Open Issues and/or Actions

|  |  |
| --- | --- |
| **Open Issue** | **Issue description** |
| Physical Subscription | Hybris is not ready with the definition of the subscription architecture |
| Single SignOn | Ability to sign-in once for app and Philips Flagship store |
| Digital Subscription | The arhcitecuture and the backend definition is still open |
|  |  |
|  |  |
|  |  |

Table of Contents

[1 DOCUMENT INTRODUCTION 3](#_Toc480358016)

[1.1 Purpose 3](#_Toc480358017)

[1.2 Scope 3](#_Toc480358018)

[1.3 References 3](#_Toc480358019)

[1.4 Terminology & Abbreviations 3](#_Toc480358020)

[2 Overview 4](#_Toc480358021)

[3 Architecture 4](#_Toc480358022)

[3.1 Architectural Overview 4](#_Toc480358023)

[3.2 Architecture Principles 4](#_Toc480358024)

[3.3 Architecture Views 5](#_Toc480358025)

[3.3.1 Customer View of the component 5](#_Toc480358026)

[3.3.2 Application view 9](#_Toc480358027)

[3.3.3 Application Architecture 10](#_Toc480358028)

[4 Allocation of Quality Aspects 11](#_Toc480358029)

[5 Design Details 12](#_Toc480358030)

[5.1 Elements 12](#_Toc480358031)

[5.1.1 Software 12](#_Toc480358032)

[5.1.2 Sequence diagram for different scenerios 13](#_Toc480358033)

[5.2 External Interfaces 17](#_Toc480358034)

[5.2.1 External interface -1 18](#_Toc480358035)

[5.2.2 External interface -2 18](#_Toc480358036)

[5.3 Internal Interfaces - NA 18](#_Toc480358037)

[6 Revision History 19](#_Toc480358038)

[7 Approval 19](#_Toc480358039)

# DOCUMENT INTRODUCTION

## Purpose

This document describes the technical design of InApp Purchase.

## Scope

The aim of this document is to present the architecture of the InApp Purchase common component. It will contain different views of this architecture and different modules needed to realize this. This is a living document meaning that it is far from complete as the first customer will be the Tuscany project and the document is aimed at supporting the MVP for this. When functionality is added this document will grow.

## References

| **Reference** | **Identification** | **Title / additional remarks** |
| --- | --- | --- |
| Hybris REST/JSON API | -- | Latest doc can be accesed from the Hybris documentation page |
| OCC API documentation | -- | Latest doc can be accesed from the Hybris API documentation page |
| IAP000008 | Integration Document | iOS Integration document with API information |
| IAP000009 | Integration Document | Android Integration document with API information |
| IAP000001 | Requirement Doc | In App Purchase Requirement Document |
| PLL000005 | Software Architecture Design | Software Architecture Design – CDPP Platform |

## Terminology & Abbreviations

| **Terminology & Abbreviations** | **Description/Definition** |
| --- | --- |
| IAP | In App Purchase |
| OCC | [Omni Commerce Connect](https://www.hybris.com/en/release46) |
| PCI | Payment Card Industry |

# Overview

The InApp Purchase, is a common component that can be plugged into any vertical app. This component provides the vertical with the ability to buy the products from the app, thereby giving in app purchase expericen to the user.

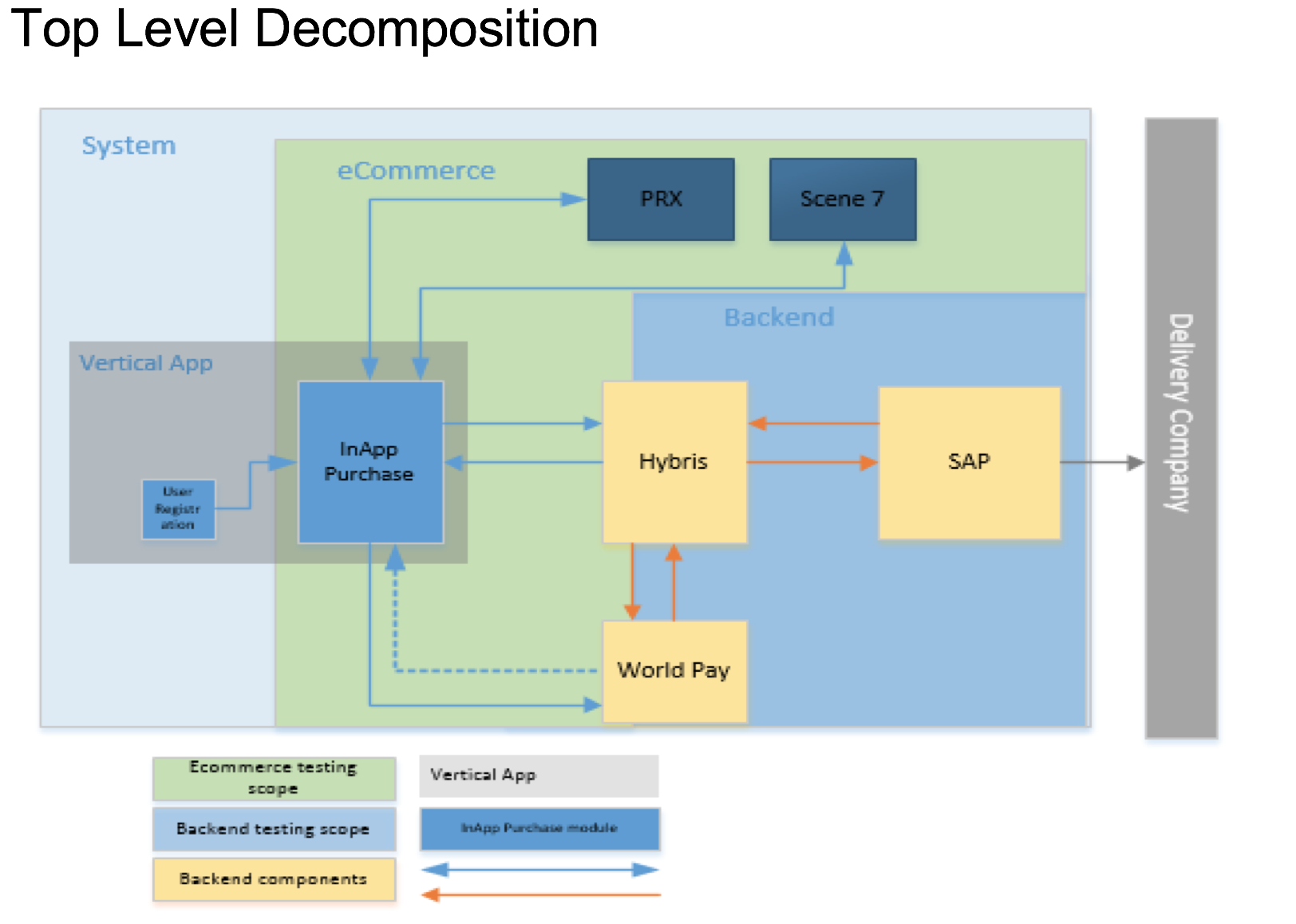
User get the option to purchase the product in the app, along with that user also have the option to purchase the products from the retailers like Amazon, Target etc.

