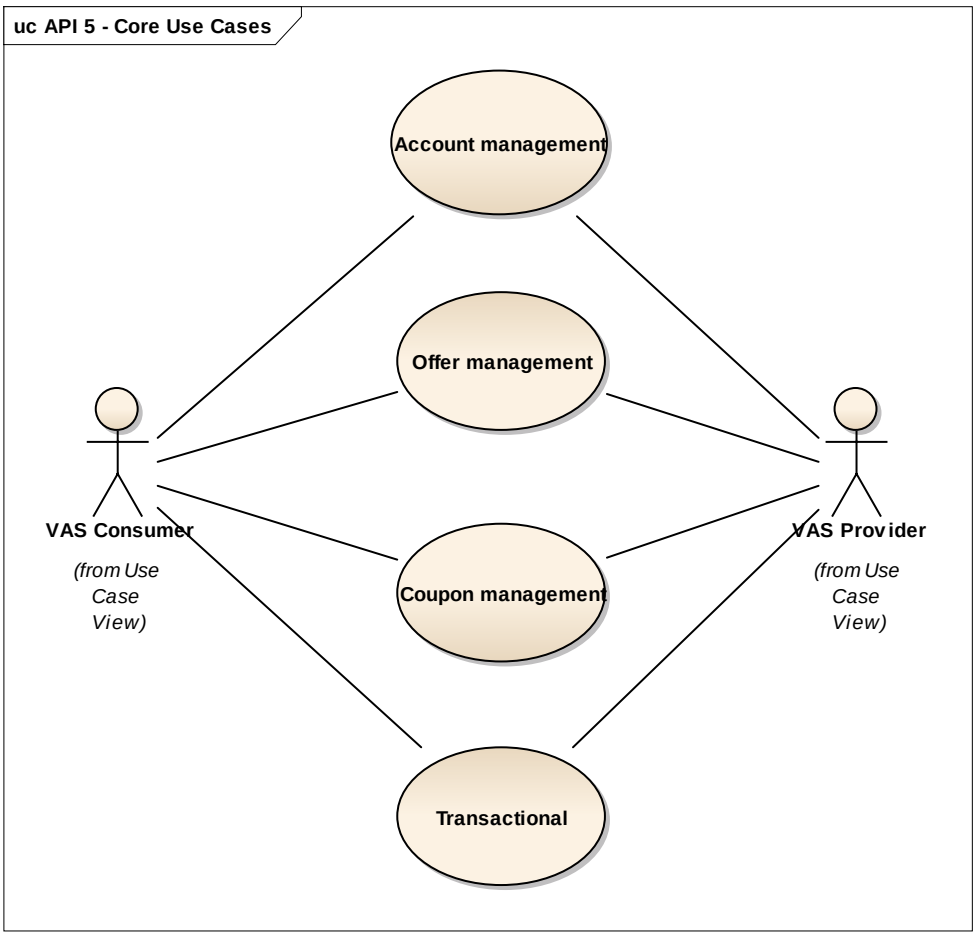


Figure 47**: API 5 high level use cases**

The set of use cases enables the MNO wallet app server to support desired functionality within the wallet app and provides an interface to external services that consume, publish and manage coupon and loyalty data:

- **Account management**: create, update and query MNO wallet and VAS Provider account data
- **Offer management**: enable VAS Consumers and VAS Providers to manage and query offers
- **Coupon management**: enable VAS Consumers and VAS Providers to manage and query coupons
- **Transactional**: support informative communication between the MNO wallet application and VAS Consumers and VAS Providers.

Each parent use case is composed of sub-use cases that comprise the core capabilities of the MNO wallet app server interface.

#### 6.5.3.1 VAS account management

The account management use cases enable the MNO wallet app server to create and manage account attributes such as personal data, contact preferences and associated loyalty cards.

Facilities are provided for the consuming services to query and display all relevant account information as well as update via multiple channels; including the consumer facing MNO mobile application and customer or self-service portals.

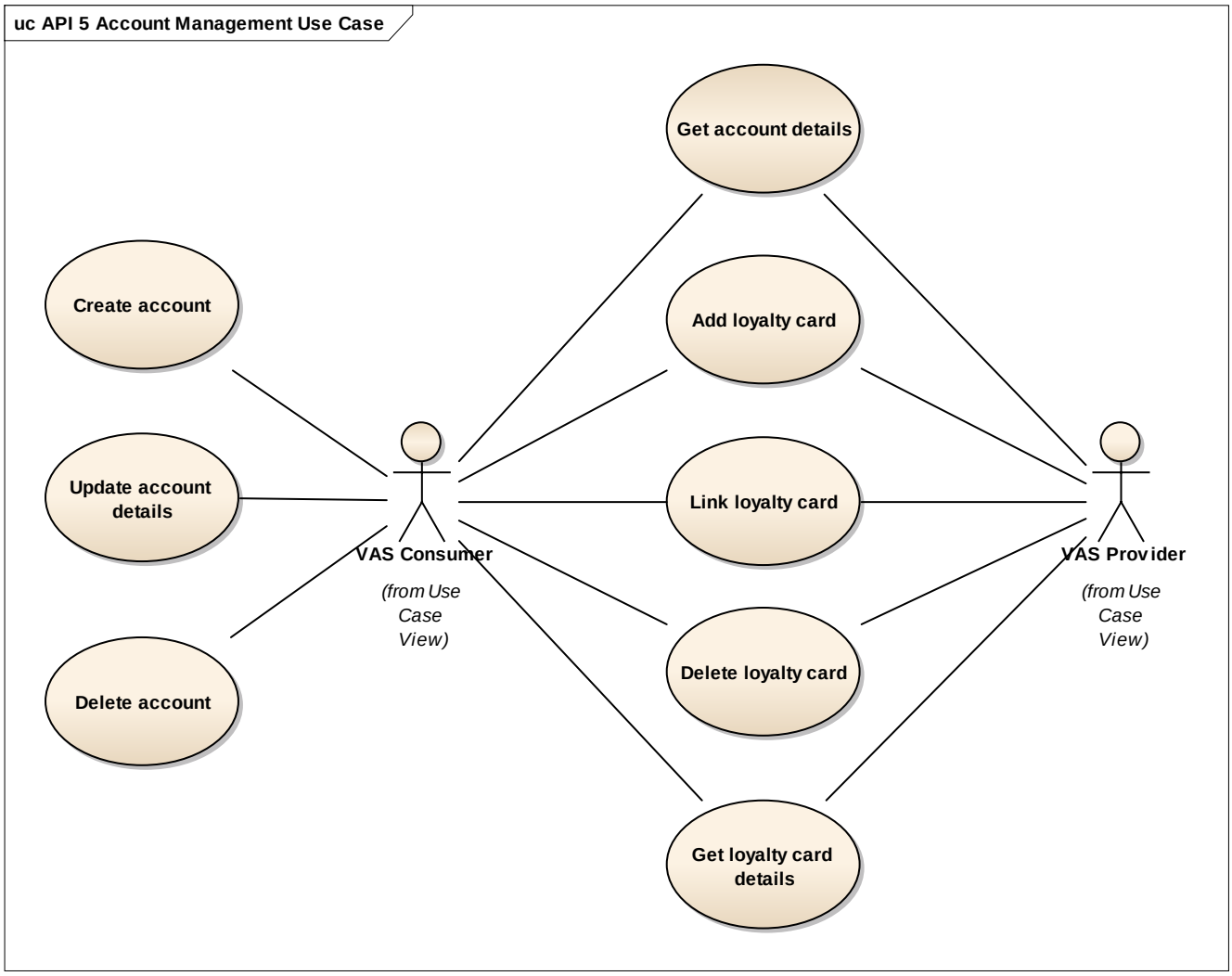


Figure 48**: Account management use cases**

| Use Case | Description |
|---|---|
| **Create account** | **Source**: VAS consumer<br>**Target**: MNO wallet app server<br>**Method**: Web-service call<br>**Description**: Creates a new MNO wallet user account, passing initialisation details. Used by mobile application, consumer facing portal or merchant to create a new account in the wallet service. Must support password/authentication service as well as ability to register initial account attributes. |
| **Update account details** | **Source**: VAS consumer<br>**Target**: MNO wallet app server<br>**Method**: Web-service call<br>**Description**: Allows a caller to update some or all attributes of an account. The source of such request may be the MNO wallet application or some other VAS consumer, such as an MNO self-service portal. |
| **Delete account** | **Source**: VAS consumer<br>**Target**: MNO wallet app server<br>**Method**: Web-service call<br>**Description**: Remove a nominated MNO wallet user account and all associated loyalty cards, offers and coupons |
| **Get account details** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Enables caller to request specific details of a MNO wallet user account for analysis, display and further processing. For example, querying an account to determine loyalty cards, issued coupons or personal details to display within a customer service portal. |
| **Add loyalty card** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Add a loyalty card for a specific scheme to the MNO wallet user account. This will initiate the creation of a new loyalty account with the identified VAS provider. Specific implementation will be dependent on MNO capabilities, VAS provider capabilities, and the technical requirements and third-party business relationships of individual MNO implementations. This use case is out of scope for this proposal |
| **Link loyalty card** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Add a pre-existing scheme loyalty card to the MNO wallet app. This does not create a new entry in the merchant/service provider system, but instead adds a reference to some existing loyalty identifier (card). Interaction may be necessary with third-party loyalty services to validate account identity and authority to link the requested scheme card. |

| Use Case | Description |
|---|---|
| **Delete loyalty card** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Remove a specified loyalty card from a merchant/loyalty service provider account. This may be implemented as a hard or soft delete and, in the latter case, be considered an unlink operation. |
| **Get loyalty card details** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Retrieves full details of a specific loyalty card, including textual and graphical content. |

**Table 65: Account management use case descriptions**

#### 6.5.3.2 Offer management

The use cases for offer management show the core capabilities of MNOs to support the distribution of existing offers.

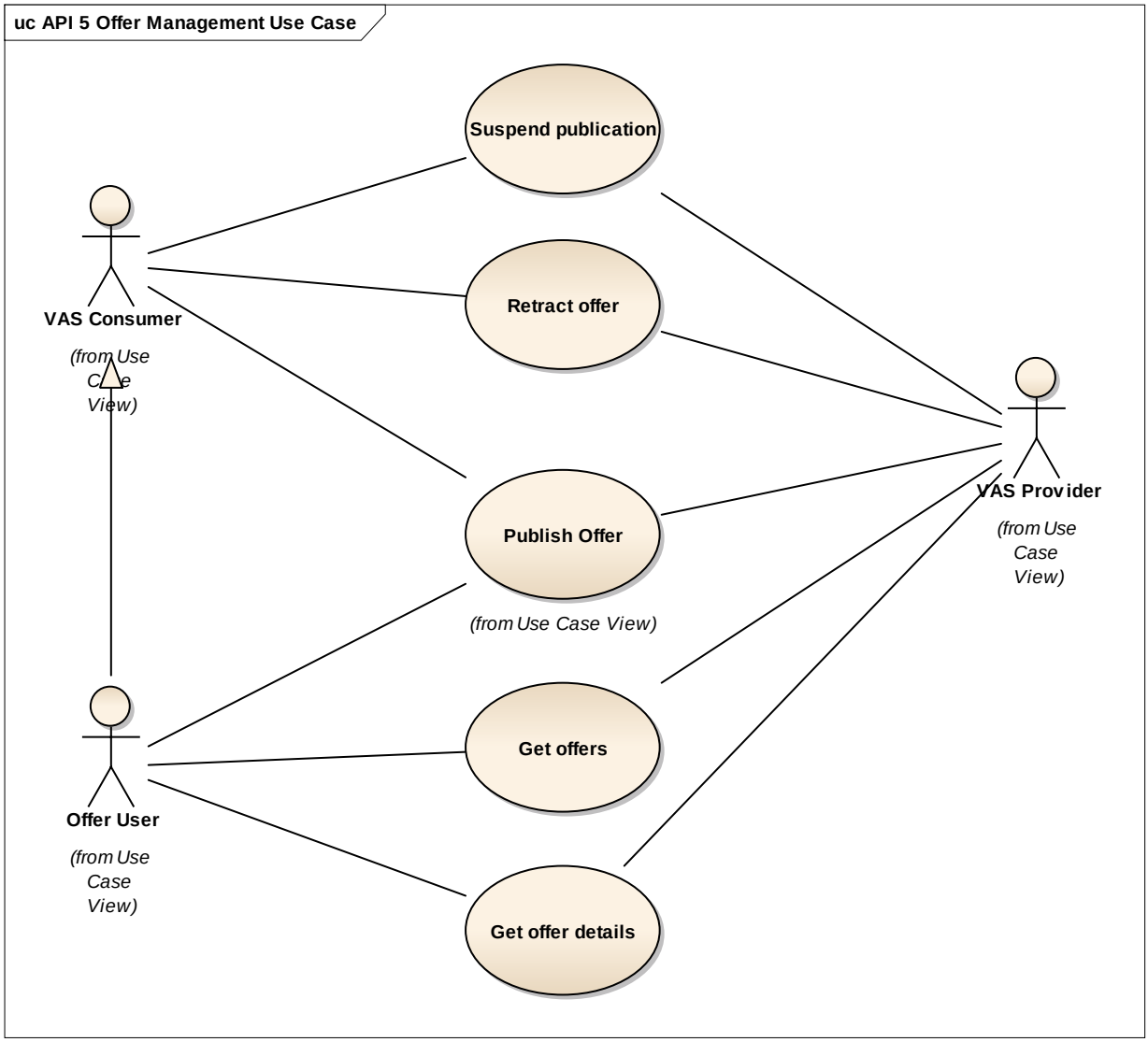


Figure 49**: Offer management use cases**

**Note:** The “create offer” use case is considered out of scope of this document. The process of creating an offer either through offline manual configuration or a program interface is considered an enabler for all subsequent use cases.

| Use case | Description |
|---|---|
| **Publish offer** | **Source**: VAS consumer, VAS provider<br><br>**Target**: Offer user, MNO wallet app<br><br>**Description**: An interface conforming to the GS1 “publish offer” use case [4] and enabling a created offer to be distributed to a number of wallet users. Should allow specification of target account(s), account groups, and by segmentation rules.<br><br>May be initiated by either an MNO wallet server (explicitly pushing an offer to a consumer) or service provider (some condition in a customer account, for example a transaction trigger) causing a new offer to become available to a specific account. |
| **Suspend publication** | **Source**: VAS consumer, VAS provider<br><br>**Target**: MNO wallet app<br><br>**Description**: Temporary suspension preventing further distribution of a published offer. For example, in the scenario that new members join a loyalty scheme and would previously have been eligible for an offer. Will leave offers published on those devices to which they have already been distributed and allow coupon acquisition to continue. |
| **Retract offer** | **Source**: VAS consumer, VAS provider<br><br>**Target**: MNO wallet app<br><br>**Description**: Will retract published offers from selected accounts. Should allow specification of target account(s), account groups and by segmentation rules. Will allow any acquired coupons requested under the now retracted offer to continue to be redeemed. |
| **Get offers** | **Source**: Offer User, VAS consumer<br><br>**Target**: VAS provider<br><br>**Description**: Get a list of all offers that are available to a specified MNO wallet account. Intention is to return high-level data, enough to display offer in tabular form on chosen channel e.g. summary view on mobile device or ecommerce/customer portal. |
| **Get offer details** | **Source**: Offer user, VAS consumer<br><br>**Target**: VAS provider<br><br>**Description**: Retrieves full details of a specific offer, including textual and graphical content. The service provider may provide an interface to elicit offer details, alternatively these could be defined during offer creation phase (out of scope), which would need the MNO wallet app server to accept complete offer details when created (definition). |

**Table 66: Offer management use case description**

#### 6.5.3.3 Coupon management

The coupon management use cases define the core MNO wallet app server operations that manage coupons derived from distributed offers. A coupon is to be considered the concrete instance of an abstract offer. A coupon can be presented at the point-of-sale to affect some changes to a sub-total or trigger some other reward.

