CDP

Project ID:

**eCommerce (InAppPurchase)**

**Interface specification**

**Author : Rahul**

**Version : 2.2**

**Date : 2016-05-14**

**Status : Submitted**

**Document data and history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document data** | | | |
| **Project name** | CDP – Horizontal Component InApp Purchase | | |
| **Document name** | InApp Purchase – Interface Design – API Interface Specification | | |
| **Document id** | Platform- iOS/Android | Date | 06-May-16 |
| **Version** | 2.1 | Status | Submitted |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document history** | | | |
| **Version** | **Date** | **Author** | **Reason** |
| 0.1 | 07-Jan-15 | Rahul | Initial draft covering the interface and detail design |
| 0.2 | 31-Jan-16 | Rahul | Integrated comments from BhaUpa |
| 1.0 | 02-Feb-16 | Rahul | API3 changed wrt to parameters passed. Integrated review input from Adriaan |
| 1.1 | 07-Feb-16 | Rahul | Integrated input from Raymond & Aravind |
| 1.4 | 03-Mar-16 | Rahul | Updated the API’s |
| 2.0 | 21-Apr-16 | Rahul | RE-defining APi’s to reduce the exposed level of API |
| 2.1 | 06-May-16 | Rahul | Error code for non-signed in user will be popped |
| 2.0 | 14-Jun-2016 | Rahul | Android API changes to support Fragment and Activity |

**Open issues:**

|  |  |
| --- | --- |
| **Date** | **Subject** |
| 02-Feb-16 | HybrisID to replace JanRainID, feasibility check to be done. As this would be helpful since it’s in plan to move to HybrisID. Needs to check Hans |
|  |  |

**DEFINITIONS & ABBREVATIONS**

|  |  |
| --- | --- |
| **Abbreviation** |  |
| eCom | eCommerce |
| CDP | Connected Digital Propositions |
| UI | User Interface |
| HTTP | Hyper Text transfer Protocol |
| API | Application Interface |
| IAP | InApp Purchase |

|  |  |
| --- | --- |
| **Author** | Rahul |
| **Approved by** |  |
| **Email Id** | [Rahul@philips.com](mailto:Rahul@philips.com) |

**Table of Contents**

**1** **INTRODUCTION** 5

**1.1** **Purpose & Scope** 5

**1.2** **Target Audience** 5

**1.3** **Reference** 5

**1.4** **Guiding Principles** 5

**2** **ARCHITECTURAL ANALYSIS** 5

**2.1** **Product Overview** 5

**2.2** **Future Extension** 5

**2.3** **Design method** 5

**2.4** **Platform Dependency & Interface Definition** 6

**2.4.1** **Class Names** 6

**2.4.2** **Errors** 6

**2.4.3** **User Interface** 6

**2.4.4** **InApp Interface – Vertical App** 6

**3** **DYNAMIC DESIGN** 9

**3.1** **Use Cases** 9

**3.1.1** **Complete buying cycle flow** 9

**3.1.2** **Use case: Initialization Module** 10

**3.1.3** **Use case: fetching shopping cart detail** 11

# **INTRODUCTION**

## **Purpose & Scope**

This doc basically contains the information to help the verticals to integrate the InApp Purchase common component to their verticals.

This contains the interface API’s, which defines what the API does along with when and how to call the same.

## **Target Audience**

Document is primarily intended for members of eCommerce team which consists of mobile app developers, testers.

This doc will be used by the vertical team to integrate the InApp Purchase library with their propositions.

## **Reference**

* Wireframe UX Flow - UX flow\_ purchasing\_v0.9
* System Architecture - HANAMS20151209-01V01 Software Architecture eCommerce
* CDP Common Components Brief In-App Purchase
* Hybris B2C Solution architecture

## **Guiding Principles**

Vertical propositions should get in touch with the InApp Purchase team if there is any need to extend or modify the component or interface to fulfil their requirements.

# **ARCHITECTURAL ANALYSIS**

## **Product Overview**

In App Purchase is one the main features of the mobile applications to be developed. As we see mass people are moving to the smartphone and mostly in the eCommerce section. Providing an ability to buy the products from the app would give the user an added advantage, boosting the sales.

As part of Connected Digital Proposition, InApp Purchase is a horizontal component that can be reused across various applications. The subsequent sections provide the detail of the InApp Purchase as a re-usable component.

## **Future Extension**

The architecture is scalable and can support future extension in the same line. But the actual is yet TBD.

## **Design method**

* Strict interaction: Layers can only interact with layers below. This will make sure that modifications in one layer will only affect layers above

## **Platform Dependency & Interface Definition**

While Object Oriented concepts are used in iOS and Android, the implementation details differ. The following sections provide a brief overview of such implementation details.

### **Class Names**

The class names used here can be modified based on the platform. In iOS, class names are prefixed with product names (Ex: User class is implemented as IAPUser. IAP Indicates In APP Purchase). In Android it is prefixed with package names like com.philips.cdp.iap.

### **Errors**

iOS uses NSError to indicate errors while Android uses exceptions, which are the standard defined by the respective platforms.