The In App Purchase component is based on the underlying support of occ/Hybris which act as the one point gateway to the server which has the store information. All the communication are based on the RESTFul api’s and uses Philips standard registration mechanishm for the authentication and token generation.

Propositions has the ability to consume the same by mere settting up the store at the backend followed by the needed configuration, detailed under getting started with eCommerce.

# Architecture

## Architectural Overview

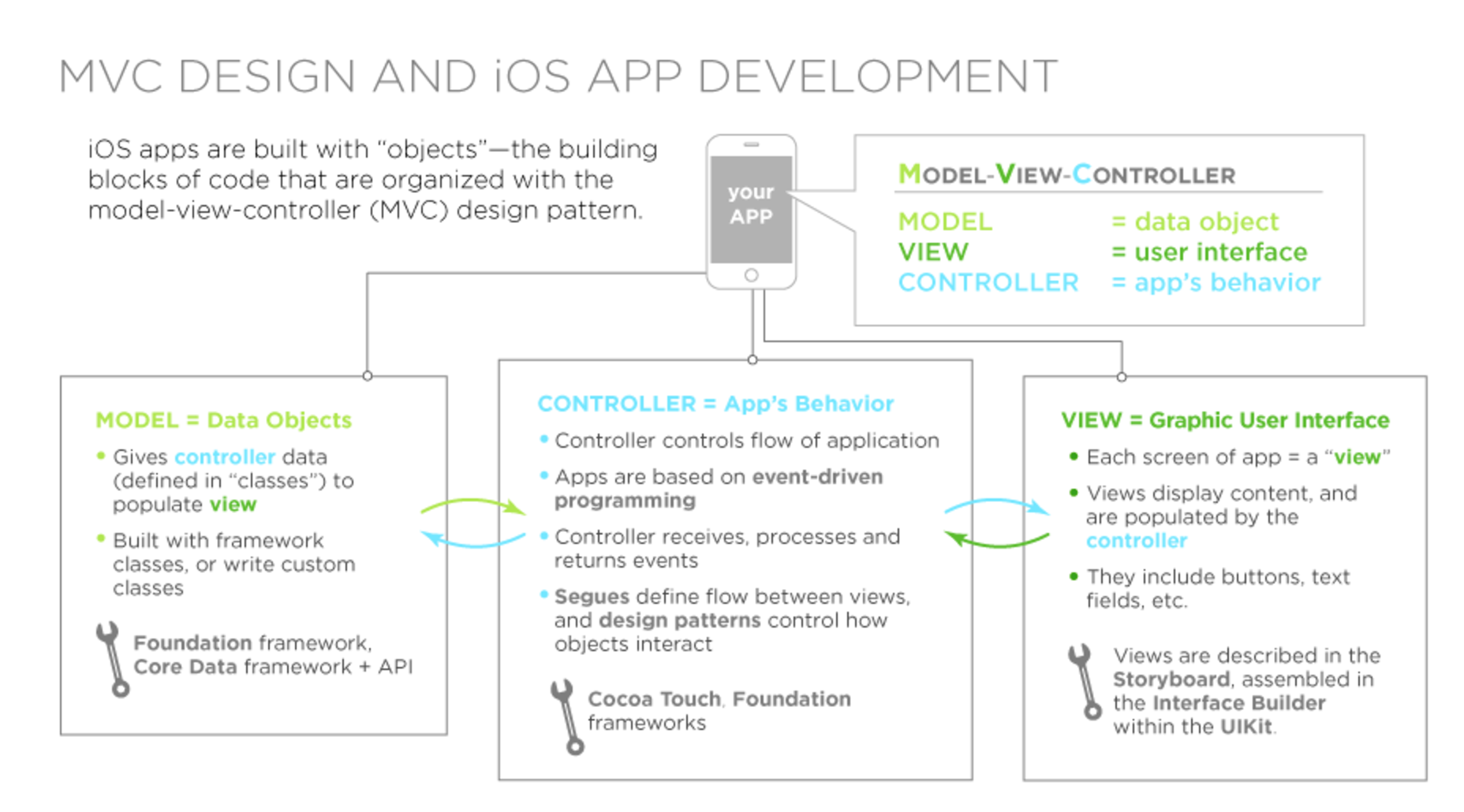
**

## Architecture Principles

*As we know Mobile application development share common software design principles with the underlying Operating system ( Android and iOS ) in this case as well we are following the same.*

***Model View Controller (MVC)***

*The Model View Controller (MVC) pattern is a very popular approach for the development of a mobile application.*

******

***Layered Abstraction -*** The development would follow the layered Abstraction in a sense the UI will not be directly talking to the layer that is responsible for making the API calls.

The basic **SOLID** design principles during the course of development wherever applicable.Which states Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.

## Architecture Views

### Customer View of the component

This component will be created as a common component and therefore be used by different propositions. The component will live in the mobile app developed for the proposition.



Figure 1 Customer view

This component will provide a true in app commerce functionality. The use cases it must support are:

* In app purchasing of replacement items. E.g. replacement brush heads for a toothbrush
* In app purchase of accessories. E.g. products/accessories related to a toothbrush
* Purchasing replacement items with a one-click buy option.

#### Philips Store

There are different stores, as a rule of thumb for each country there is a store. Reasons for this are different currencies, tax laws and products that we as Philips have to offer to this market (not all products are available in all regions). There are naturally exceptions for this e.g. Luxembourg. This country makes use of the French store because of population size and culture similarities. There are also countries where we as Philips do not do business for political reasons. When trying to use the Philips store from one of those countries an error will be presented to the user that Philips does not ship to that country.

The Philips store backend that will be used for the in app purchasing is the same backend used for the Philips store on the internet.

For every sales area an URL exist where the store is located. For example the US store is located at: http://www.philips-store.com/ and the UK store is located at: http://www.philips-shop.co.uk/

The specific proposition stores will be a site on this URL. E.g. [http://www.philips-store.com/<SITEID>/](http://www.philips-store.com/%3cSITEID%3e/), where SITEID indicates the proposition.

The interface will be REST/JSON and is documented on a WIKI1.

#### Payment methods

Supported payment methods must be the ones that are custom for the country/sales area. General payment methods are credit cards and PayPal. Certain countries can have additional custom methods e.g. iDEAL in the Netherlands.

For ease of reading the word credit cards is used in the rest of the document but this can also include PayPal or country specific payment methods.