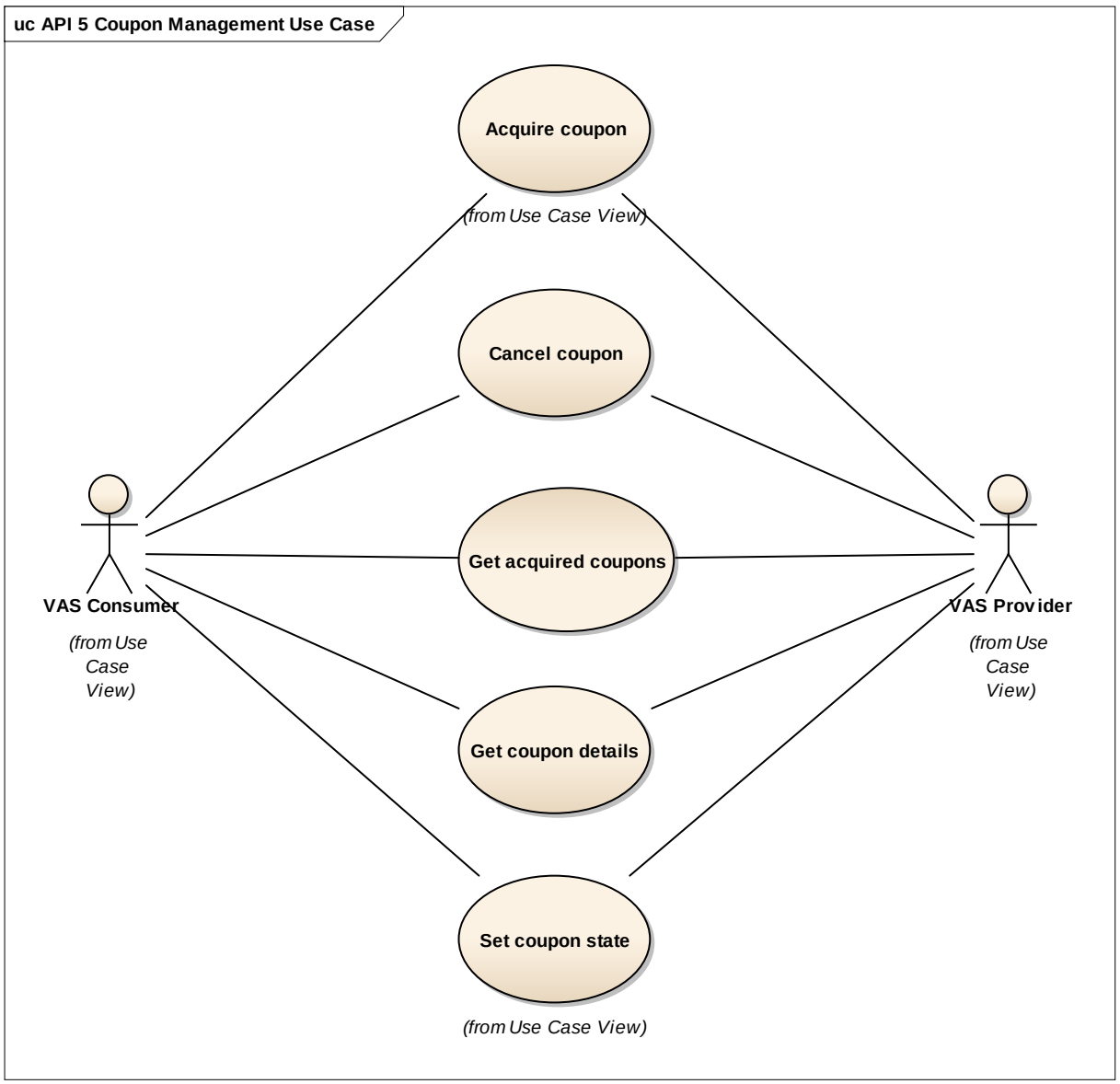


Figure 50**: Coupon management use cases**

The coupon management use cases define a high-level set of capabilities to manage the coupon lifecycle as defined in earlier sections of this document; spanning issuance, query, state control and removal.

| Use Case | Description |
|---|---|
| **Acquire coupon** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Used to create a unique coupon identifier for a specific offer. The MNO wallet app server shall invoke the proprietary loyalty service provider interface (with the offer ID stored during offer creation) to elicit a concrete coupon for the specific offer (potentially issued against a nominated account). Acquire coupon should allow a coupon to be issued in a specified state (issued, activated, other). |
| **Cancel coupon** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Used to delete an acquired coupon and reinstate any value to an account that was consumed during coupon acquisition. |
| **Get acquired coupons** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Returns a list of all acquired coupons that are available to a specified account. Intention is to return high-level data that is enough to display coupons in tabular form. |
| **Get coupon details** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Retrieves full details of a specific coupon, including textual and graphical content. The VAS Provider may provide an interface to elicit coupon details, alternatively these may be defined during offer creation phase (out of scope), which would need the MNO wallet app server to accept complete offer details during offer creation. |
| **Set coupon state** | **Source**: VAS consumer<br>**Target**: VAS provider<br>**Method**: Web-service call<br>**Description**: Used to activate or deactivate a coupon previously acquired. This will be used in the case where only a loyalty account ID is passed during POS redemption and the POS can retrieve issued coupons in a specified state (active, for example) and apply those to the transaction. |

**Table 67: Coupon management use case descriptions**

#### 6.5.3.4 Transactional

The transactional use cases represent MNO wallet app server functionality that enables communication between entities within the system.

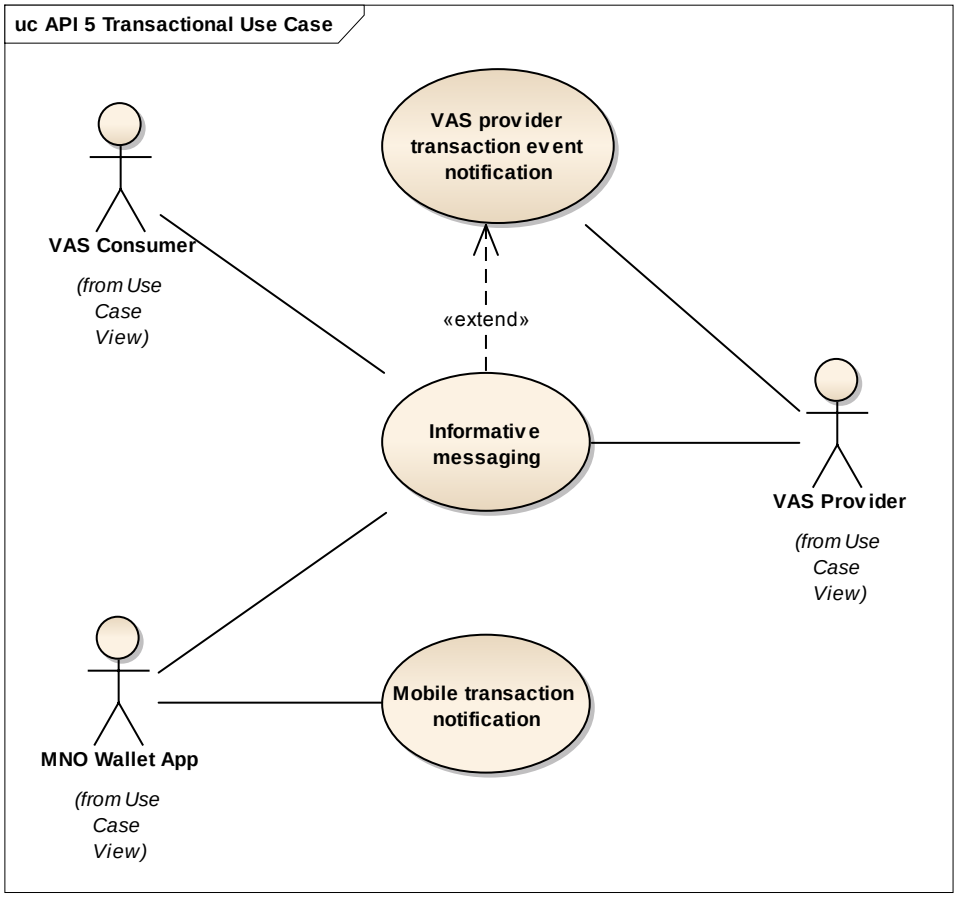


Figure 51**: Transactional use cases**

| Use case | Description |
|---|---|
| **VAS provider transaction event notification** | **Source**: VAS provider<br>**Target**: MNO wallet app<br>**Method**: Web-service call<br>**Description**: provides a mechanism allowing a VAS Provider to notify the MNO wallet app server that an event has occurred that affected the account of a scheme member. This allows real-time updates to be applied to the mobile application following e.g. loyalty service provider interaction. |

| Use case | Description |
|---|---|
| **Informative messaging** | **Source**: VAS Provider, VAS Consumer<br>**Target**: MNO wallet app<br>**Method**: Web-service call<br>**Description**: Provides a mechanism allowing the server side components of the system to transfer informative, rich media, messaging to the MNO wallet app such as geo-triggered notifications, receipts or service information. This differs from the transaction event notification, which originates from the service provider to notify of specific coupon or loyalty mechanic operations. |
| **Mobile transaction notification** | **Source**: MNO wallet app<br>**Target**: MNO wallet app server<br>**Method**: Web-service call<br>**Description**: Provides a mechanism by which the mobile device can convey information to the MNO wallet app server that some operation has been performed (consumer use statistics, geo-location information or some information about coupons that have been viewed or sent via NFC to a contactless terminal). The MNO wallet app server may use this information to inform next-best offer/coupon issuance or trigger some other event or messaging to other components. |

**Table 68: Use case descriptions**

### 6.5.4 Data structures

The following sub-sections explore the types of mandatory data elements that transit the system components. These types will be exchanged between source and destination actors and form the objects exchanged by the MNO wallet app server system interfaces.

#### 6.5.4.1 `requesterID` and `responderID`

These data elements are derived from a common base, `commonID`, which is used to identify an individual service endpoint that is initiating a request to/from API 5. In addition to the identification of a calling end-point, relevant security data should be passed to confirm that the caller and caller end-point has the rights to access the called service.

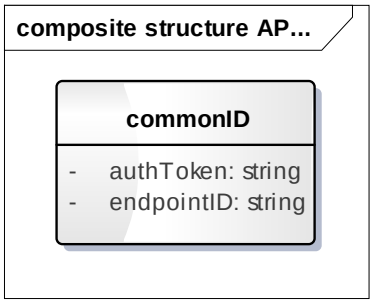


Figure 52: **API 5** `requesterID`/`responderID` **data structure**

- `endpointID` identifies the calling endpoint within the enterprise; differentiates source of request and is used to perform any role specific access control.
- `authToken` is a security mechanism to validate authenticity of the caller and ensure content of the message is not manipulated

#### 6.5.4.2 `transactionStatus` class

This is used to convey the result of a request to the calling service, supporting both a response code and optionally a response message that may be displayed at the caller to provide informative output.

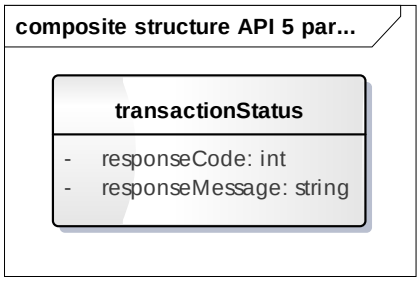


Figure 53: **API 5** `transactionStatus` **data structure**

- `responseCode` contains the success code of a requested operation. Response codes are MNO service and implementation specific.
- `responseMessage` is an optional textual message from service accompanying response code.

#### 6.5.4.3 `accountInfo` class

Represents the complete summary of a customer's account. This composite is also used to update and return account information based on appropriate requests.

This structure may be overloaded in MNO implementations to add implementation specific attributes e.g. necessary customer data to satisfy customer on-boarding requirements. However, the mandatory attributes defined below represent:

32. The unique account ID within the MNO wallet service and
33. The associated individual loyalty accounts.

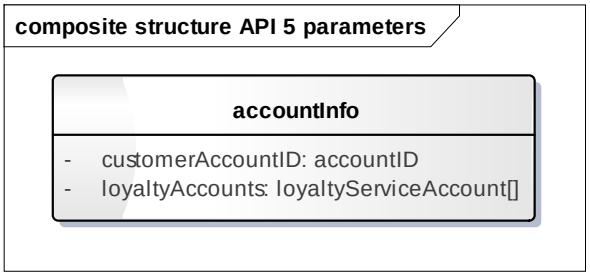


Figure 54**:API 5** `accountInfo` **data structure**

`In Section 6.6, a walletUserID` is used to uniquely identify an individual customer within the MNO service. The `accountID` defined here is mapped to this `walletUserID`.

#### 6.5.4.4 `offerID, couponID` and `customerAccountAlternateID`

These types were previously specified as part of the `VASOfferInfo` and `VASCouponInfo` structures defined in section 4.

- `offerID` represents a unique offer code that applies to a specific offer created by the Issuer and to be published by the distributer. The MNO wallet app server must ensure this ID is unique to each offer across the platform.
- `couponID` is resolved from an `OfferID` as part of the Acquire Coupon use case. The code is defined as a simple string to allow for free-form coupon code identifiers. The MNO wallet app server must ensure this ID is unique to each coupon across the platform (though not necessarily each instance).
- `customerAccountAlternateID` is a unique identifier for the loyalty card belonging to the user. The MNO wallet app server must ensure this ID is unique to each loyalty card product across the platform.

#### 6.5.4.5 `eventInfo` and `eventType`

`eventInfo` encapsulates data being transferred in notification exchanges. The enumeration `eventType` defines the mandatory notification operations that MNOs must support. These operations enable informative, asynchronous events to be dispatched from the MNO wallet app and loyalty service provider to initiate some action by the MNO wallet app server.

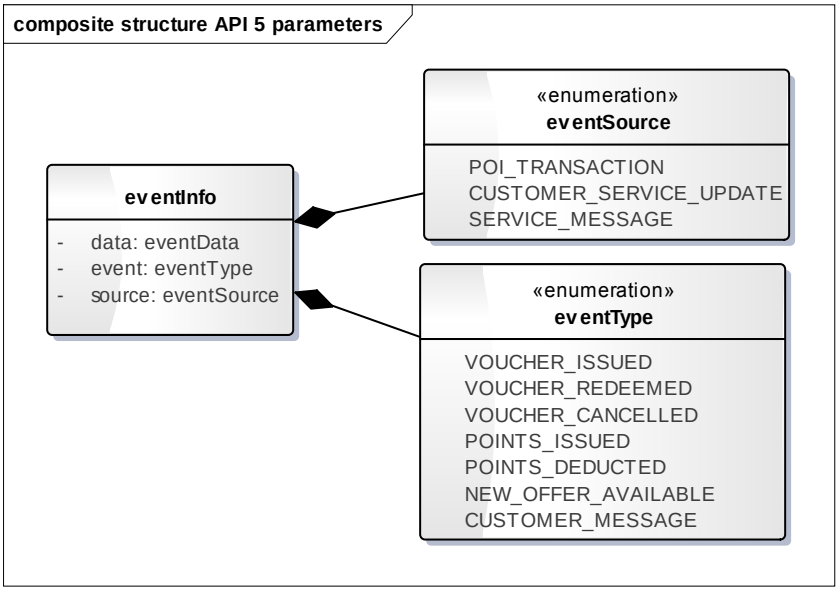


Figure 55: **API 5** `eventInfo`**/**`eventType` **data structure**

An `eventInfo` has a `source`, an `event` and associated `data`. The `source` identifies the event stimulus, the `event` identifies the event triggered, and the event specific information is contained within the `data` field.

| Enumeration | Description |
|---|---|
| `TRANSACTION_IND` | Notification to the MNO wallet app server that some transaction has been performed that has potentially affected the loyalty items stored within a consumer's merchant-specific loyalty service account. The MNO wallet server may act on this information, such as by querying the loyalty service or issuing some update to the MNO wallet app. The `eventInfo` data field contains the loyalty service. |
| `CUSTOMER_SERVICE_UPDATE` | Notification to the MNO wallet app server about a customer action. This update can be about customer accepting/rejecting coupons, their browsing behaviour, geo-location or other related status. |
| `SERVICE_MESSAGE` | Notifies that the calling entity has dispatched some service specific message. This may be stored/processed at the MNO wallet app server or displayed as appropriate at the MNO wallet app. |

**Table 69: Enumerations for** `eventSource`

### 6.5.5 Messages and parameters

This section talks about the various categories of messages, the methods used and related parameters.

#### 6.5.5.1 Interface methods and parameters

| Category | Method | Parameters |
|---|---|---|
| Account management | createAccount | REQUEST P1: requesterID<br>REQUEST P2: accountInfo<br><br>RESPONSE P1: responderID<br>RESPONSE P2: walletUserID<br>RESPONSE P3: transactionStatus |
| | updateAccount | |
| | getAccountDetails | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: accountInfo<br>RESPONSE P3: transactionStatus |
| | deleteAccount | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | addLoyaltyCard | REQUEST P1: requesterID<br>REQUEST P2: VASLoyaltyInfo<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | linkLoyaltyCard | |
| | deleteLoyaltyCard | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: loyaltyCardID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | getLoyaltyCardDetails | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: loyaltyCardID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus<br>RESPONSE P3: VASLoyaltyInfo |
| Offer management | publishOffer | REQUEST P1: requesterID<br>REQUEST P2: walletUserID[]<br>REQUEST P3: VASOfferInfo<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |

| | | |
|---|---|---|
| | suspendOffer | REQUEST P1: requesterID<br>REQUEST P2: walletUserID[]<br>REQUEST P3: offerID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | retractOffer | REQUEST P1: requesterID<br>REQUEST P2: walletUserID[]<br>REQUEST P3: offerID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | getPublishedOffers | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: offerID[] |
| | getOfferDetails | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: offerID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus<br>RESPONSE P3: VASOfferInfo |

| | | |
|---|---|---|
| Coupon management | acquireCoupon | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: offerID<br>REQUEST P4: numberOfCoupons (int)<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus<br>RESPONSE P3: VASCouponInfo[] |
| | activateCoupon | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: couponID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | deactivateCoupon | |
| | cancelCoupon | |
| | getAcquiredCoupons | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus<br>RESPONSE P3: couponID[] |
| | getCouponDetails | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: couponID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus<br>RESPONSE P3: VASCouponInfo |
| Transactional | eventNTF | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: eventInfo[] |
| | informativeMessageNTF | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: eventInfo[]<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus |
| | mobileTransactionNTF | REQUEST P1: requesterID<br>REQUEST P2: walletUserID<br>REQUEST P3: eventInfo[] |

**Table 70: Summary of API 5 service interface**

#### 6.5.5.2 Message sequences

The previous sections have explored the use cases and service interfaces to be considered mandatory to all MNOs that wish to comply with this specification. In this section, each service interface is explored and individual message sequence flows are elaborated and explained in more detail.

#### 6.5.5.2.1 Account management message sequences

The account management use cases enable the MNO wallet app server, or affiliated third-party service, to create and manage loyalty and promotion accounts that can be exposed through multiple channels, including a consumer facing application on a mobile device.

##### 6.5.5.2.1.1 `createAccount` message sequence

The MNO or affiliate should be capable of creating new accounts in the MNO service domain. A new account represents an individual identity that may be subsequently interacted with through a variety of physical channels. The account on its own does not provide any service other than as a placeholder against which individual merchant and manufacturer loyalty cards and accounts may be registered.

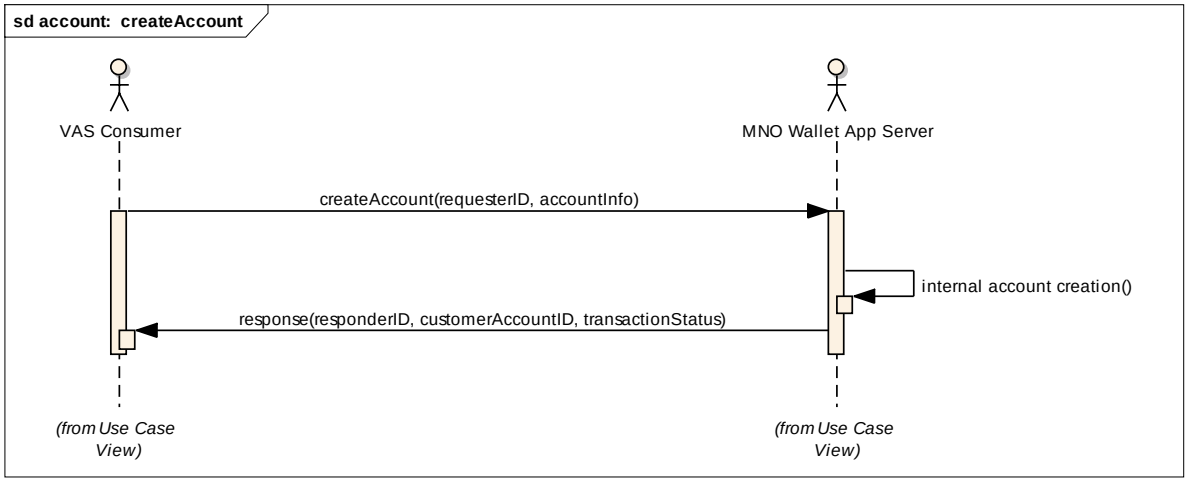


Figure 56: `createAccount` **message flow**

Access to this call should be protected with an appropriate layer of security to ensure the calling entity has rights to create new accounts and that the content of any call, which may contain personal data, is appropriately protected.

Figure 56 illustrates the simple request/response flow of the `createAccount` interface. The service consumer (MNO wallet app or authorised third-party service) makes a call to the MNO wallet app server, passing appropriate `accountInfo` that conveys the account holder's details.

The MNO wallet app server shall determine an appropriate `accountID`, create the account and link to the assigned identifier. This can possibly be either `customerAccountID` or `customerAccountAlternateID` (see Figure 9) The response contains a `transactionStatus` block, indicating the success of the operation and, in a success scenario, the response will indicate to the caller the newly created `accountID` to be used in subsequent calls.

#### 6.5.5.2.1.2 `updateAccount` message sequence

A created account might contain out of date information that needs updating. Updates are supported using the `updateAccount` interface, which enables a calling service to update specific fields of the `accountInfo` structure.

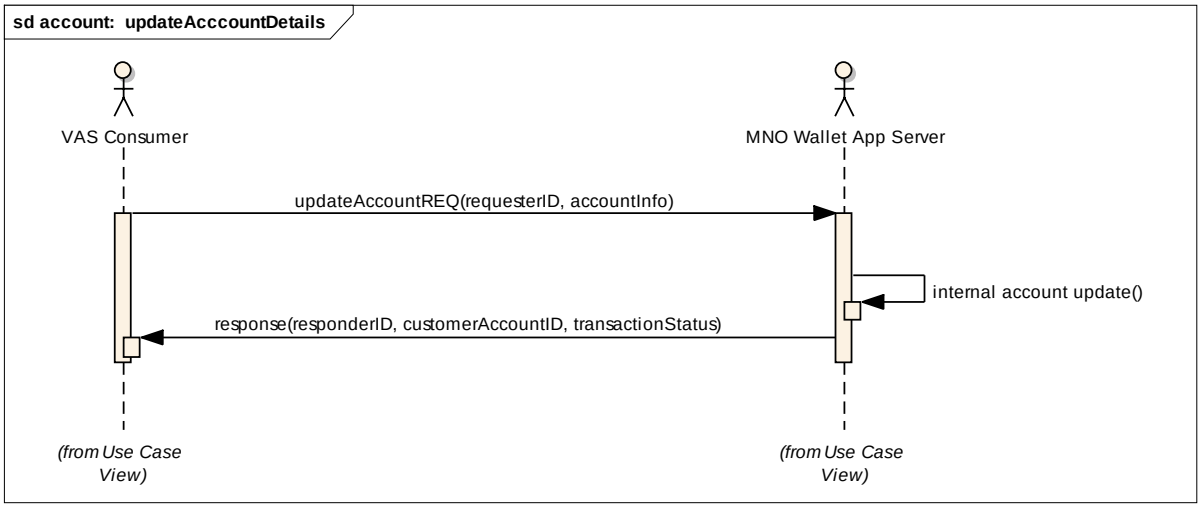


Figure 57**:** `updateAccount` **message flow**

Figure 57 illustrates the simple request/response flow of the `updateAccount` interface. The service consumer (MNO wallet app or authorised third-party service) makes a call to the MNO wallet app server, passing appropriate `accountInfo` that conveys the account holder's details.

The MNO wallet app server shall determine the appropriate update to apply and update the loyalty account accordingly.

#### 6.5.5.2.1.3 `getAccountDetails` message sequence

During operation, a calling service (such as the MNO wallet app, some third-party ecommerce site and call centre) might need to query the details of an MNO wallet holder's account. `getAccountDetails` interface provides this functionality and enables account detail summary to be returned for presentation through the mobile application or some third-party tool.

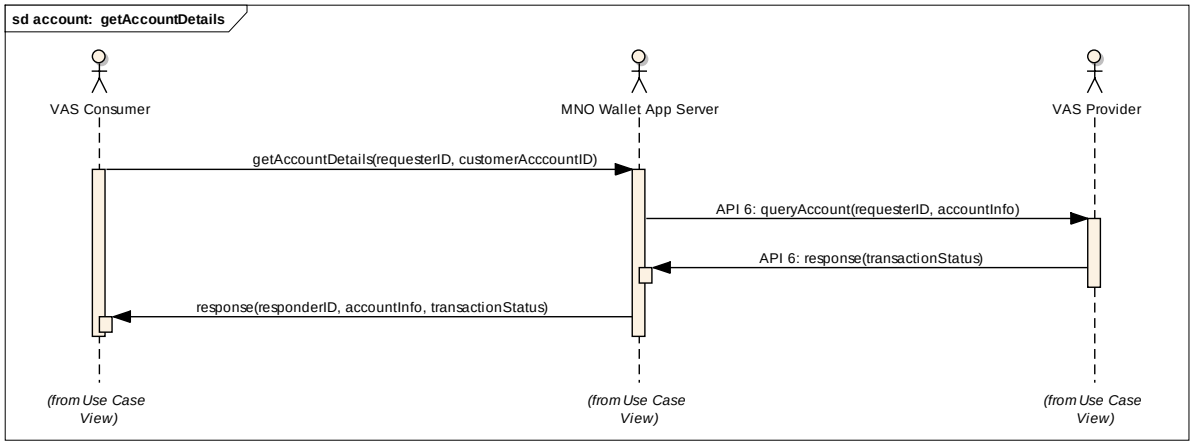


Figure 58: `getAccountDetail` **message flow**

To fulfil this request it may be necessary for the MNO wallet app server to call out to the appropriate set of third-party loyalty service providers to retrieve details of all relevant accounts held by the MNO wallet holder. Real-time calling and aggregation of multiple third-party loyalty service providers may not be sufficiently responsive to provide a timely response to the calling entity and it is likely that some local caching be performed. This proposal does not mandate how, or how frequently updates are pulled from subordinate loyalty service providers; this could be anything from real-time service calls to fixed-frequency batch updates.

#### 6.5.5.2.1.4 `deleteAccount` message sequence

A MNO wallet holder that decides to withdraw from the service should have some mechanism to delete their account from the system. This may take the form of a hard delete where the account, linked services and loyalty accounts are physically erased, or a soft, logical delete where the account is merely suspended and may be resurrected at a later date.

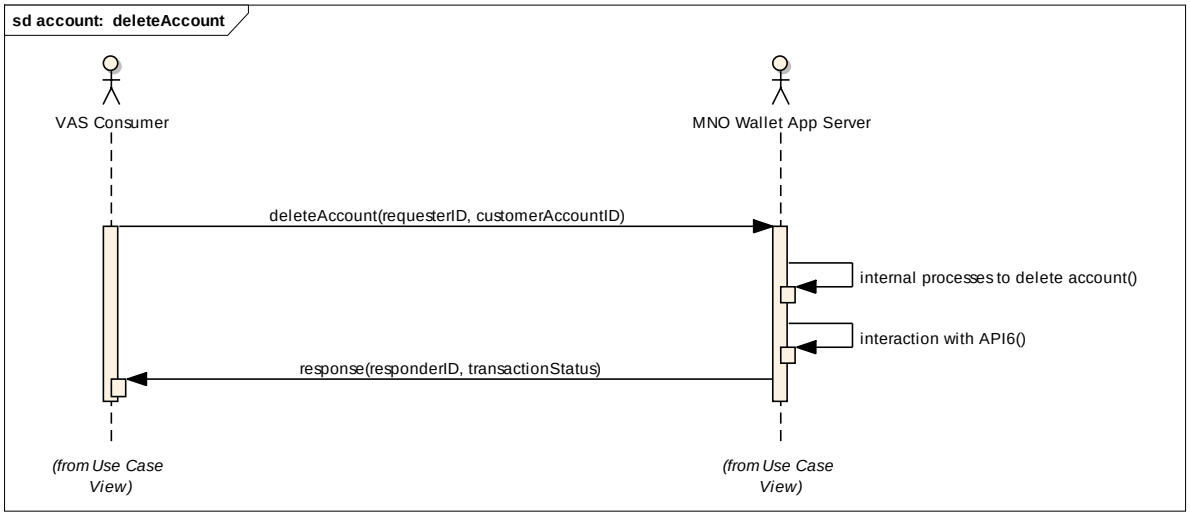


Figure 59**: delete account:** `deleteAccount` **message sequence**

Depending on the MNO implementation it may be necessary for the service to call out to linked or subsidiary services, such as loyalty service providers, to delete or clear linked accounts.

#### 6.5.5.2.1.5 `linkLoyaltyCard` message sequence

An MNO wallet app user might wish to add already existing or already issued loyalty cards to their wallet. The addition of existing loyalty schemes to an MNO wallet is enabled by the `linkLoyaltyCard` service call. The consumer-facing interface must be capable of eliciting sufficient detail of existing loyalty cards (such as the scheme name or account ID) and conveying these details to the MNO wallet app server for processing and links.

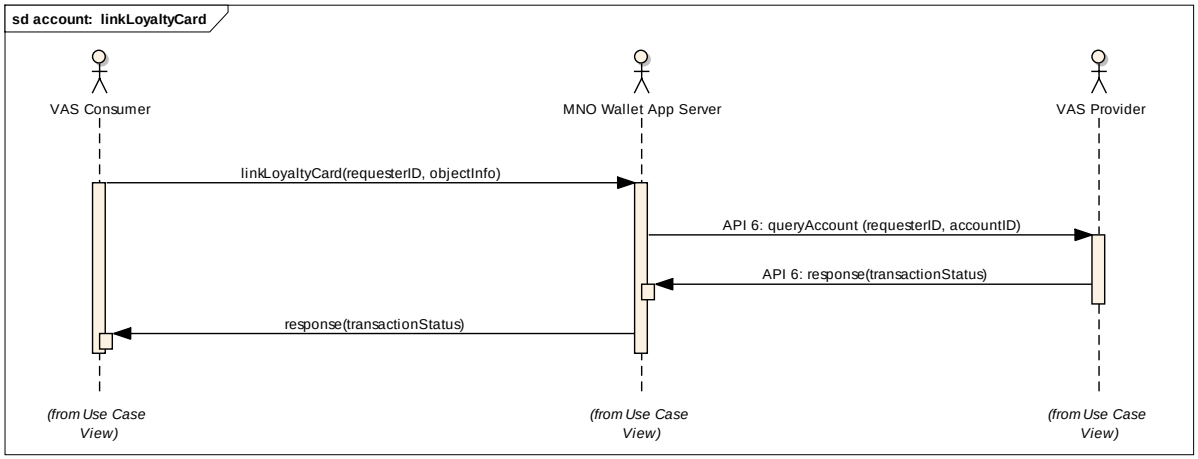


Figure 60: `linkLoyaltyCard` **message sequence**

The MNO wallet app server receives a call from the service consumer that contains sufficient data to identify a specific existing loyalty card and perform the appropriate processing to link this card (account) to the existing MNO wallet account. The MNO and loyalty service provider will need to establish a mechanism to ensure authentication and prevent fraudulent addition of cards. In Figure 60, a simple call is posted to the loyalty service provider, via API 6, to elicit key account details, which can be used to verify key details submitted by the caller (communicated within `ObjectInfo`).

#### 6.5.5.2.1.6 `deleteLoyaltyCard` message sequence

It should be possible for a consumer to delete a third-party loyalty account added through their MNO wallet. This is enabled through the `deleteLoyaltyCard` method and removes a specific manufacturer or retailer loyalty card from the MNO service and from the wallet itself. This may need interaction with third-party loyalty services that may or may not expose delete functionality through their public API. The MNO service shall provide the facility to physically or logically delete all data relating to a specific loyalty card (and subordinate loyalty service provider account).

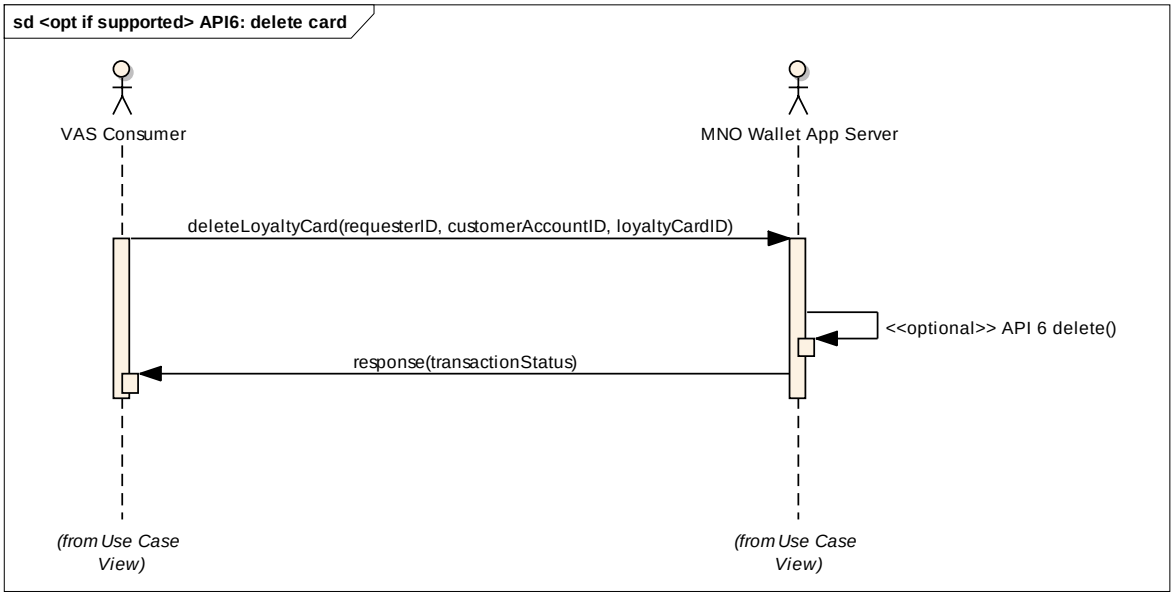


Figure 61: `deleteLoyaltyCard` **message sequence**

The calling entity passes key parameters that uniquely identify the card to be deleted, and the MNO service shall perform all tasks necessary to delete that card from the MNO and third-party service provider accounts. The third-party service provider or parent retailer/manufacturer may have pre-existing rules and processes for account deletion and data retention that should be adhered to by the MNO service, and reflected in the terms and conditions presented to the consumer.