#### One click buy scenario

To make it easier and faster for the customer it would be nice if the user can just buy a product by simply clicking the buy button on a page displaying the product and a description. This scenario will be part of the e-commerce component.

To process an order and to deliver the goods certain information is needed:

* Creditcard or PayPal information (iDEAL currently does not support the one click buy scenario)
* Delivery/shipping address
* Delivery mode (method)
* Billing address

The user has to enter this information at least once. Every payment is stored in the history of WorldPay, this includes these payment details (credit card information and billing address). The delivery address and mode is stored in hybris.

For the one click buy scenario these payment details must be retrieved from a previous payment in WorldPay and set on the new pay request. From the previous order in hybris the delivery address and the delivery mode can be retrieved and be set on the new order. In this scenario the flow can now proceed with no user interaction after he/she clicks the buy button. Having an additional overview and asking for confirmation is off course good practice.

When the creditcard in expired or the PayPal account no longer valid an error will be returned from WorldPay this should trigger the normal scenario where the user/customer has to enter which payment method he/she wants to use.

#### Native Apple store and Google play possibilities

Apple and google both exclude physical goods from in app purchasing through either the app store or the play store. This means that no physical goods can be sold through the app store. However when selling additional features or functionality of the app, meaning any digital goods, then only the app store is allowed. This means it is not allowed to sell these using an own store.

An exception is when digital good can be used outside the app, then it may be sold using a system outside the app/play store. An example of these digital products are:

* E-books in pdf ( or any industry standard format)
* Music in an industry standard format
* Movies in an industry standard format

On everything sold through either the play store or the app store the revenue is shared (70/30) with either google or apple.

Below are actual parts from the App store and google play store guidelines explaining this in more detail.

##### App store review guidelines:

Apps that unlock or enable additional features or functionality with mechanisms other than the App Store will be rejected

Apps utilizing a system other than the In-App Purchase API (IAP) to purchase content, functionality, or services in an App will be rejected.

##### Google play store guidelines

**Paid and Free Apps**

In-store purchases: Developers charging for apps and downloads from Google Play must use Google Play’s payment system.

In-app purchases:

* Developers offering products within a game downloaded from Google Play or providing access to game content must use Google Play In-app billing as the method of payment.
* Developers offering products within another category of app downloaded from Google Play must use Google Play In-app Billing as the method of payment, except:
  + where payment is solely for physical products; or
  + Where payment is for digital content that may be consumed outside of the app itself (e.g., buying songs that can be played on other music players).

A list of examples describing appropriate use cases for Google Play In-app Billing is available in the Google Play Developer Help Center.

* In-app virtual currencies must only be used within the app where they were first purchased.
* Developers must not mislead users about the apps they are selling nor about any in-app services, goods, content or functionality they are selling. If your product description on Google Play refers to in-app features to which a specific or additional charge applies, your description must clearly notify users that payment is required to access those features.

**Subscriptions and Cancellations**

Google's subscription cancellation policy is that a user will not receive a refund for the current billing period when canceling a subscription, but will continue to receive issues and updates of the relevant subscription content (if any) for the remainder of the billing period, regardless of the cancellation.

You (as the content or access provider) may implement a more flexible refund policy with your users directly, and it is your responsibility to notify your users of those policies and ensure that the policies comply with applicable law.

### Application view



Figure 2 Context diagram

#### PRX

PRX will provide the url of the store to use based on the location and language. PRX also contains the details of the products. These details consists of a descriptive text about the product concerned, which is accesses via product CTN. PRX is part of PIL.

#### Scene7

Scene7 is a system of the PIL landscape that contains images of different sizes. From Scene7 an image of the product can be downloaded.

#### Philips/store Hybris

This is the actual backend of the flagship store. It is accessible through a REST/JSON interface. It contains functionality for creating shopping carts, adding items to shopping carts, placing orders, retrieving product lists etc. For details of the containing functionality see Chapter **Error! Reference source not found.** **Error! Reference source not found.**.

#### WorldPay

Worldpay is the partner of Philips for handling the payments. For handling credit card data like credit card numbers and expiration dates an application needs to be PCI (see **Error! Reference source not found.**) compliant. To avoid this compliancy an external party (like WorldPay) is needed.

World pay will retain a history of previous payments.

#### FIL SAP

This is a system for creating product catalogs, product descriptions, prizes etc. This is an administrative system that is used by a Business/marketing expert for maintaining product lists, product details, prices, campaigns etc.

The in app purchase will not need a change here. We use what is already in place.

#### SAP

This is the database system that contains the product lists and information including the prices. It also holds records of the orders and where to ship these.

The in app purchase will not need a change here. We use what is already in place.

#### Current transformation from ATG to Hybris deployment

Presently the Philips flagship store is deployed on a different system then Hybris, namely ATG. This does not support in app purchasing only through app web based purchasing. At the moment ATG is being replaced by Hybris.

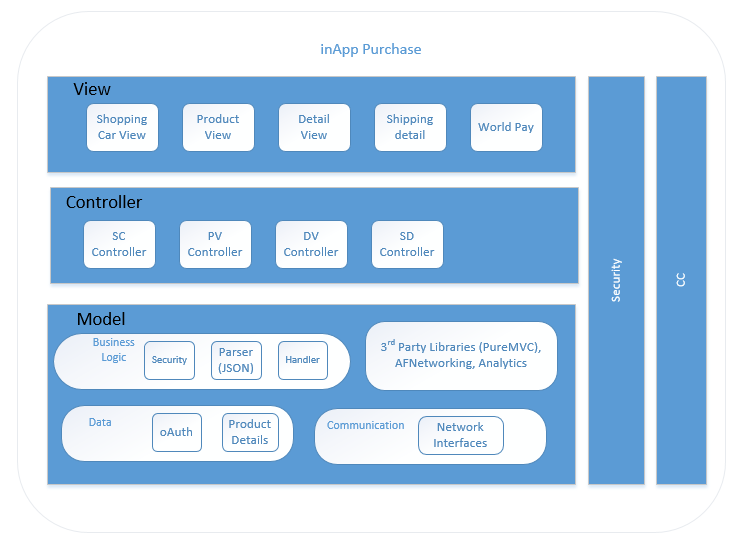
The underlying systems namely SAP will not change this means that the existing systems will remain to be used for the in app purchasing. For example, creating product catalogs, setting and adapting prices creating campaigns, vouchers etc.

### Application Architecture

#### App Architecture

Below Image depicts the architecture of the application.

#### Layering Model



#### Design Method

The design used Object Oriented concepts which are supported by majority of the platforms like Swift, Objective C and Java. Solid design principle will be followed along with design patterns like Façade, Delegation, Notification and Extensions are used.

All communication with server (Hybris, PRX, WorldPay) will be using SSL certificates. Plain HTTP communication will be used only to download the images from the Scene7.