#### 6.5.5.2.1.7 `getLoyaltyCardDetails` message sequence

Querying an account with `getAccountDetails` will return a summary of details such as offers, issued coupons and points balances for each loyalty service account linked to the MNO wallet account. Specific item details should be queried with an appropriate method and full details of a linked loyalty card can be returned using the `getLoyaltyCardDetails` method.

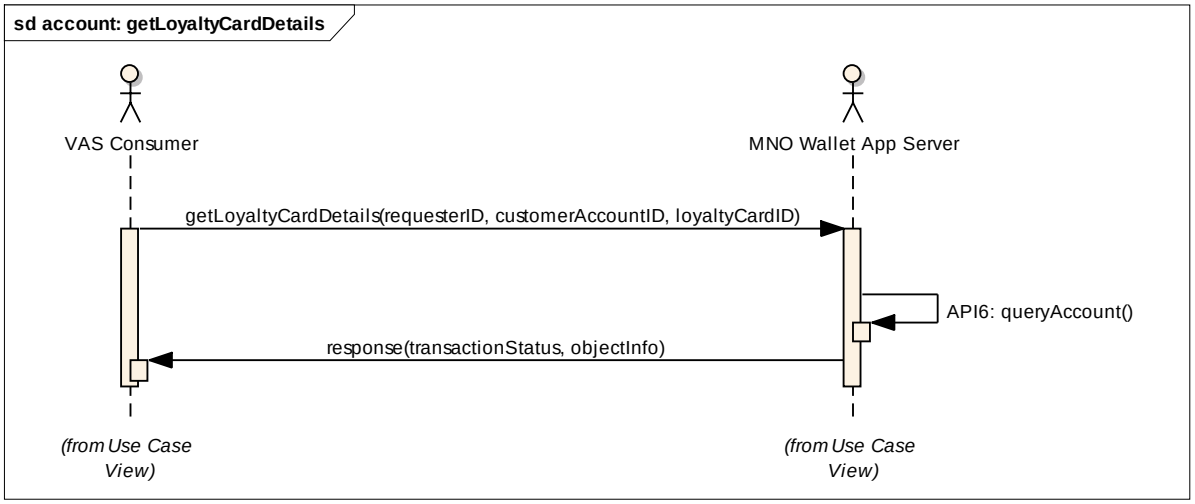


Figure 62: `getLoyaltyCardDetails` **message sequence**

The request/response `getLoyaltyCardDetails` exchange is used to elicit full details of a specific added or linked loyalty card and returns full details including multimedia content that can be rendered on a mobile device or ecommerce site.

API 6 does not mandate a method to elicit details of specific coupons, offers or loyalty cards. Some loyalty service providers may provide this functionality and the MNO wallet app server may use that mechanism to programmatically extract rich coupons, offers or loyalty card details. The information could also be provided as part of the merchant/manufacturer on-boarding phase; either programmatically via offer/service creation APIs or through manual processes, which are out of scope of this document.

### 6.5.5.2.2 Offer management sequences

The offer management cases enable the MNO wallet app server, or affiliated third-party service, to expose offers to multiple channels, including a consumer-facing application on a mobile device. Offers are only a representation of potential coupon rewards that might subsequently be requested.

#### 6.5.5.2.2.1 `publishOffer` message sequence

The `publishOffer` method is used to present an existing offer in the service provider of a linked loyalty account. These offers are configured within individual back-office systems and exposed/distributed to MNO wallet app users via the MNO service.

The `publishOffer` exchange is a simple request response sequence in which the calling service requests the publication of an existing offer. The requester must pass all information that identifies the specific offer, the hosting service provider and target accounts to which the offer should be published.

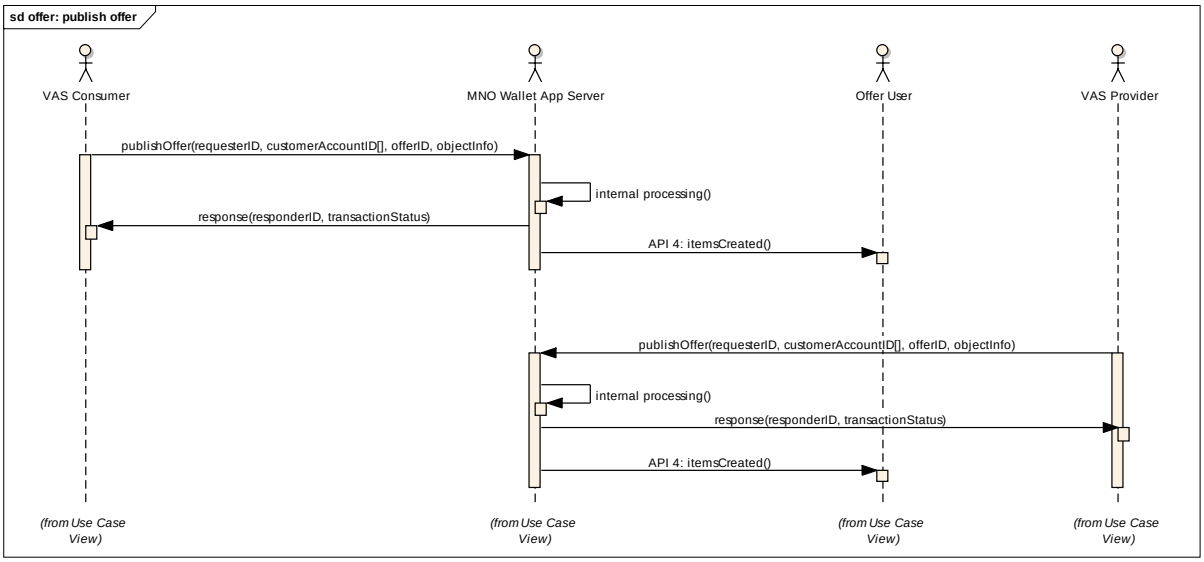


Figure 63**:** `publishOffer` **message sequence**

As defined by Figure 63, calls to this MNO service method may come from either VAS Consumers (such as wallet app or third-parties) or the VAS Provider themselves. For example, a loyalty service provider may implement logic that supports the triggering of offer publications based on POS events or internal insight capabilities. In this case, the offer is created within the loyalty service provider and exposed to the MNO channel by the loyalty service provider.

#### 6.5.5.2.2.2 `suspendOffer` message sequence

The suspension of an offer simply prevents further distribution of an already published offer. An example of this scenario is where an offer is already published and valid for existing members of an MNO service but is not to be made visible to new joining members.

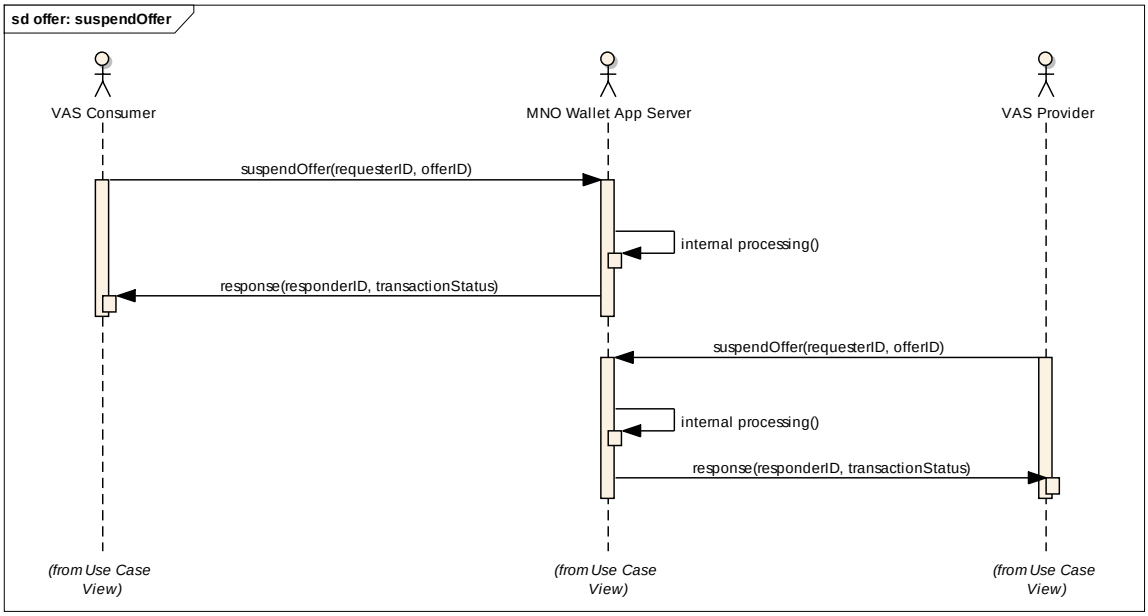


Figure 64: `suspendOffer` **message sequence chart**

The method is a simple request/response sequence that does not initiate any specific communication with wallet applications. The call merely affects the behaviour of the MNO wallet service and prevents further distribution of the specified offer.

#### 6.5.5.2.2.3 `retractOffer` message sequence

Offer retraction enables a calling third-party to actively recall a previously published offer from the channels where it had been published. From a caller perspective, the sequence is request/response, but there is potential additional pro-active communication with individual MNO wallet applications to actively prevent the offer being visible.

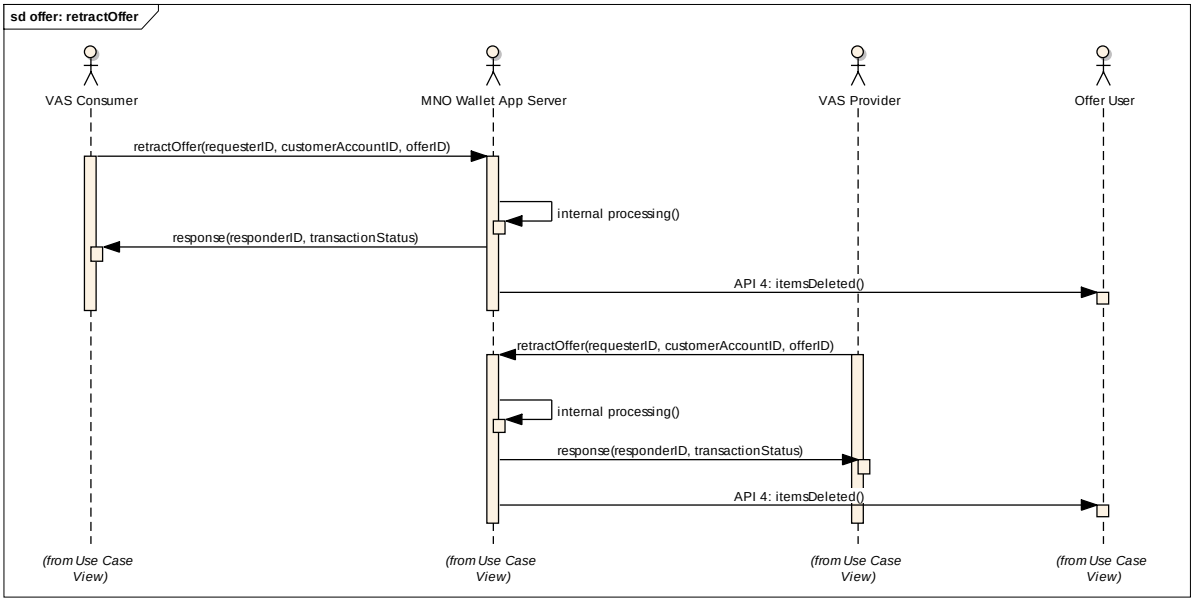


Figure 65**:** `retractOffer` **message sequence**

It may also be that the MNO wallet application refreshes available offers from the MNO service and that the retracted offer is simply not displayed in any updates subsequent to the retract call being made.

#### 6.5.5.2.2.4 `getPublishedOffers` message sequence

`getPublishedOffers` allows a calling entity to query the offers published and applicable to a specific MNO wallet account holder. The sequence for this operation is illustrated in Figure 66 and constitutes a simple request/response from the caller's perspective.

Internally, the MNO service may interact with VAS Providers to prompt further details about published offers that are available to the specific account holder. This may need real-time interrogation of multiple service providers or simply the return of offer information cached from previous batch account queries.

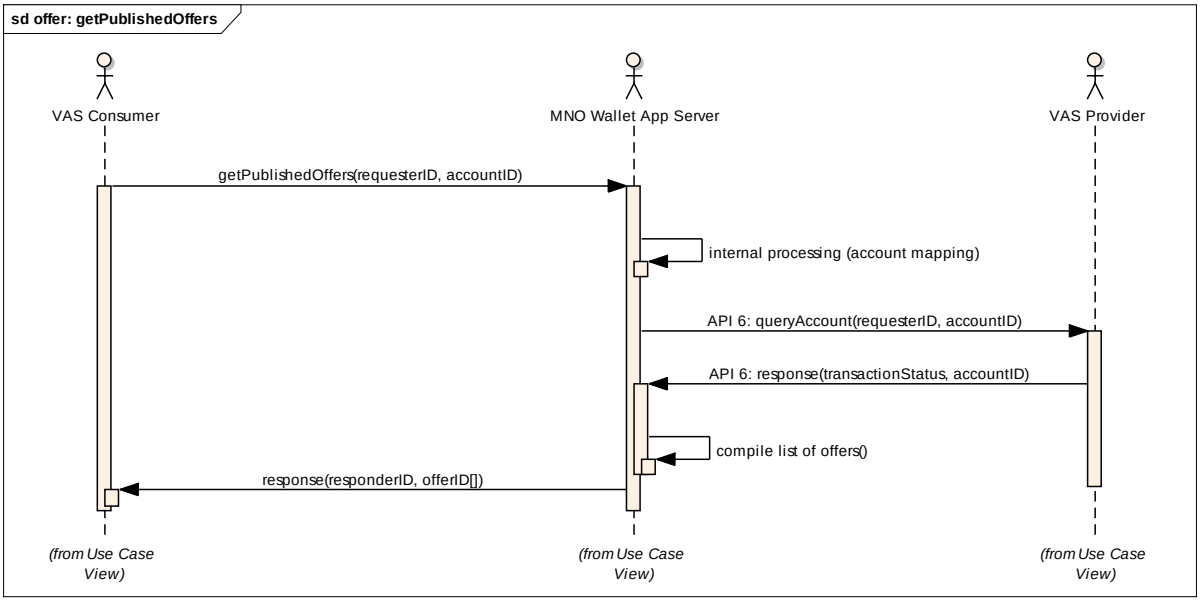


Figure 66: `getPublishedOffers` message sequence

Offer details are collated by the MNO wallet app server and returned to the calling entity to be presented on the mobile application, ecommerce site or other channel interface. The response contains only a summary of available offers, with specific details including offer copy being returned with the `getOfferDetail` method.

#### 6.5.5.2.2.5 `getOfferDetails` message sequence

The `getPublishedOffers` method returns a summary list of offers assigned to an individual MNO wallet holder account; across all VAS Providers. If a calling service needs specific details on individual offers (such as expiry date, valid product list and rich media content) then the `getOfferDetail` method is used:

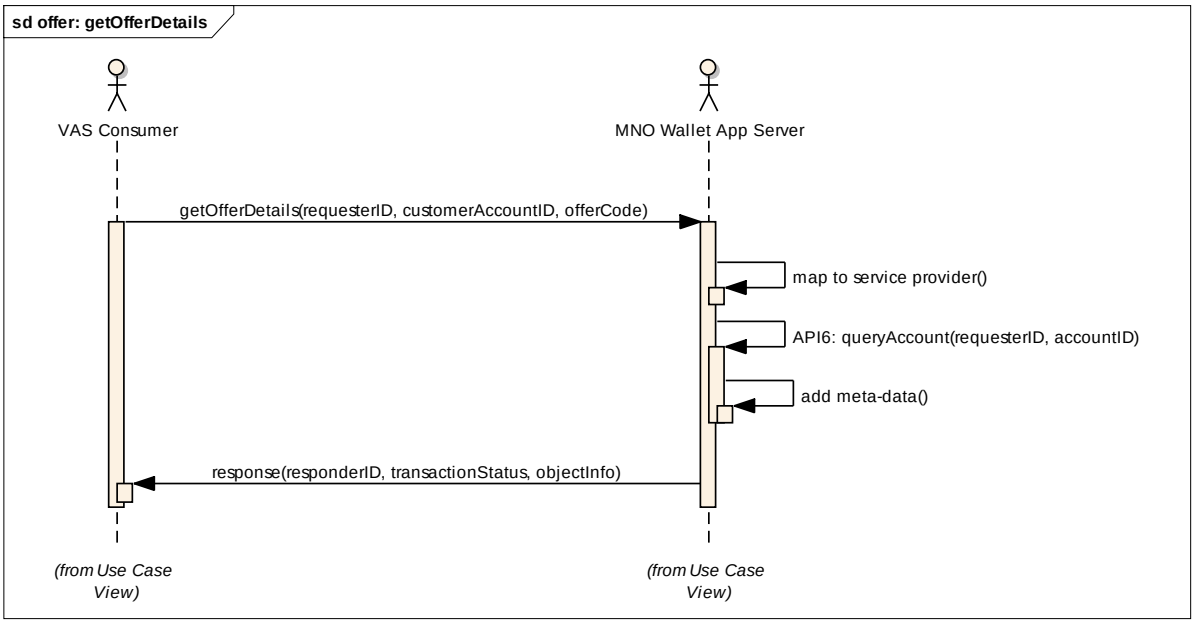


Figure 67: `getOfferDetails` **message sequence**

`getOfferDetails` is a simple request/response method, returning complete data about a specific published offer to the calling entity for subsequent presentation on the chosen channel. Fulfilment of this call may need further calls to merchant and manufacturer loyalty services, or optionally this information may be cached locally with the MNO service, updated infrequently via batch processes.

API 6 does not mandate a method to prompt details of specific coupons, offers or loyalty cards. Some loyalty service providers may provide this functionality and the MNO wallet app server may use that mechanism to programmatically extract rich coupon, offer or loyalty cards detail. The information may also be provided as part of the merchant/manufacturer on-boarding phase; either programmatically via offer/service creation APIs or through manual processes, which are out of scope of this document.

#### 6.5.5.2.3 Coupon management sequences

Coupons for the concrete realisation of an offer; are individual items that are redeemable at the POS. This proposal identifies a number of use cases and interfaces for coupon management, with the associated flows detailed in the following sections.

##### 6.5.5.2.3.1 `acquireCoupon` message sequence

Coupon acquisition follows from offer publication and is the process by which a concrete item is created from a generic offer. Multiple entities can request that a coupon is assigned to a MNO wallet holder's account and the MNO service is responsible for orchestrating this operation with the merchant and manufacturer services that support the requested offer, as defined in Figure 68.

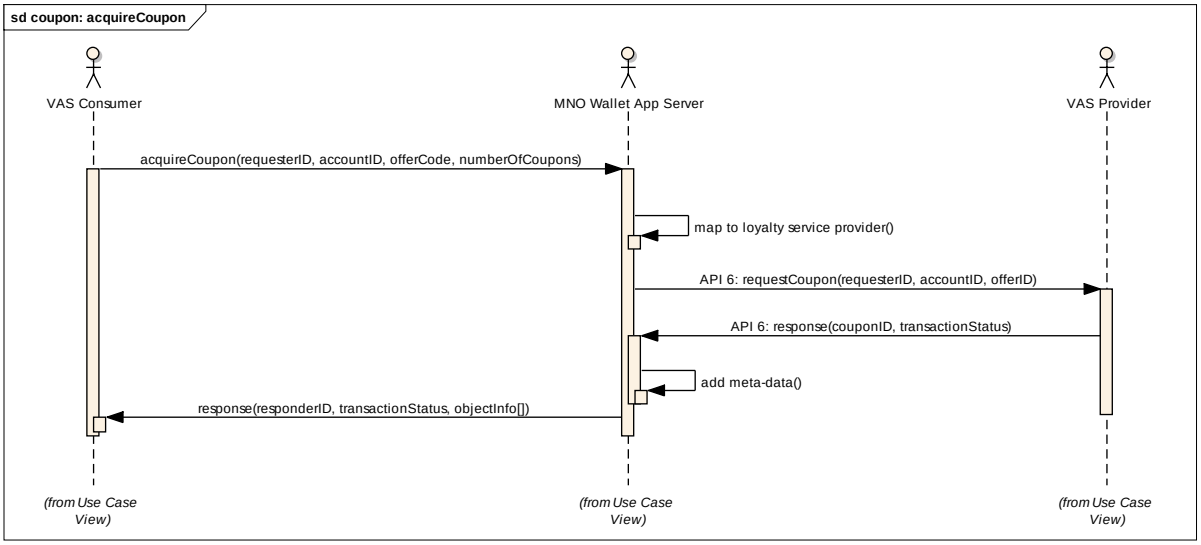


Figure 68: `acquireCoupon` **message sequences**

The calling service should request that a specific offer is awarded and should identify the offer, the account for the offers, and potentially other parameters, such as number of coupons to generate.

The MNO wallet service is responsible for conducting exchanges with the supporting merchant and manufacturer service providers to attempt an award of the coupon from the requested offer.

There may be offer-specific conditions enforced within the MNO service or the third-party loyalty service provider. The MNO service is responsible for the management of service provider integration and corner cases. It will return a response to the calling entity to define the success status of the operation, along with coupon details as appropriate.

#### 6.5.5.2.3.2 `activateCoupon`/`deactivateCoupon` message sequence

Issued coupons may be presented at the POS via the MNO wallet application. However, an alternate flow at POS is that only VAS Provider identifiers are passed and the POS queries VAS Providers to elicit a list of coupons to be applied to the current transaction.

To facilitate this mode of operation, it is necessary for the MNO wallet service (and supporting VAS Providers) to indicate the specific issued coupons to be used for the next transaction. The coupons to apply could be a subset of those issued against the customer's account and there must be a mechanism to differentiate between the selected coupons and the coupons not to be applied. This is achieved using the `activateCoupon`/`deactivateCoupon` call, which marks coupons in VAS Provider services as valid/not valid for use at a subsequent transaction.

Activation and deactivation is a simple request/response method and requires real-time support from the VAS Provider to support, affect and acknowledge the requested change.

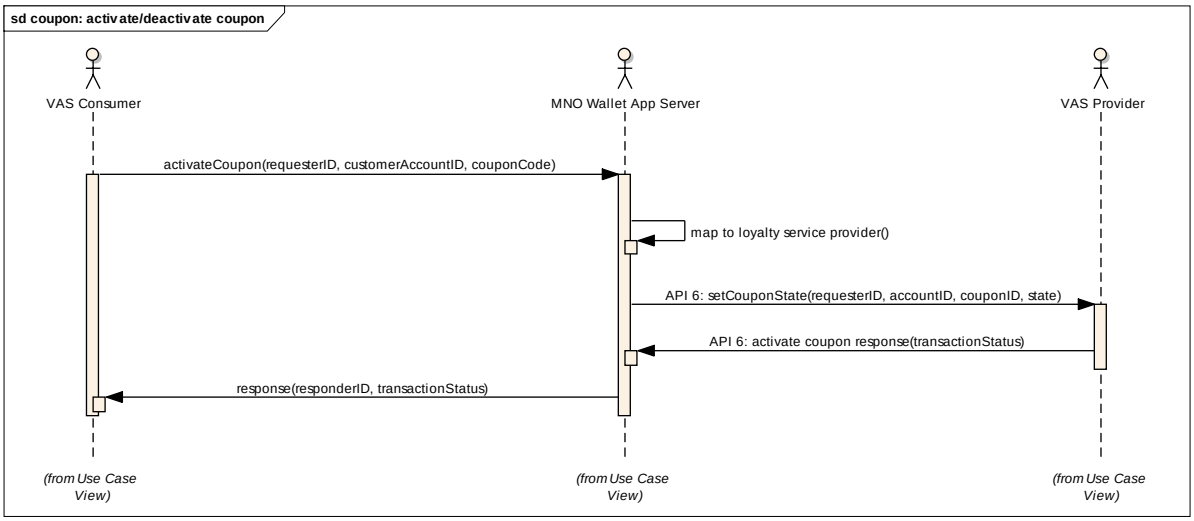


Figure 69: `activateCoupon`/`deactivateCoupon` **message sequence**

The MNO wallet service may implement variations of this activation/deactivation behaviour, for example, activating coupons in a VAS Provider service for a defined period only after which they "time-out" and are deactivated; MNO and VAS Provider specific extensions are out of scope here.

#### 6.5.5.2.3.3 `cancelCoupon` message sequence

A coupon might have mistakenly been requested by the mobile device, MNO or third-party service and, therefore needing to be cancelled. The `cancelCoupon` method is a simple request/response sequence to support cancellation.

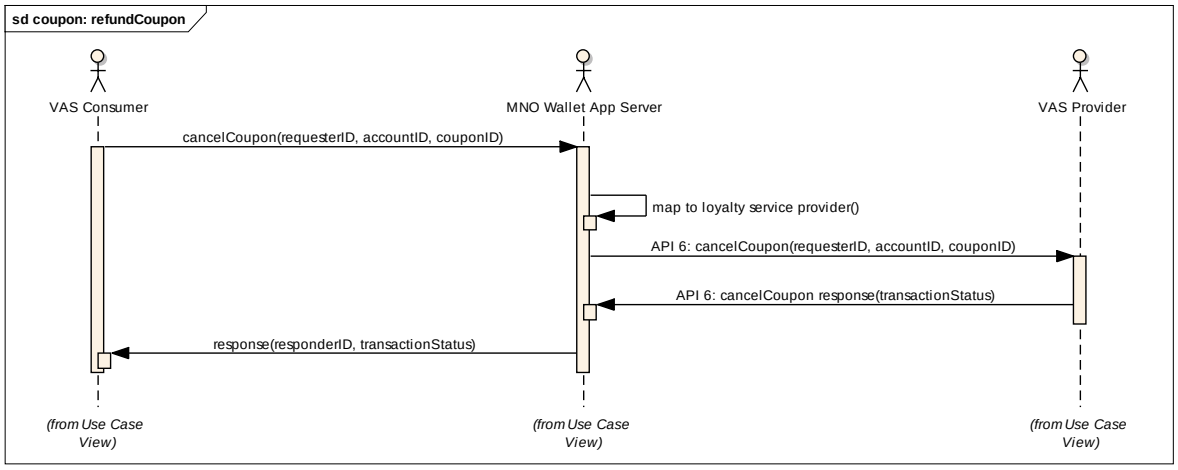


Figure 70: `cancelCoupon` **message sequence**

It shall be necessary to interact with the VAS Provider acting as coupon issuer to cancel the coupon. This may be done in real-time, as shown in Figure 70 or by some batch non-real-time method. The latter would require clear communication through the consumer facing channels.

#### 6.5.5.2.3.4 `getAcquiredCoupons` message sequence

The `getAcquiredCoupon` method shall be used by calling entities to elicit a list of coupons that are assigned to a MNO wallet user account. The request specifies the wallet user account to query and the response shall contain a simple list of individual coupons with a minimal subset of information about each individual coupon.

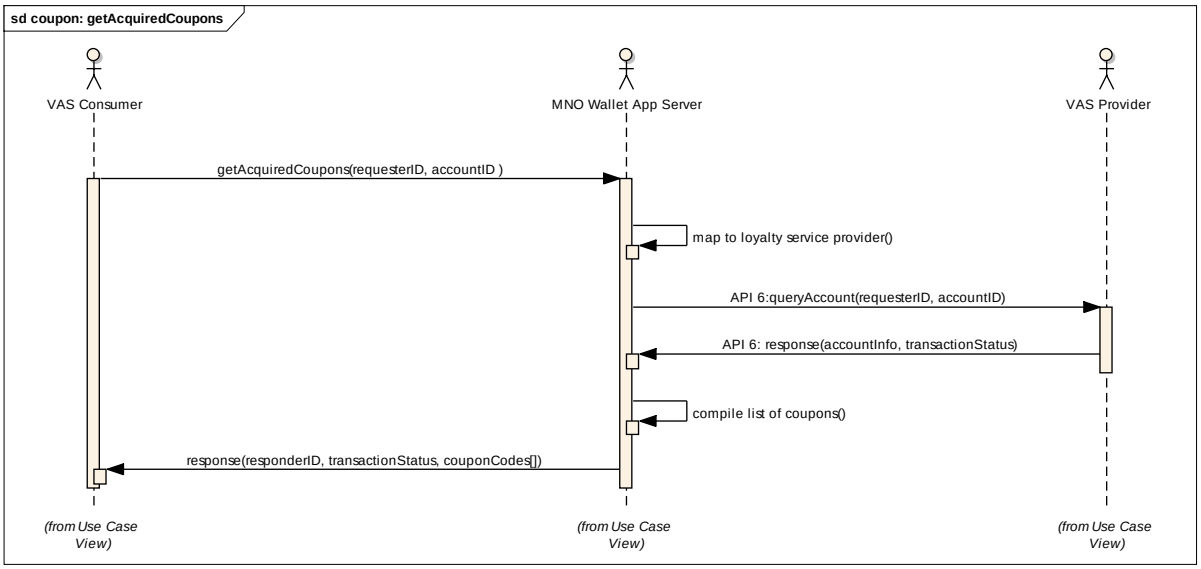


Figure 71: `getAcquiredCoupons` **message sequence**

Figure 71 illustrates real-time interaction with a third-party loyalty service. However, the MNO service may retrieve the coupon detail from each linked manufacturer/merchant loyalty service in real-time as illustrated, or from a local cache, updated via some batch mechanisms, to facilitate a quick response. Integration with VAS Provider services is MNO service-specific and this implementation is not mandated in the proposal.

#### 6.5.5.2.3.5 `getCouponDetails` message sequence

The `getCouponDetails` method is used by a calling service to elicit full details of an individual issued coupon. This provides a facility to present all information, including rich-media content, in a customer-facing channel.

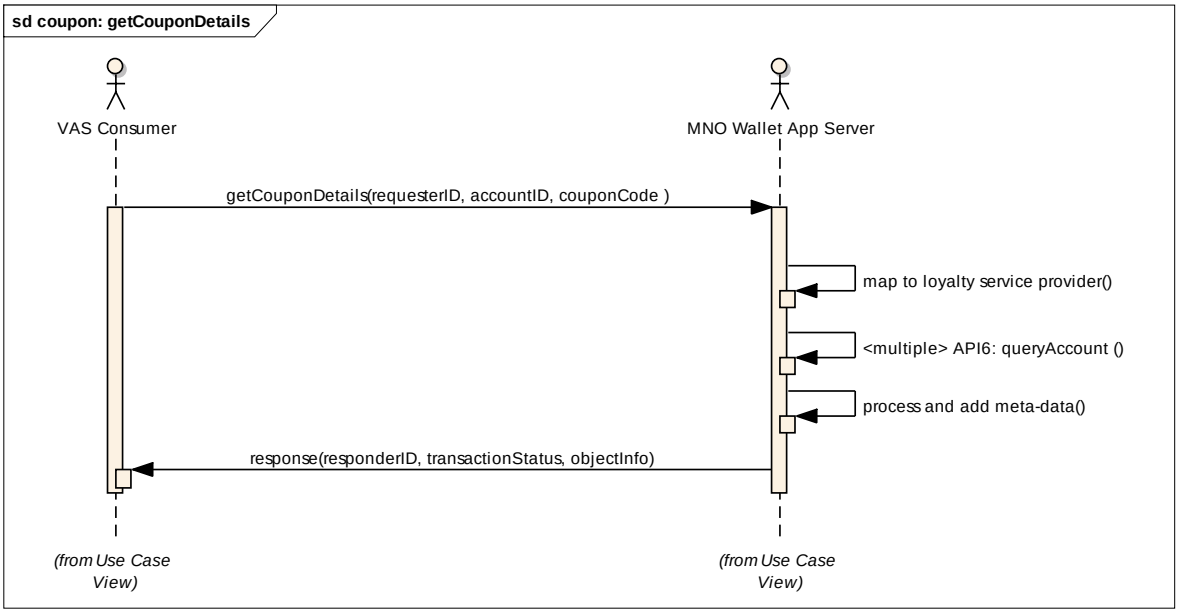


Figure 72: `getCouponDetails` **message sequence**

The MNO service shall collate all information available about the requested coupon and return that to the calling service. It is possible that information about the individual coupon is distributed across a number of services; for example, the VAS Provider might support only textual information with rich-media content stored in some external content management service. The MNO service may provide content management and shall be responsible for collating a consistent and complete response dispatched back to the entity that made the original call.

API 6 does not mandate a method to elicit details of specific coupons, offers or loyalty cards. Some loyalty service providers may offer this functionality and the MNO wallet app server may use that mechanism to programmatically extract rich coupon, offer or loyalty cards details. The information may also be provided as part of the merchant/manufacturer on-boarding phase; either programmatically via offer/service creation APIs or through manual processes, which are out of scope of this document.

#### 6.5.5.2.4 Transactional sequences

Transactional messages support informative communication between components. The mandatory use cases support:

a) service provider notification of transaction events
b) message distribution to mobile devices
c) mobile device notification of transactional interaction.

##### 6.5.5.2.4.1 `eventNTF` message sequence

The `eventNTF` message is an unsolicited notification from a VAS Provider service that indicates that some operation occurred that affected the status and content of a linked loyalty account (such as available offers, issued coupons or points balances).

The MNO service must expose an interface and is responsible for appropriate registration of this interface and definition of any specific set of events with VAS Provider services.

Once event registration is complete, the MNO service should await notification messages and then process them as received.

This proposal does not mandate the complete set of notification messages, nor how any individual message is to be processed. It does, however, mandate that such an interface exists and that it can be used to drive real-time, post-transaction updates to a mobile application as the result of a change in a linked loyalty account.