The components follow the layered approach. Following rules are enforced for how the layers interact with each other.

* One way interaction (Top to down): Higher level layers can interact with layers below, but lower layers cannot interact with layers above. This is to avoid circular dependencies between layers.

# Allocation of Quality Aspects

* *The app size should not be more than 2 MB, if its more valid justification needs to be provided in.*
* *Performance- API should respond within 1 sec*
* *Memory Usage should be done judiciously*

# Design Details

## Elements

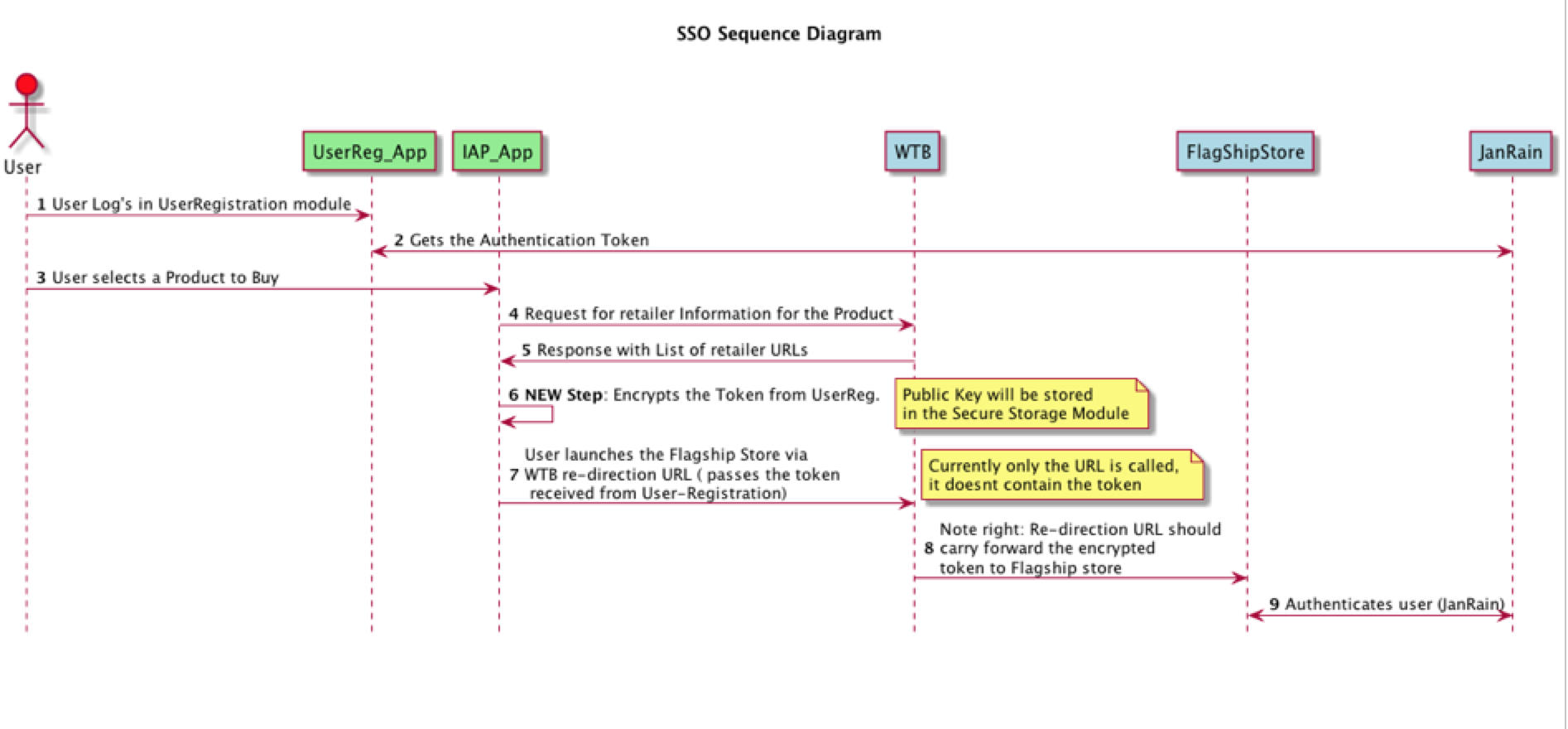
### Software

* *Inititalization – the IAP component must be inititalize on the successful initialization of User Registration*
* *Memory management – No information must be stored in the phone, all of them will be cached and once the user logs out the same must be deleted. Card information not be be retained anywhere, not even cached.*
* *Logging, retrieving, and storing data – App infra component shall be used for logging. Where as no data shall be stored back on the phone.*
* *Exception handling – When the user in the IAP component, the exceptions must be handled.*
* *User authentication – User will be authenticated based on the token passed from the registration module. Backend, hybris will be autneticating the user ( in backend with conjuction with JanRain/hsdp), post that will be providing the app with autenthication token and the refresh token. This will be used further for the communication ( RESTFul API ) with the server*
* *Self-test – Automation will be created to verify the application on the API and UI level*
* *Security – Ensure no user information is stored in the app,* 
  + *All the calls to the server should be done under https*
  + *Hybris provided/ approved certificates should be enabled the application i.e no bypass of the certificates*
  + *Security assessment ( static and dynamic should be carried out at regular*
* *SOUP - its covered under the another sheet of Open source license*