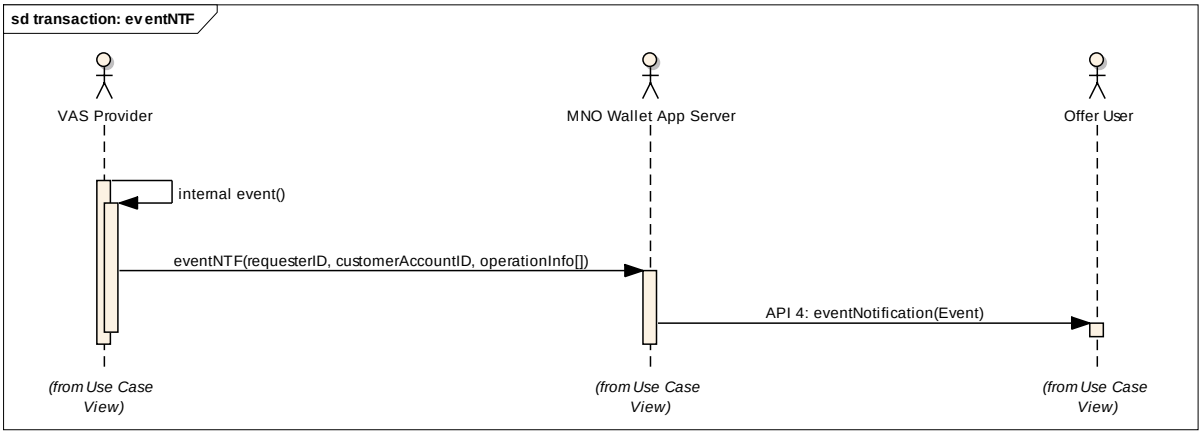


Figure 73: `eventNTF` **message sequence**

The sequence chart in Figure 73 illustrates a simple scenario where some internal event within a loyalty service provider triggers a notification to the MNO wallet service. The internal event within the loyalty service provider could be a transaction registered within a merchant POS environment, some post-transactional reconciliation event (for example FMCG coupon redemption), a customer service amendment or similar events.

The internal event is dispatched in an `eventNTF` message to the MNO service, indicating `source`, `account` and `eventType`, and in Figure 73 the MNO service chooses to post an update to the MNO wallet application.

#### 6.5.5.2.4.2 `informativeMessage` message sequence

Informative messages are used to convey general information from a source to a nominated MNO wallet account holder. The message can originate from a general service consumer as well as a VAS Provider and can be generated as a result of a variety of triggers; geo-location, new available offers, time of day, or general communication request from merchant or manufacturer.

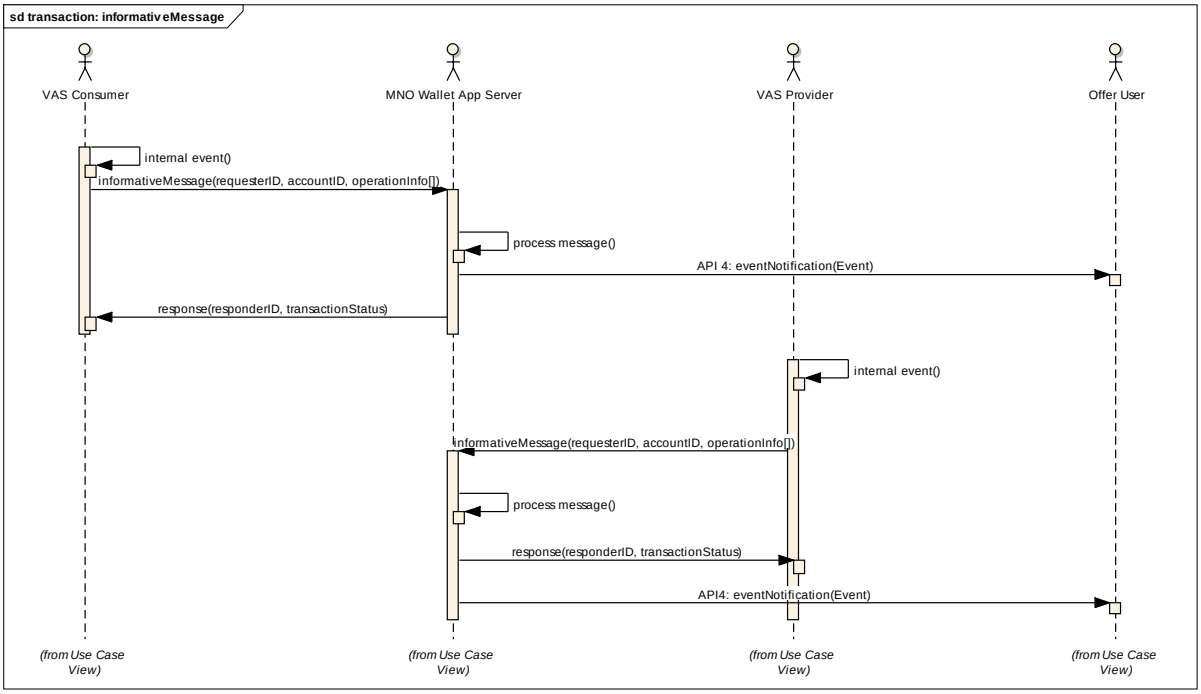


Figure 74: `informativeMessageNTF` **message sequence**

In the example of Figure 74 the MNO service receives `informativeMessageNTF` requests from differing sources and in real-time relays appropriate message notifications to the MNO wallet app.

#### 6.5.5.2.4.3 `mobileTransactionNTF` message sequence

Some interactions at the POS will trigger real-time communications to a supporting VAS Provider service; other interactions, as in the case of FMCG, are likely to operate in more traditional batch mode. In the latter case, it is possible to use the mobile to MNO service channel as a more real-time means to update a loyalty account (such as points earned and coupons consumed).

The `mobileTransactionNTF` interface supports this style of communication and allows for real-time reconciliation where the MNO wallet app can confirm post-transaction events in non-real time POS environment.

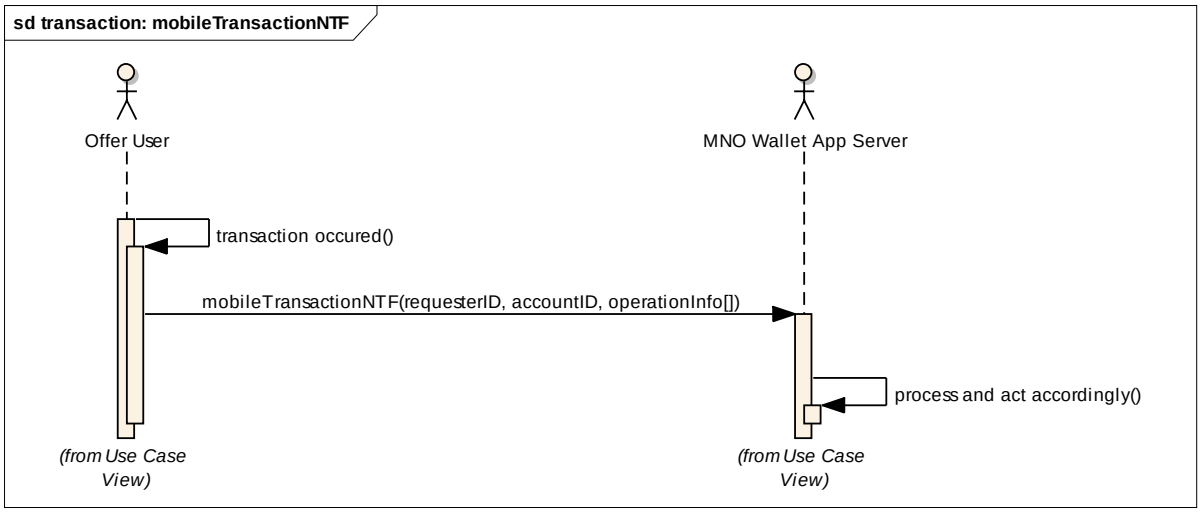


Figure 75: `mobileTransactionNTF` **message sequence**

## 6.6 API 6 – Interface from the MNO wallet app server to the backend systems

### 6.6.1 Scope

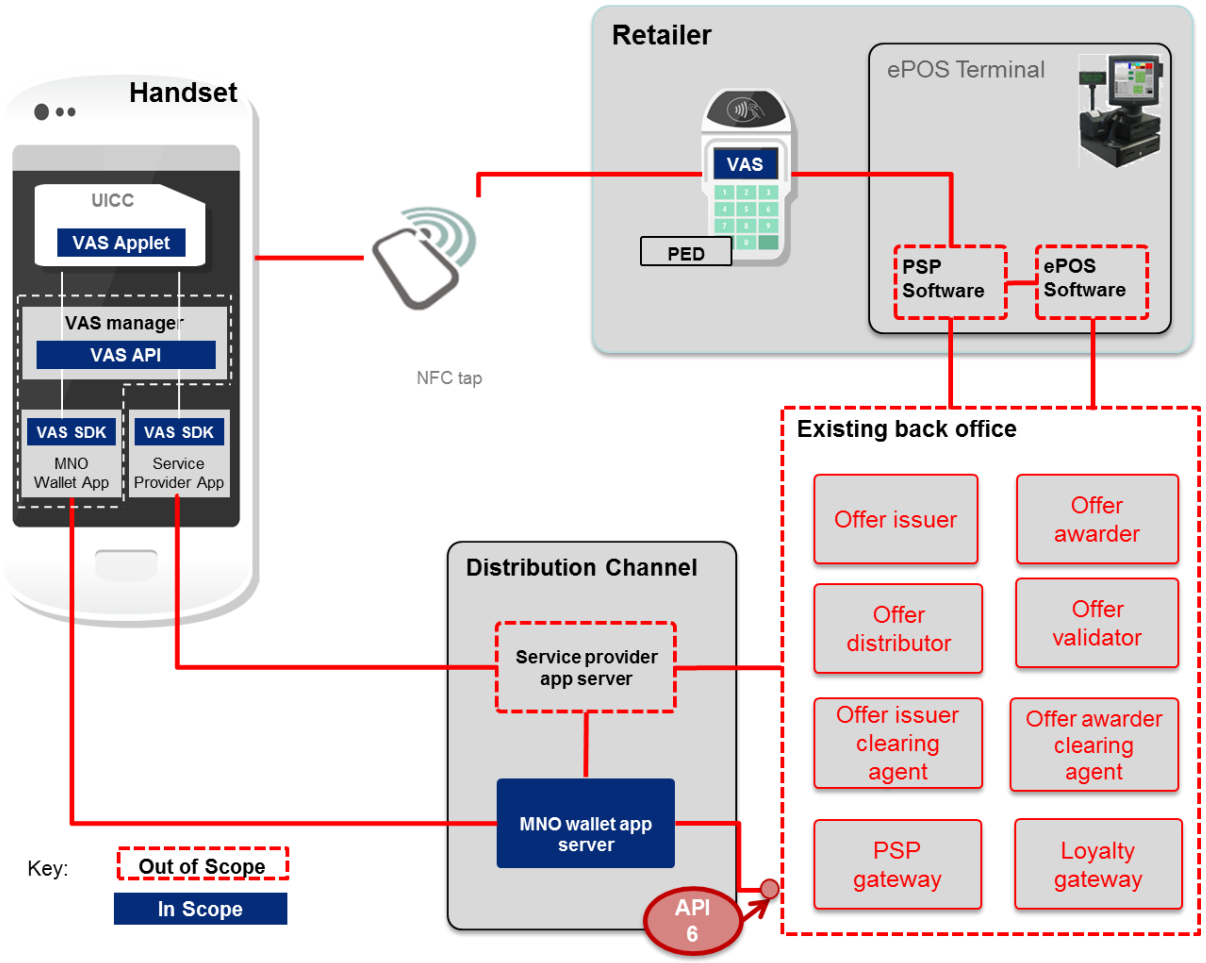


Figure 76**: Location of API 6 within the proposed architecture**

This section defines API 6 of the GSMA Retail Technical Architecture proposal, which represents the service interface from the distribution channels to the back office systems.

Unlike earlier chapters, API 6 does not prescribe the exact service methods or their individual interface prototypes. Rather than defining a concrete interface, this section instead focuses on the minimum set of functionalities that a loyalty service provider must expose to support the MNO value added services platform.

### 6.6.2 Actors

The key external actor of API 6 is the MNO wallet application server, which interacts with the merchant or manufacturer loyalty service provider to affect the coupon and loyalty experiences.

VAS Provider services will present interfaces and capabilities for the merchant and manufacturer systems, but these are out of scope of this document. These out of scope interfaces will facilitate offers and campaign definitions, as well as operational support for real-time interactions and maintenance at the point of sale and across all multi-channel points of interaction, such as kiosk, customer service and ecommerce portals.

### 6.6.3 Use cases

API 6 represents the service interface of a merchant or manufacturer loyalty service. The use cases illustrated in Figure 77 represent the minimum set of functionality required. The merchant, manufacturer or third-party loyalty service provider may readily extend the core cases as necessary, which can be deployed as a differentiator proposal.

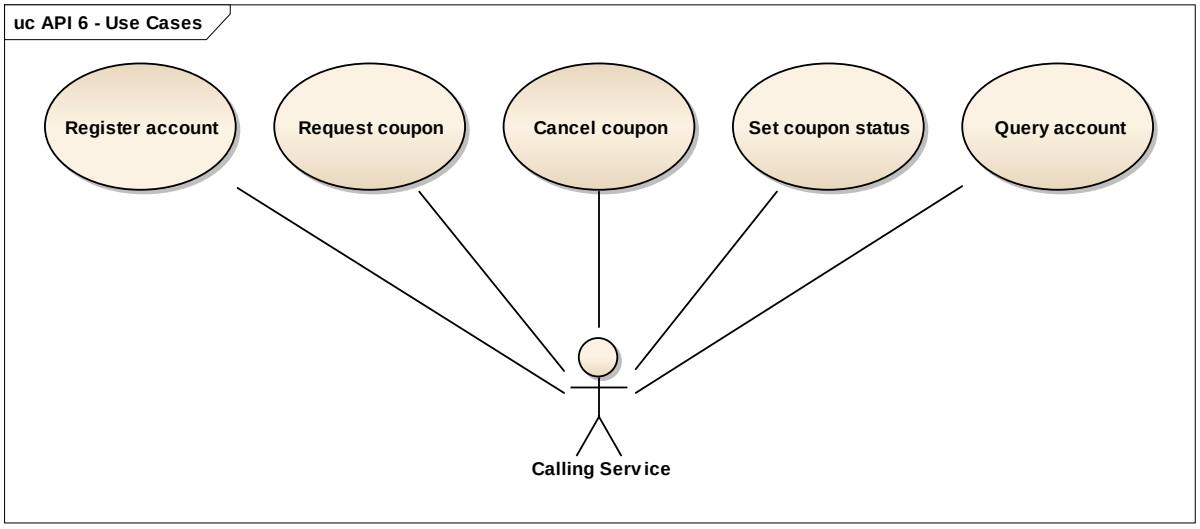


Figure 77**: API 6 core use cases**

| Use Case | Description |
|---|---|
| Register account | **Source:** MNO wallet app server<br>**Target:** VAS provider<br>**Description:** Service provider interface to support the dynamic registration of a new account within the nominated loyalty scheme. Caller must identify the scheme in the call and have necessary authorisation to create new accounts within that scheme. |
| Query account | **Source:** MNO wallet app server<br>**Target:** VAS provider<br>**Description:** Service provider interface to enable the MNO wallet app server to query the details of a specific loyalty service account. Caller must identify the scheme in the call and have necessary authorisation to create new accounts within that scheme. |

| Use Case | Description |
|---|---|
| Request coupon | **Source:** MNO wallet app server<br>**Target:** VAS provider<br>**Description:** Service provider interface to enable the MNO wallet app server to assign a unique coupon to a nominated wallet user account, to which the offer was previously published and is currently available. Caller must identify the offer in the call and have necessary authorisation to request coupon assignment. |
| Cancel coupon | **Source:** MNO wallet app server<br>**Target:** VAS provider<br>**Description:** Service provider interface to enable the MNO wallet app server to remove a previously assigned coupon. Caller must identify the coupon in the call and have necessary authorisation to request coupon cancellation. |
| Set coupon status | **Source:** MNO wallet app server<br>**Target:** VAS provider<br>**Description:** Service provider interface to enable the MNO wallet app server to set the state of an individual coupon. Coupon may e.g. be set active to indicate it is to be redeemed at the next POS interaction (see section 4.2.8 for full list of available status codes). Caller must identify the coupon in the call and have necessary authorisation to amend the coupon state. |

**Table 71: Use case descriptions data structure**

| Name | Description |
|---|---|
| `offerID` | Primitive data type to uniquely identify offers throughout the platform. |
| `couponID` | Primitive data type to uniquely identify coupons throughout the platform. |
| `loyaltyCardID` | Primitive data type to uniquely identify loyalty cards throughout the platform. From an implementation perspective, `customerAccountAlternateID` (see Figure 9) can be used at this. |
| `requesterID` | Identity of the calling service, used to authenticate the connection and confirm that the calling entity has sufficient rights to affect the change requested by the API call. |
| `responderID` | Identity of the responding service, used to authenticate any response to the API caller. |
| `accountInfo` | Represents the complete summary of a customer's account. This composite is also used to update and return account information based on appropriate requests. |

**Table 72: Data structure**

### 6.6.4 Messages and parameters

| Method | Parameters | Description |
|---|---|---|
| queryAccount | REQUEST P1: requesterID<br>REQUEST P2: accountID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: accountInfo<br>RESPONSE P3: transactionStatus | Queries details of the loyalty account identified within specified loyalty scheme. The ID's used here will be mapped back to high level class. |
| requestCoupon | REQUEST P1: requesterID<br>REQUEST P2: accountID<br>REQUEST P3: offerID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: couponID<br>RESPONSE P3: transactionStatus | Requests that a coupon be generated that can be subsequently redeemed to achieve a specified offer. |
| cancelCoupon | REQUEST P1: requesterID<br>REQUEST P2: accountID<br>REQUEST P3: couponID<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus | Deletes a specified (previously issued) coupon from a loyalty account holder's account. |
| setCouponStatus | REQUEST P1: requesterID<br>REQUEST P2: accountID<br>REQUEST P3: couponID<br>REQUEST P4: couponStatus<br><br>RESPONSE P1: responderID<br>RESPONSE P2: transactionStatus | Sets the couponStatus of an issued coupon (as defined in section 4.2.8). Status can subsequently be used at the POS to determine whether the issued coupon is to be applied to the current transaction. |

**Table 73: Messages and parameters**

# 7 Security

## 7.1 Introduction

Value-added services transaction security can reduce the risk of fraud and theft of services and identities. A common security framework is necessary, using lightweight, well-known and widely-used security protocols. This section aims to define a security model that will meet the given business and regulatory requirements with minimal impact on performance. Security implementations can vary substantially from one solution to another and this is therefore out of scope. Future versions of this document will look at defining the security implementation.

This security model and policy section helps those designing, developing, testing, reviewing or auditing the system to understand how the system works in security terms at a high-level. It focuses on what data is to be secured (assets), where it lives (domains), and what the associated use cases are for this data (data flow). Additionally, threat modelling with attack trees complements the assets-domains-data flow view, giving a complete picture of the system's security. It allows security analysis of the system, and definition of the security requirements (independent of how those requirements are implemented). It is mainly focused on the contactless and handset interfaces (handset to contactless reader/terminal, handset to MNO wallet app server) and assumes that the acceptance system and the distribution channel provide the necessary protections.

The optimal implementation to meet these requirements is dependent on several factors such as hardware and software capabilities. It is a trade-off between the sensitivity of the data involved and the overheads of enabling interface security. This section summarises the different levels of security provided, and the risks and trade-offs for choosing each level.

Level 0 – No security

Level 1 – Entity authentication (VAS applet to MNO wallet app server, VAS applet to contactless reader)

Level 2 – Signed offers end-to-end, optional end-to-end encryption

Level 3 – Securing the NFC and over the air (OTA) channels

## 7.2 Requirements

The main requirements of the security system for coupons and loyalty systems are assumed to be:

- Providing a secure authenticated value-added services transaction
- Use of well-known key agreement methods
- Entity authentication
- Known-key security – unique session keys
- Minimal overhead.

## 7.3 Assets

Assets are sensitive data that need protection and can be classified with one of the classifications from Figure 58 to define the type of protection needed:

| Security Requirement | | Description | Security Level |
|---|---|---|---|
| C | Confidentiality | Privileged data that protects the system, not to be disclosed to unauthorised entities | Open (O)<br>Confidential (C)<br>Secret (S)<br>Top Secret (TS) |
| I | Integrity | What are the effects if the data is modified? Corrupted? Is it important to detect that the data was modified | High/ Medium/ Low |
| A | Authenticity | Authentication needed | Yes/ No/ Maybe |
| P | Privacy | Personal/ public sensitive information, not to be used to invade privacy | Yes/ No |

**Table 74: Key to protection classifications**

| Asset | C | I | A | P | Description |
|---|---|---|---|---|---|
| Payment like service | C | High | Yes | No | Monetary value such as coupons |
| High value service | C | High | Yes | No | e.g. above $X |
| Low value service | O/C | Low | Maybe | No | e.g. below $X |
| Customer details | C | Low | Yes | Yes | Customer's private data |
| Transaction details | C | High | Maybe | Yes | Amount; trans ID; type of goods |
| Wallet/account details | C | Yes | Yes | Yes | Wallet details, such as ID, wallet credentials |
| AID | O | No | No | No | Application ID |
| Secret keys | TS | High | Yes | N/A | Protection asset, not to be disclosed |
| Public keys | O | High | Yes | N/A | Secondary protection asset |
| PIN/passcode | TS | High | Yes | Yes | Protection asset, not to be disclosed |
| UN/nonce | O | High | Yes | N/A | Secondary protection asset |
| Authentication data | O | High | Yes | N/A | Secondary protection asset |

**Table 75: Assets**

## 7.4 Domains

The system can be broken into domains. A domain is a primary node on the asset communication path where assets are being kept or exchanged. In general, security of the communication between domains relies on the security of the domains themselves.

### 7.4.1 Terminal

| Store | Calculated/Generated | Transmit |
|---|---|---|
| Services | | Services |
| Payment-like data | | Payment-like data |
| Account, wallet data | | Account, wallet data |
| Authentication Data, i.e.: certificates | Authentication data, i.e.: challenge, Message Authentication Code (MAC) | Authentication data, i.e.: certificates |
| Transaction\redemption details | | Transaction\redemption details |
| Transaction logs and reports | | Transaction logs and reports |
| Public keys | | Public keys |

**Table 76: Terminal domain**

### 7.4.2 Contactless reader

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Configuration data | Authentication data i.e. challenge, MAC | Services |
| Merchant data | | Payment-like data |
| Authentication data, i.e. certificates | | Account, wallet data |
| Public keys | | Authentication data, i.e. certificates |
| Merchant data | | Transaction/redemption details |
| | | Transaction logs |
| | | Public keys |
| | | Merchant data |

**Table 77: Contactless reader domain**

### 7.4.3 Contactless reader secure area

Trusted computing base and secure storage

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Nonce, Authentication data | Nonce, Authentication data | Nonce, Authentication data |
| Keys | Keys | Cryptograms |
| Certificates | | Cipher text |

**Table 78: Terminal domain**

### 7.4.4 Contactless reader operating system (OS)

Trusted computing base and secure storage

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Nonce, Authentication data | Nonce, Authentication data | Nonce, Authentication data |
| Keys | Keys | Cryptograms |
| Certificates | | Cipher text |

**Table 79: Contactless reader OS domain**

### 7.4.5 VAS applet

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Services | | Services |
| Payment-like data | | Payment-like data |
| Account, wallet data | | Account, wallet data |
| Authentication data, i.e. certificates | Authentication data i.e. challenge, MAC | Authentication data, i.e. certificates |
| Transaction/redemption details | | Transaction/redemption details |
| | | Transaction logs and reports |
| Public keys | | Public keys |
| Secret keys | | |
| Mobile device/wallet PIN | | |

**Table 80: VAS applet domain**

### 7.4.6 Handset

App and OS

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Services | | Services |
| Account, wallet data | | Account, wallet data |
| Authentication data, i.e. certificates | | Authentication data, i.e. certificates |
| Transaction details | | |
| Transaction/redemption logs and reports | | |
| Public keys | | Public keys |

**Table 81: Handset domain**

### 7.4.7 MNO wallet app server

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Services | | Services |

| Store | Calculated/ Generated | Transmit |
|---|---|---|
| Payment-like data | | Payment-like data |
| Account, wallet data | | Account, wallet data |
| Authentication data, i.e. certificates | Authentication data i.e. challenge, MAC | Authentication data, i.e. certificates |
| Transaction\redemption details | | Transaction/redemption details |
| Transaction logs and reports | | Transaction logs and reports |
| Public keys | | Public keys |
| Secret keys (using Hardware Security Module) | | |

**Table 82: MNO wallet app server domain**

### 7.4.8 Back office

Out of scope

### 7.4.9 Point of sale

Out of scope

## 7.5 Interfaces – asset flows

This section describes the various system interfaces. Where assets are being exchanged between domains, the arrows describe the direction of the asset flow.

### 7.5.1 Asset flow

Figure 78 illustrates an asset flow from configuration to reporting on the acceptance side.

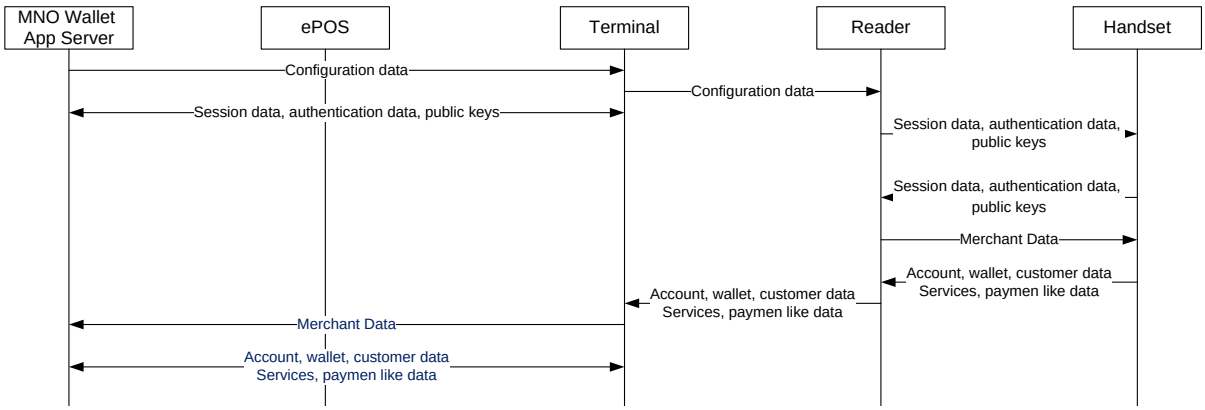


Figure 78**: Asset flow**

Note: the extra exchange for a backend base solution is presented in blue. In this case only merchant and account information is exchanged between the handset and the contactless reader.

Figure 79 illustrates an asset flow from configuration to reporting on the issuing side.

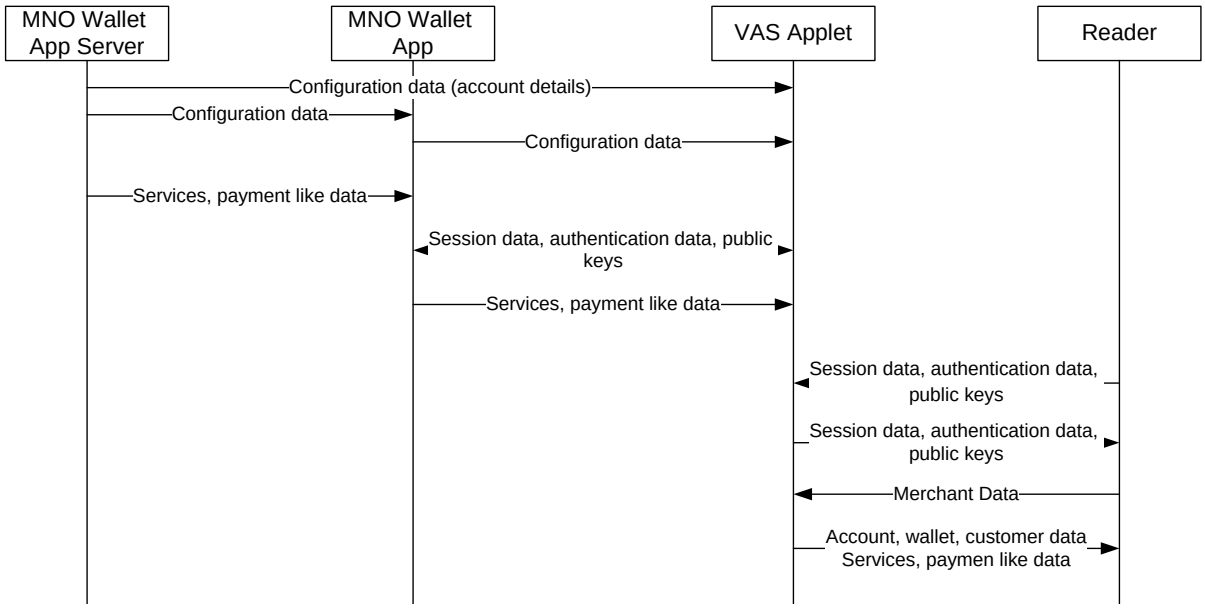


Figure 79**: Asset flow with reporting**

### 7.5.2 Interfaces

This section describe the different interfaces in which the assets flow and their direction

#### 7.5.2.1 MNO wallet app server - VAS applet (direct)

This connection is facilitated via the MNO wallet app and VAS Manager.

| Asset | MNO wallet app server to VAS applet | VAS applet to MNO wallet app server |
|---|---|---|
| Services | → | ← |
| Payment like data | → | |
| Account, wallet data | → | |
| Customer data | → | |
| Authentication data | → | ← |
| Session data | → | ← |
| Public keys | → | ← |
| Session keys | → | ← |
| Configuration data | → | |

**Table 83: Logical flow from MNO wallet app server to VAS applet**

#### 7.5.2.2 MNO wallet app server – handset (API 4/API 5)

| Asset | MNO wallet app server to handset app | Handset app to MNO wallet app server |
|---|---|---|
| Services | → | ← |

| Asset | MNO wallet app server to handset app | Handset app to MNO wallet app server |
|---|---|---|
| Customer data | → | |
| Merchant data | → | ← |
| Authentication data | → | ← |
| Session data | → | ← |
| Public keys | → | ← |
| Session keys | → | ← |
| Transaction\redemption data | → | ← |
| Transaction logs and reports | → | ← |
| Usage reports | | ← |

**Table 84: Logical flow from MNO wallet app server to handset**

#### 7.5.2.3 VAS applet – handset (API 2)

| Asset | VAS Applet to handset app | Handset app to VAS Applet |
|---|---|---|
| Services | → | ← |
| Payment-like data | → | |
| Merchant data | → | |
| Authentication data | → | ← |
| Session data | → | ← |
| Public keys | → | ← |
| Session keys | → | ← |
| Transaction\redemption data | → | |

**Table 85: Logical flow from VAS applet to handset**

#### 7.5.2.4 VAS applet – contactless reader (API 1)

| Asset | Handset to contactless reader | Contactless reader to handset |
|---|---|---|
| Services | → | ← |
| Payment-like data | → | |
| Account, wallet data | → | |
| Customer data | → | |
| Merchant data | | ← |
| Authentication data | → | ← |
| Session data | → | ← |
| Public keys | → | ← |
| Session keys | → | ← |

| Asset | Handset to contactless reader | Contactless reader to handset |
|---|---|---|
| Transaction/redemption data | | ← |

**Table 86: Logical flow from VAS applet to contactless reader**

#### 7.5.2.5 Contactless reader – terminal/POS (proprietary API)

| Asset | Contactless reader to terminal/POS | terminal/POS to Contactless reader |
|---|---|---|
| Services | → | ← |
| Payment-like data | → | |
| Account, wallet data | → | |
| Transaction details and usage data | | ← |
| Customer data | → | |
| Authentication data | → | ← |
| Public keys | → | ← |
| Configuration data | | ← |
| Transaction\redemption data | | ← |

**Table 87: Logical flow from contactless reader to POS**

#### 7.5.2.6 Terminal/POS – back office

| Asset | Terminal to back office | back office to terminal |
|---|---|---|
| Services | → | ← |
| Payment-like data | | ← |
| Account, wallet data | | ← |
| Authentication data | → | ← |
| Session data | → | ← |
| Public keys | → | ← |
| Transaction/redemption data | → | |
| Configuration data | | ← |

**Table 88: Logical flow from POS to back office**

## 7.6 Threats

This section briefly describes some of the threats of attack and offers various solutions that provide protection.

### 7.6.1 Fraud (fake wallet/service)

This attack can be performed by creating a fake wallet or service to receive offers or information at the point of interaction. Such an attack can lead to malicious capture and use of offers and customer data. However, any implementation of such an attack would be highly complex given the need to develop a fake wallet/service.

### 7.6.2 Electronic pickpocketing/skimming (fake contactless reader)

This attack can be performed by activating the wallet and requesting services and data without the customer's knowledge and/or permission. A fake contactless reader capable of authenticating to the wallet would be necessary for electronic pickpocketing/skimming.

### 7.6.3 Eavesdropping

This attack could be performed by placing a listening device within range of the two communicating devices (point of interaction) to retrieve data, for replay attacks and to fake a wallet or data. Alternatively, this attack could be performed by recording/sniffing the OTA link between the handset and the wallet app server. The impact of this threat can be varied depending on the asset being compromised. For example, eavesdropping is a concern when transmitting sensitive data, but might not be a threat if an attacker is only able to know the level of discount a consumer has received, as an example.

### 7.6.4 Data corruption

This attack would be performed by placing a jamming device within range (1 metre passive, 10 metres active) of the two communicating devices (point of interaction)

This attack is not too complicated, but it does not allow the attacker to manipulate the actual data. It is basically a Denial of Service attack

### 7.6.5 Denial of service

This attack would be performed by placing a Jamming device within range (1 metre passive, 10 metres active) of the two communicating devices (point of interaction).

### 7.6.6 Data modification

This attack would involve the attacker wanting the receiving device to receive some valid, but manipulated data. This is different from just data corruption. The feasibility of this attack significantly depends on the applied strength of the amplitude modulation.

See Section 7.6.7 for man-in-the-middle attack for data modification over-the-air.

### 7.6.7 Man-in-the-middle

This attack is a form of active eavesdropping by making independent connections with the two parties relaying the messages between them, making them believe they are communicating with each other. It is impractical to mount a man-in-the-middle attack in a real-word scenario, either via NFC or OTA channels.