#### Single Sign on

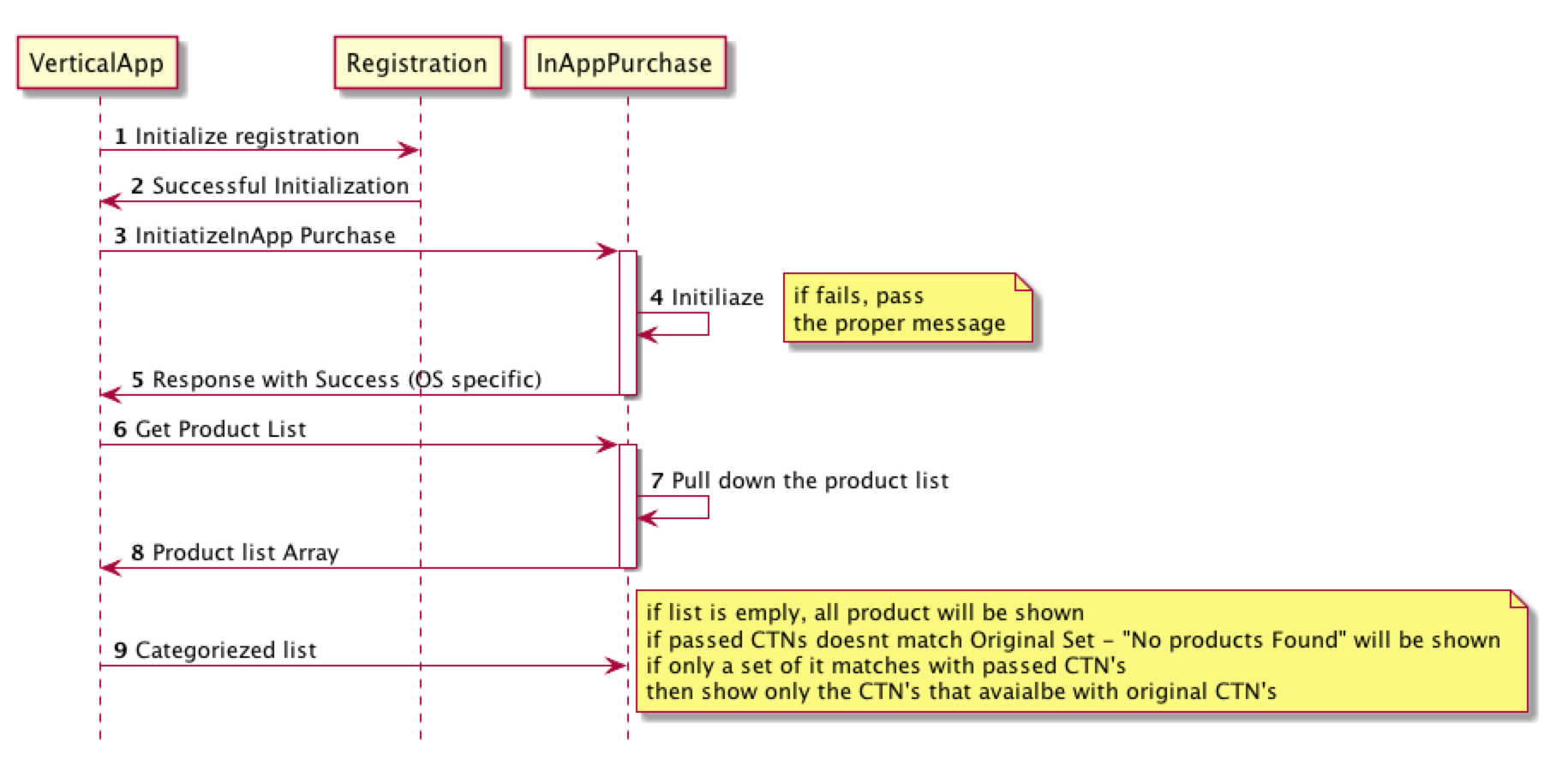
#### Design Constraints

Single Sign On – Allowing user to signon once using the Philips registration and able to buy the products on the Philips Store. Below architecture would be required for the implementation of the same.