### 7.6.8 Replay

This attack would be performed by recording and replaying a transaction at a point of interaction to receive offers or information without the customer authorisation or knowledge.

### 7.6.9 Relay

This attack can be performed by placing a proxy contactless reader that can relay the messages to a proxy handset at a different point of interaction to receive offers or information without the customer authorisation or knowledge.

A relay attack can be seen as a simple range extension of the contactless communication channel.

#### 7.6.10 Pharming

This attack would be performed by redirecting traffic to a different website

#### 7.6.11 Web spoofing

This attack would be performed by sending web pages or scripts from a fraudulent website (to impersonate an actual website).

### 7.7 Countermeasures

#### 7.7.1 Wallet activation control

This measure uses wallet PIN/passcode notifications to prevent establishing a session with unauthorised contactless reader.

#### 7.7.2 Detection

This measure checks radio-frequency signals while sending data, to detect a data modification/corruption attack.

#### 7.7.3 Transaction/session specific data

This measure uses session-specific data/dynamic data to prevent replay attacks.

#### 7.7.4 Wallet authentication

This measure uses authentication to prevent establishing sessions with unauthorised wallets.

#### 7.7.5 Contactless reader authentication

This measure is used to prevent information sent to unauthorised contactless readers.

#### 7.7.6 Wallet app server authentication

This measure is used to prevent sessions established with unauthorised wallet app server.

#### 7.7.7 Dynamic authentication

Dynamic authentication is used to prevent relay and replay attacks.

#### 7.7.8 Data authentication

Data authentication is use to determine whether the data source is trusted.

#### 7.7.9 Encryption

Encryption is used to prevent eavesdropping and replay attacks.

## 7.8 Security level cost versus benefit analysis

This section describes the possible levels of security. It presents the different levels and available options based on the security requirements and the type of protections implemented on the application.

The appropriate security level is defined by the scheme introduced in section 7.1. Security requirements for each level are indicated in Table 89.

| Benefits | | | | | | |
|---|---|---|---|---|---|---|
| Levels | Wallet Auth | Contactless reader/Server Auth | Dynamic data unique session data | Encryption, unique session keys | Data auth/MAC'ing | Wallet activation control |
| 0 | No | No | No | No | No | No |
| 1 | Yes | Yes | Yes | No | No | Yes |
| 2 | Maybe | Maybe | Yes | Yes | Yes | Yes |
| 3 | Yes | Yes | Yes | Yes | Yes | Yes |

**Table 89: Security requirements analysis**

Relative security costs for each level are indicated as high (H), medium (M) and low (L) in Table 90.

| Costs | | | | | | |
|---|---|---|---|---|---|---|
| Levels | Storage e.g. certificates and other authentication data | Secure Storage e.g. secret/private keys | Data transfer size e.g. authentication data, certificates and public keys | Data transfer speed e.g. encryption, authentication, key exchange | Computation e.g. key generation, computation, signing | Logistics e.g. managing PKI, certificates, key pairs |
| 0 | L | L | L | L | L | L |
| 1 | M | M | M | M | M | M |
| 2 | L | L | M | L | L | H |
| 3 | H | H | M | H | H | H |

**Table 90: Relative security costs**

## 7.9 Attack trees

This section provides structural diagrams on how an attack could occur. This section is only for identifying reference attack trees and does not intend to provide any counter measures. Details on counter measures are out of scope of the present document.

Note to read attack trees:

34. Read from the top-down
35. One top node defines the attack objective (red)
36. The sub-nodes detail how this attack objective can be achieved (blue)
37. A successful attack path is from the top of the figure to any node on the next-to-bottom of a branch
38. The bottom nodes define the countermeasures for each of the attack end nodes (green)

### 7.9.1 Get service data

Figure 80 presents the attack tree to steal service data or obtain unauthorised service access, such as stealing a coupon or gift card from the wallet.

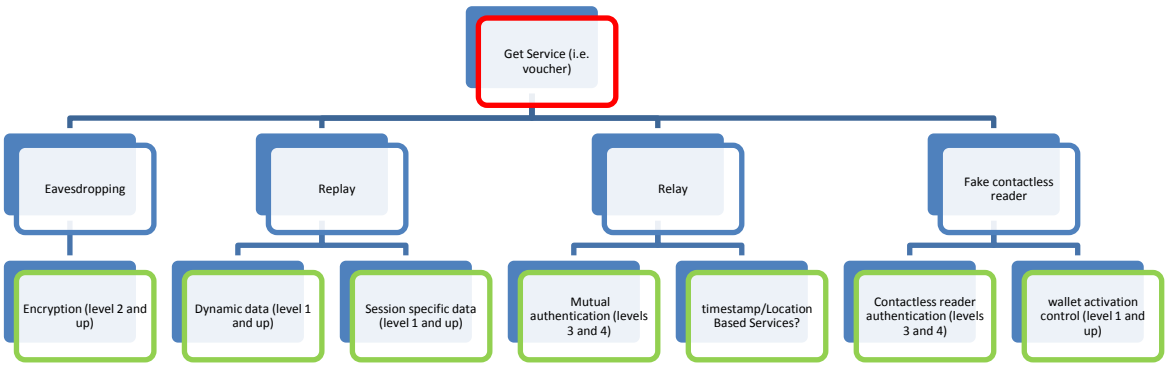


Figure 80**: Get service data – attack tree**

# Annex A An example of coupons, loyalty cards and their acceptance

The requirements of coupon and loyalty acceptance are quite complex and rely on relationships between merchants, FMCG companies and loyalty platform implementers. The table below illustrates an example of combinations of merchants and the coupons and loyalty schemes that they could accept.

| VAS Object Issued | | | | | Awarder | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Issuer | Issuer Type | Offer | Object to Present | Notes | Supermarket A | Supermarket B | Supermarket C | Catalogue Shop | Department Store Chain A | Computer Retail Chain | Shoe Company (High Street Outlet) | Department Store Chain B | Newsagent Chain | Car & Cycle Parts Specialist | Car Service Chain A | Car Service Chain B | Pizza Chain | Alternative Pizza Chain | High Street Chemist | Mother & baby specialist Store |
| Soft Drink Inc | FMCG | 10% off any SD Inc cola | Coupon | | Y | Y | Y | | | | | Y | Y | | | | | | Y | |
| Toilet Paper Ltd | FMCG | 18 rolls of TPL manufacturer toilet paper for £9 | Coupon | | Y | Y | Y | | | | | Y | | | | | | | | |
| Black Tyres | FMCG | 4 for 3 on winter weather car tyres | Coupon | | Y | | | | | | | | | Y | Y | Y | | | | |
| Black Tyres | FMCG | 20% bike tyres | Coupon | | Y | | | Y | | | | | | Y | | | | | | |
| Pizza Chain (Cafe) | Merchant | 3 courses for £15 | Coupon | Cafe only | | | | | | | | | | | | | Y | | | |
| Pizza Chain (Manufacturer) | FMCG | Buy 1, get 1 free on 12" PC Pizza | Coupon | Stores Only | Y | Y | Y | | | | | | | | | | | | | |
| Shoe Company | Merchant & FMCG | 25% SC city shoes | Coupon | | Y | Y | Y | | Y | | Y | Y | | | | | | | | |
| Supermarket A | Merchant | Collect points on purchases | Loyalty ID | SMA Scheme | Y | | | | | | | | | | | | | | | |
| Supermarket B | Merchant | 20p off 2lt SD Inc cola | Loyalty ID & Coupon | SMB bespoke | Y | | | | | | | | | | | | | | | |
| Supermarket C | Merchant | £1.50 off next purchase | Loyalty ID & Coupon | SMC bespoke | | | Y | | | | | | | | | | | | | |
| High Street Chemist Ltd | Merchant | HSC points on purchases | Loyalty ID | | | | | | | | | | | | | | | | Y | |
| High Street Chemist Ltd | Merchant | 10% HSC own-manufacturer cosmetics | Coupon | | | | | | | | | | | | | | | | Y | |
| Big Loyalty Scheme | Loyalty Scheme | BLS points on purchases | Loyalty ID | >100 merchants | | | Y | Y | Y | Y | Y | | | Y | | | | Y | | Y |

# Annex B GS1 Digital Coupon Management System diagrams

## B.1 Overall Use Case View

Figure 81 shows the complete use cases for GS1 Digital Coupon management system [4]. The ones highlighted in Red have been covered in this Technical Proposal.

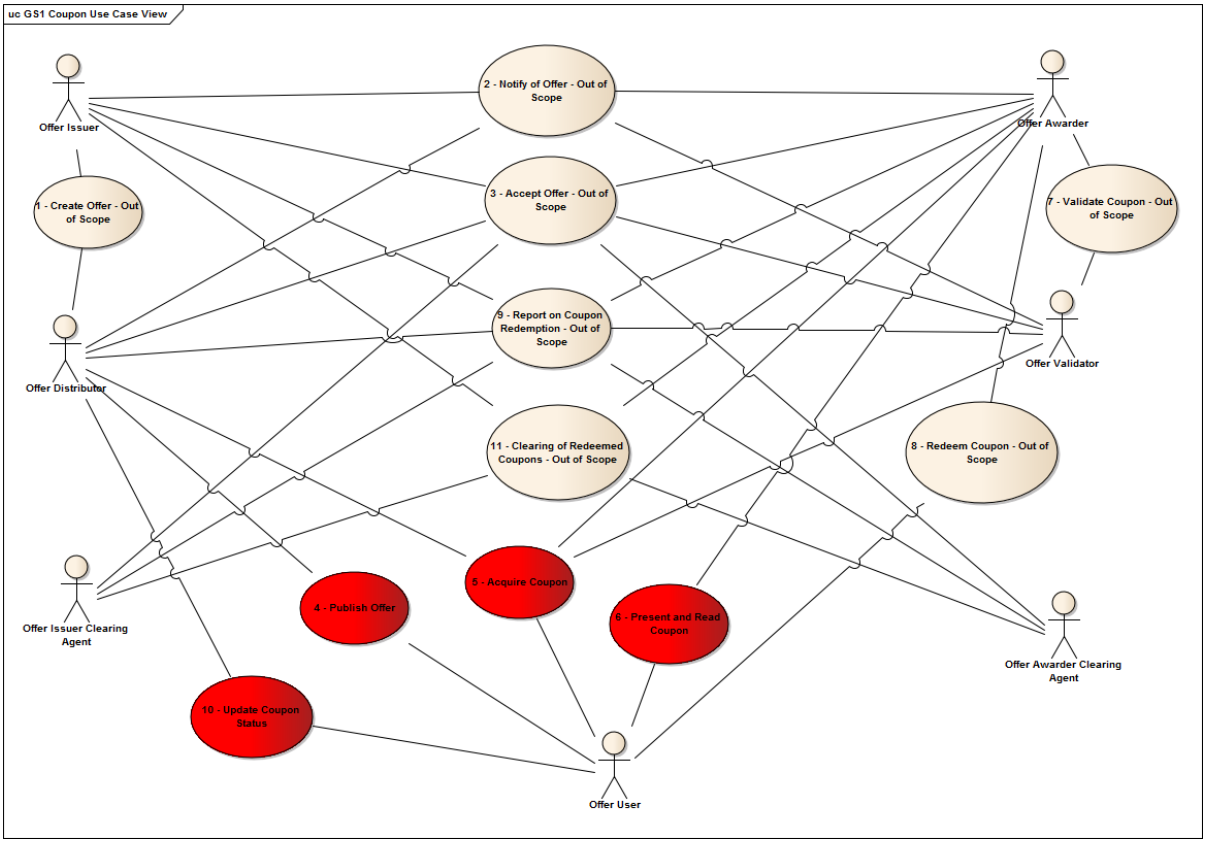


Figure 81**: Overall Use Case for GS1 Digital Coupon Management System**

# Annex C VAS Applet data structure

In this section is the description of the various tags used within API 1 and API 2.

## C.1 Basic type

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

### C.1.1 UniqueID

UniqueID is the unique identifier of a token, needed for later use by the VAS applet or VAS manager.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x20 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.2 TokenData

TokenData is the transparent token data, which is not interpreted by VAS applet but instead by the retail system.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x21 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.3 WalletData

WalletData is interpreted by the VAS manager to get the loyalty and coupons presented in the wallet.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x22 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.4 ActionSpecifier

ActionSpecifier describes the action, which should be performed on an item with a PUT VAS DATA command (see section 6.1.4.3).
Such as Create (0x00), Delete (0x01), Update (0x02), mark as Redeem (0x03) and Invalidate (0x04).

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x23 |
| LEN | | variable | |

| Field | Type | Length | Value |
|---|---|---|---|
| VALUE | Byte | variable | |

### C.1.5 ProtocolVersion

ProtocolVersion contains version information for the used protocol.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x24 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.6 VASAppID

VASAppID is the unique identification of the UICC application “VAS Applet”. This ID is used by the terminal to select the application. (See section 6.1.4.1 and 6.2.4.1)

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x25 |
| LEN | | 11-16 | |
| VALUE | Byte | variable | |

### C.1.7 SecuritySpecifier

SecuritySpecifier describes the type of security to be applied for a particular token.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x26 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.8 Signature

Signature data type is the digital signature for the token data. This is an optional entity, and comes from a previous version of the document. It is implementation specific and could be used to enable offline coupons. Implementations can also ignore it and will not break if a coupon is presented with it.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x27 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.9 MerchantID

MerchantID is the unique identification of a merchant. This ID is used by the VAS applet as one of the filter criteria to transfer the applicable coupon data to the target contactless reader. Note only the meta-data-structure of the `MerchantID` is defined in TLV format, and it is open to adopt any standard for its data. For example, use the GS1 Digital Coupon Standards [4]. Based on the implementation requirements, this could be completed as lists or through capability within VAS Manager.

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x28 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.10 LoyaltyIssuerID

LoyaltyIssuerID is the unique identification number of a loyalty program. This ID is used by the VAS applet as one of the filter criteria to transfer the applicable loyalty data to the Contactless Reader. Note only the meta-data-structure of the `LoyaltyIssuerID` is defined in TLV format, and it is open to adopt any standard for its data. For example, use the GS1 Digital Coupon Standards [4].

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x29 |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.11 CouponIssuerID

CouponIssuerID is the unique identification number of a couponing issuer. This ID is used by the VAS applet as one of the filter criteria to transfer the applicable coupon data to the contactless reader. Note only the meta-data-structure of the `CouponIssuerID` is defined in TLV format, and it is open to adopt any standard for its data. For example, use the relevant GS1 standards [4].

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x2A |
| LEN | | variable | |
| VALUE | Byte | variable | |

### C.1.12 VASAppletCapabilities

The data packet defining the VAS applet capabilities

| Field | Type | Length | Value |
|---|---|---|---|
| TAG | Byte | 2 | 0x9F:0x2C |
| LEN | | variable | |
| VALUE | Byte | variable | |

## C.2 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.

### C.2.1 Token

Token refers to the collection of all data, related to a certain token.

| Field | Type | Length | Value | Mand./Opt. |
|---|---|---|---|---|
| TAG | Byte | 1 | 0xB0 | M |
| LEN | | variable | | M |
| | UniqueID | variable | | M |
| | TokenData | variable | | M |
| | WalletData | variable | | O |
| Further optional data | …. | | | O |

### C.2.2 CouponTokenList

This table refers to the list of coupon tokens.

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

### C.2.3 LoyaltyTokenList

This table refers to the list of loyalty tokens.

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

### C.2.4 Tag list

This table refers to the list of Tags

| Field | Type | Length | Value | Mand./Opt. |
|---|---|---|---|---|
| TAG | Byte | 1 | 0x5C | M |
| LEN | | variable | | M |
| List of tags required | Bytes | variable | | O |

# Annex D Inter-process Communication – Android OS

## D.1 Implementation suggestions

IPC should be used for communication between the VAS manager and MNO wallet app. The availability of IPC mechanisms varies depending on the target operating system, with Android providing the best support. Bound services in Android offer the ability to directly expose functions to other applications running on the mobile device. While bound services follow the conventional IPC pattern, it is a complex pattern to implement. A less complex option is to implement a broadcast receiver, another pattern used in Android to notify applications of events. Custom events could be defined by the VAS manager and MNO wallet app. In addition to the notification of events, data objects can be attached to these events allowing IPC-like activity.

## D.2 Security considerations

For both IPC options in this section, there are concerns around authenticated callers. Android has a basic security mode whereby only apps signed using the same key can communicate. If the VAS manager and MNO wallet app are released (signed) by the same entity, such as the MNO, then this model will provide the necessary security. If third-party wallet providers are to integrate with the VAS manager (MNO signed) then this model will not apply and a proprietary method of authentication will be needed, which is beyond the scope of this document.

# Document management

## Document history

| Version | Date | Brief Description of Change | Approval Authority | Editor/Company |
|---|---|---|---|---|
| 1.0 | 27/01/2014 | New proposed non-binding permanent reference document submitted to document approval process. | Mobile Commerce Programme, PSMC | Saurabh Sethi/GSMA. Contributors: Aimia, Deutsche Telekom, Escher Group, Proxama Ltd, The Logic Group, VeriFone |

## Other information

| Type | Description |
|---|---|
| Document Owner | Mobile Commerce Programme, Retail Project |
| Editor/Company | Saurabh Sethi/GSMA |

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

Comments, suggestions and questions are always welcome.