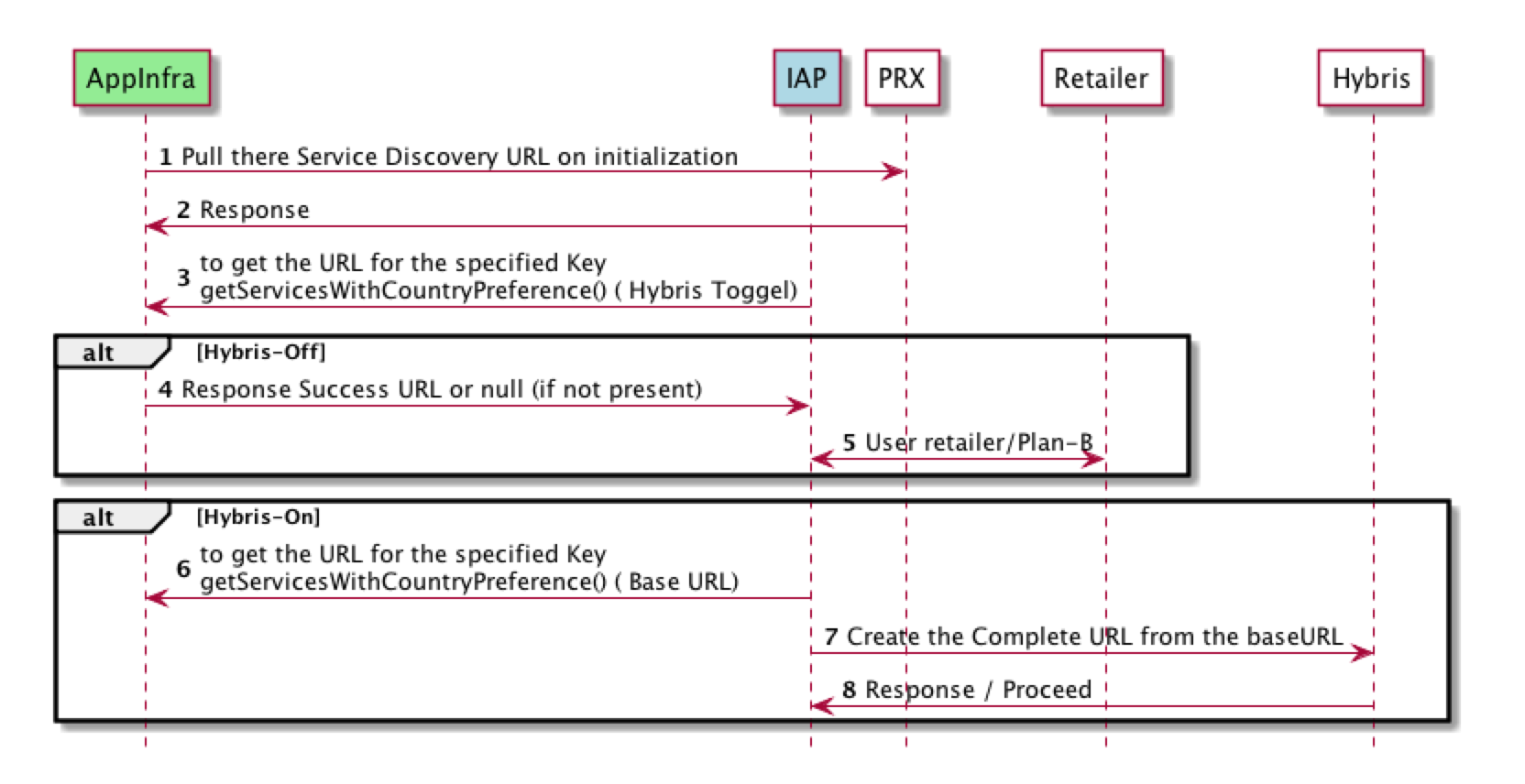


### Sequence diagram for different scenerios

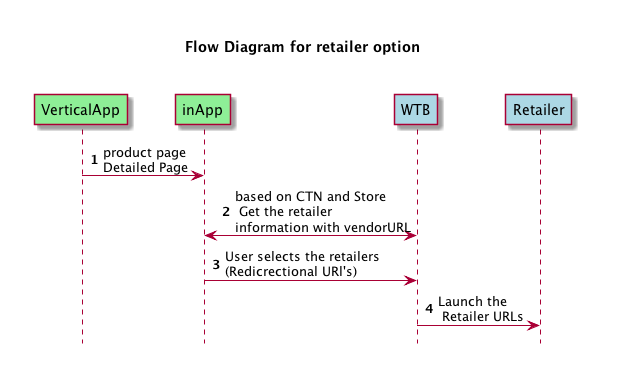
#### Initiatlization



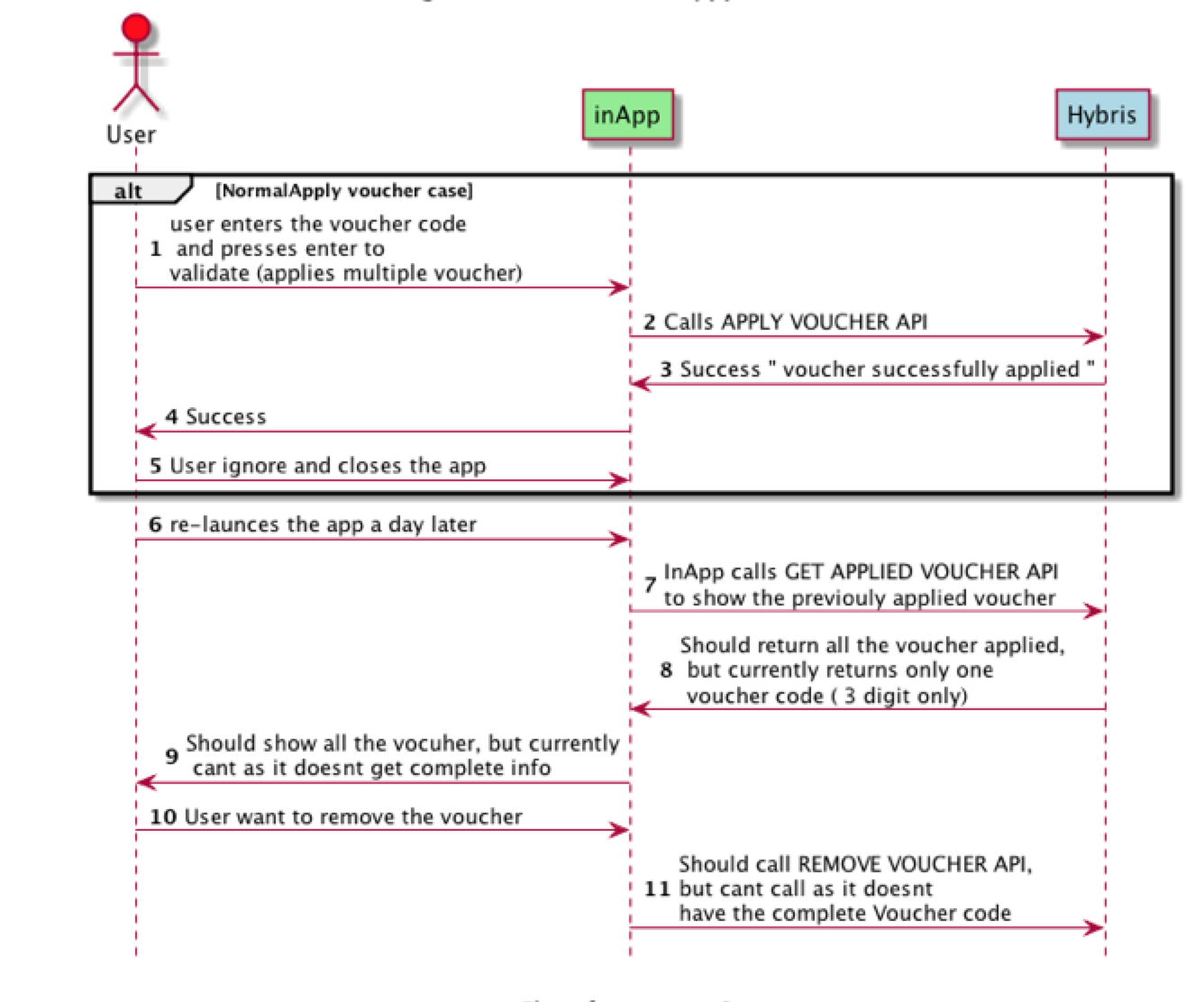
#### Service Dsicovery



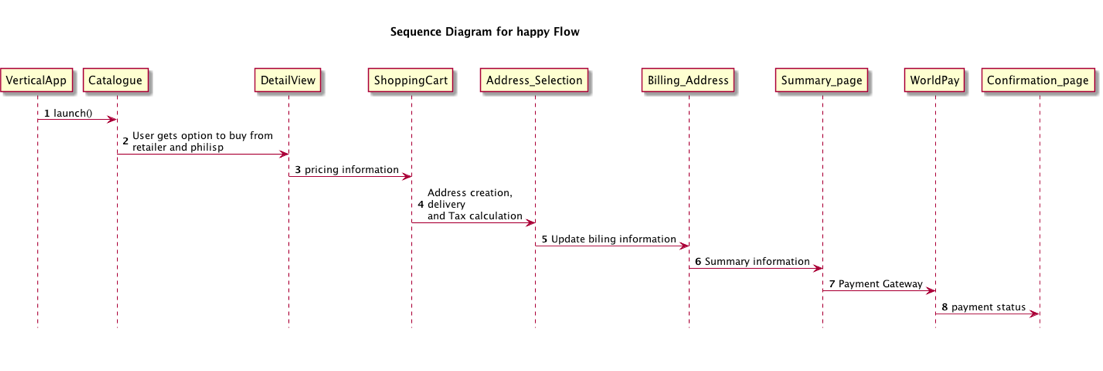
#### Retailer Flow



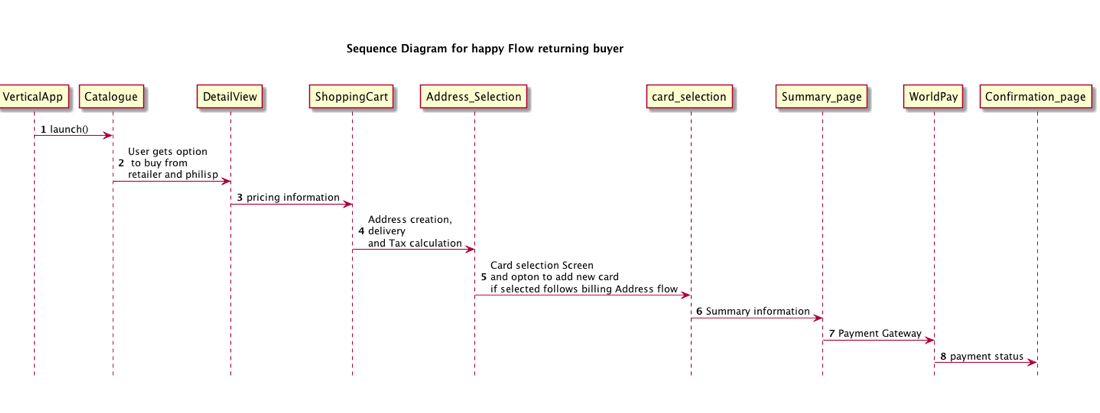
#### Voucher Code



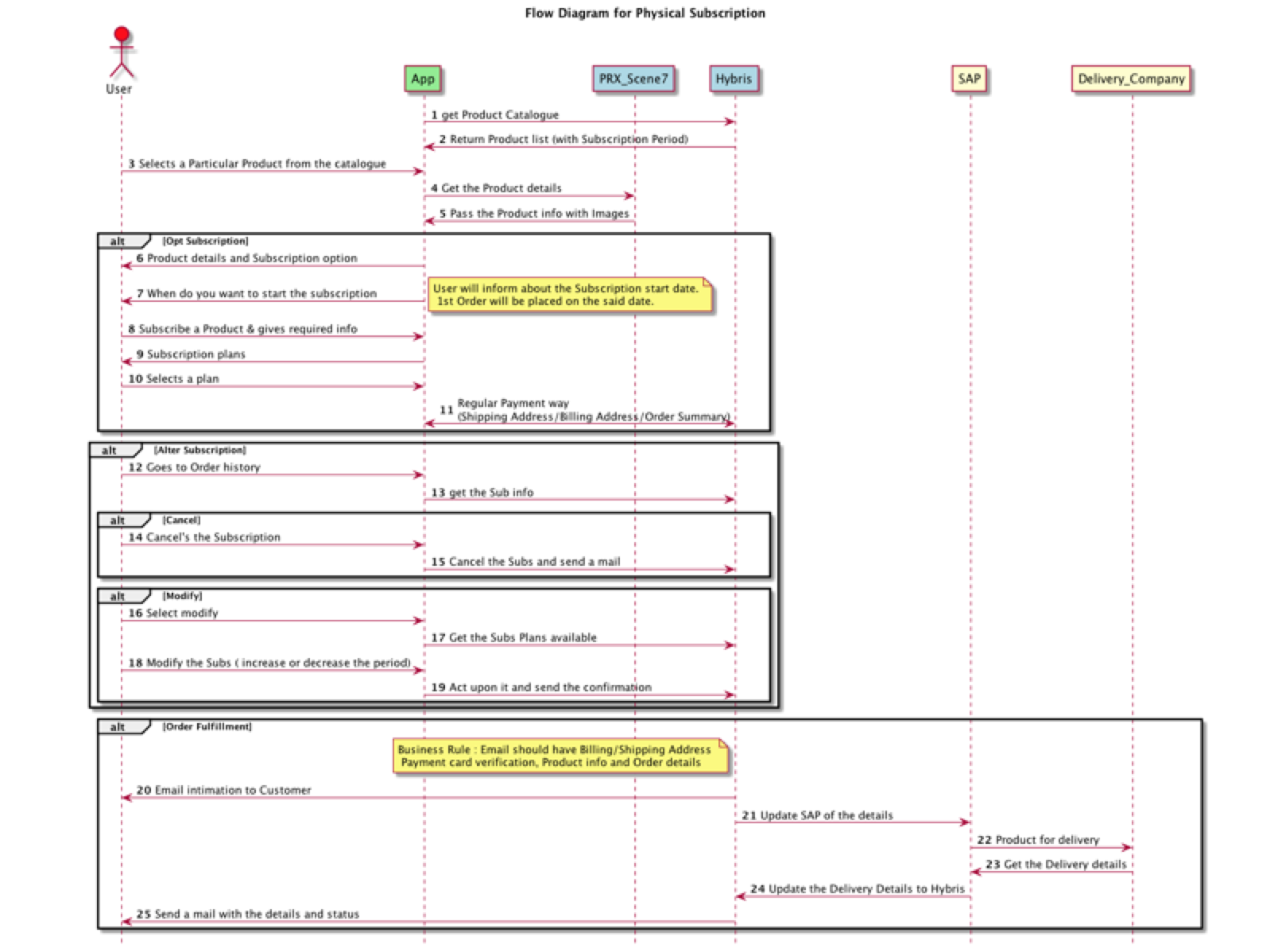
#### Purchase flow for first time buyer

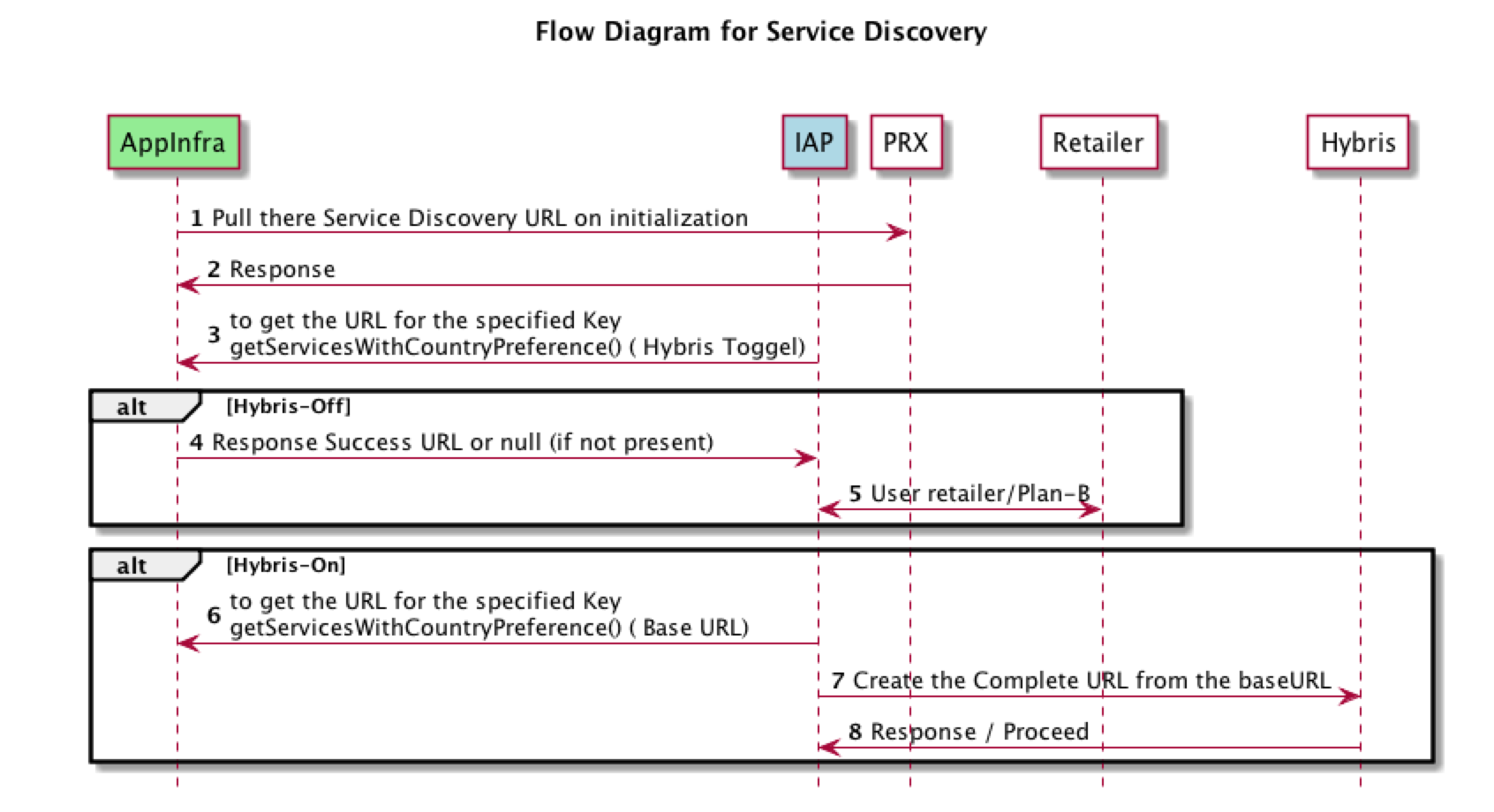


#### Purchase flow for returning buyer

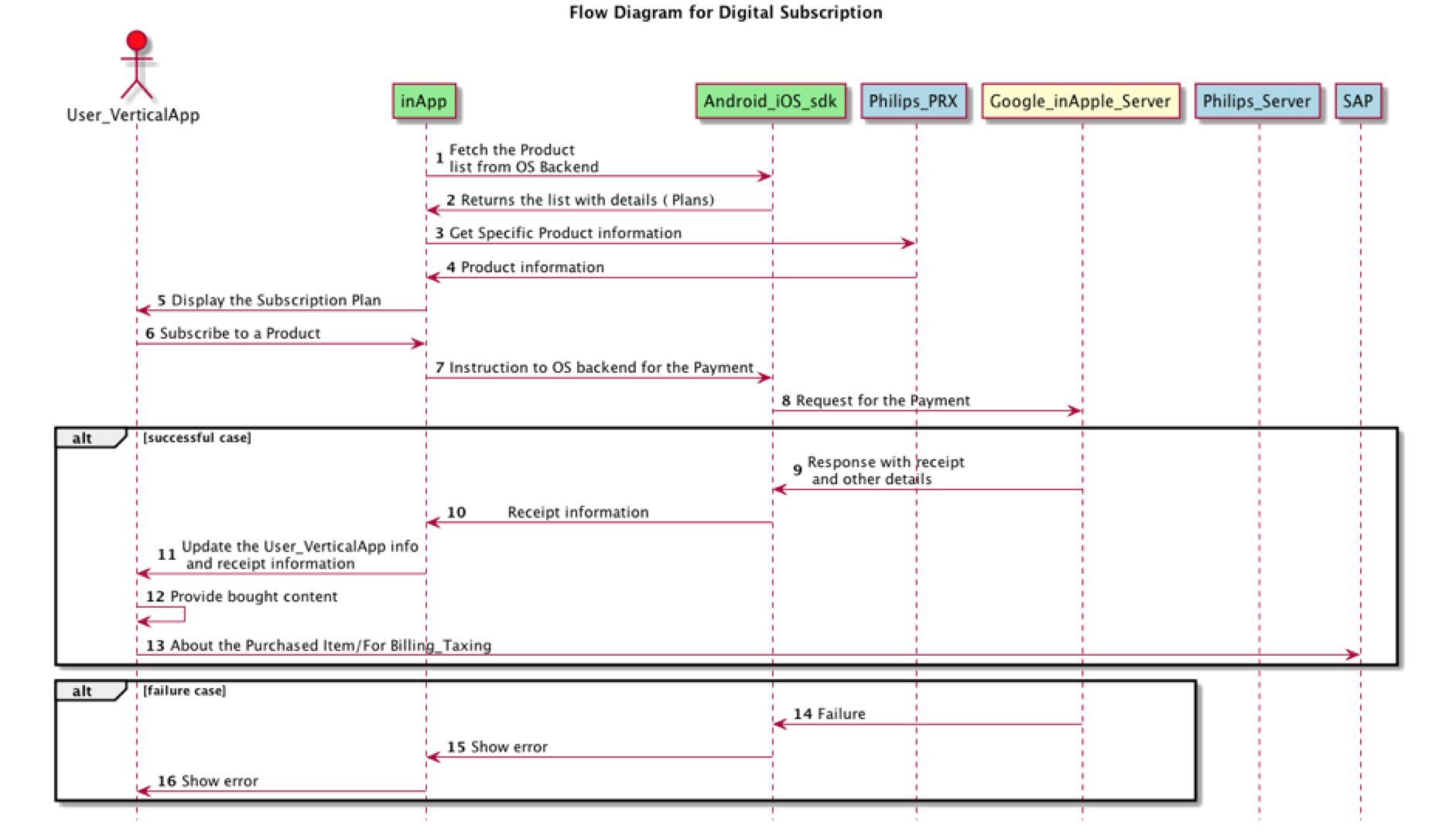


#### Physical Subscription





#### Digital Subscription



## External Interfaces

This section has been captures the external API exposed for the vertical/propositions to consume

### External interface -1

initIAP() interface is used for the initialization of the InApp purchase component, and must be done on the successful initialization of user registration.

### External interface -2

launchIAP() – is called to launch the InApp Purchase component. Based on the parameters passed, the the InApp purchase decides which screen needs to be launched. App infra is one of the dependency that’s being passed to pull different generic information like country and language.

The details of the API parameter and the integration steps are provided in detail in the intergration Document

## Internal Interfaces - NA

## Design Language Specification – (DLS)

With new DLS implementation the controls will be changed as per the specifications. The functionalities will be as it is, just the UI will align to the new design language.

# Revision History

| **Version** | **Date** | **Author** | **Description of Change** | **Reason for Change** |
| --- | --- | --- | --- | --- |
| 1.0 | 2017-APR-14 | Rahul | All section has been updated to align with new QMS doc. | Draft creation/ movement to new QMS doc |
|  |  |  |  |  |

# Approval

| **Name** | **Role / Function** | **Date** (YYYY-MON-DD) | **Signature** |
| --- | --- | --- | --- |
| Bhargavi | System Architect |  |  |
|  |  |  |  |