The design here provides an abstract of error and not the implementation details.

### **User Interface**

Storyboards are used to create views and segue is used for screen transition in iOS. Controllers are used for controlling the views.

Android uses XML editors to build views and activity to display and control them. Screen transitions are handled by intents.

### **InApp Interface – Vertical App**

#### **Responsibility/Functions**

Following functions are provided by this class.

1. Ability to fetch the cart details of the currently existing cart of the user
2. Means to initialize the complete component
3. Capacity to add the item to the cart

Configuration items to update

1. Base URL for hybris
2. propositionID

Configuration File:

JSON Configuration file needs to updated by the vertical/Demo app containing

propositionID: propositionID ( e.g – Tuscany2016 )

hostport: [https://www.acc.occ.shop.philips.com](https://www.acc.occ.shop.philips.com/)

Store URLs (hostport) – [https://www.acc.occ.shop.philips.com](https://www.acc.occ.shop.philips.com/) (need to change will as we move to production) please refer the attached JSON

#### **Interfaces & How to integrate/call the API’s**

How to integrate the API:

1. **Init()** – This API shall be called in context to the initialization of the InApp purchase. Practically this should be done once the registration is successful.
2. **getProductCartCount ()** - shall be called when the cart count has to be updated, this value is required to be shown on the cart details. onSuccess() will return the item number in the cart. onFailure(), else will return error with error type, in that scenario, the cart number shall not updated.
3. ***launchIAP***() -Whenever the user clicks on the shopping cart icon or buy now, the vertical shall create the class object and call this method. The parameter will decide whether to and, for example -

All the methods will be member function of the IAPHandler() class.

##### init()

|  |  |
| --- | --- |
| Function definition | ***init***(IAPSettings\_object) |
| Parameters | IAPSettings will have following properties:  *language* : String  *country* : String  *theme* : PUITheme  *(android*: index of uikit white background theme)  Note: *language* and *country* parameters cannot be nil and init() returns nil if so. *theme* argument is optional and by default takes default theme.  ***init***(settings : IAPSettings); // iOS    In Android the context and IAPSetting need to pass in argument.  ***init***(Context\_object, IAPSettings\_object) |
| Pre-Condition | User must be registered. |
| Brief | This method basically initializes the store backend urls. |
| Error scenarios | void |
| Callback functions | * None |
| Type | Sync call |

##### getProductCartCount ()

|  |  |
| --- | --- |
| Function definition | ***getProductCartCount***() |
| Parameters | In Android the context & callback shall also be sent |
| Pre-Condition | Registration module would had successfully executed so that the required parameter by IAP module are present. |
| Brief | This method basically fetches the card number details |
| Error scenarios | * Invalid parameter * Network issues * Not registered ( oAuth not generated) |
| Callback functions | * onSuccess – returns integer count within success block * onError – Error |
| Type | Asynchronous |

##### launchIAP()

|  |  |
| --- | --- |
| Function definition | ***launchIAP*** (*Controller/Context,* *landingView* IAPLandingViews, String *productCTN, Error errorcode*) |
| Parameters | *navigationController:* This must be UInavigationController type  *Context :* wrt to the android  *landingView:* Enum/int value indicates where the app should land as soon as InApp component launches. Values can be:  IAPProductCatalogueView: launches product catalogue view **0**  IAPShoppingCartView: launches shopping cart view 1  productCTN (optional) - if present, add the CTN and launch Shopping cart.  If *landingView* is 0, CTN will be ignored  ***launchIAP*** (*navigationController*: UINavinagationController, *landingView* : IAPLandingViews, *productCTN*: String = “”,failureHandler:(NSError)->())  In Android, the landingView, productCTN,callback shall also be sent  ***launchIAP*** (*landingView* IAPLandingViews, String *productCTN, Error errorcode*) |
| Pre-Condition | Registration module would had successfully executed so that the required parameter are present. |
| Brief | This method helps the vertical team to land on the specific IAP screen with required information. |
| Error scenarios | If the user is not signed it, it will throw the error with error code 1004. Verticals to handle it ( error code – 1004) |
| Callback functions | - |
| Type | Asynchronous |

In iOS the navigation view controller shall be sent

\*\*\*iOS Specific:

##### IAPLandingViews

IAPLandingViews enum/int (iOS/Android) is used to represent different landingViews lanuchIAP() supports. Contains following options:

IAPProductCatalogueView - 0

IAPShoppingCartView - 1

(In Android:-> IAPLandingViews.IAPLandingViews.IAP\_PRODUCT\_CATALOG\_VIEW

IAPLandingViews.IAPLandingViews.IAP\_SHOPPING\_CART\_VIEW)

##### IAPhandlerProtocol

IAPHandlerProtocol will be used to indicate InApp component is closed. Vertical apps must use this protocol and implement *didCloseInAppPurchaseController*() delegate method to dismiss the InApp controller.

Note: this will be extended when the scope of viewing/tracking history comes in scope.

#### **Interfaces between InAppPurchase and OCC module**

Please find the embedded doc specifying the RESTful API’s.



Refer SharePoint for the latest doc @

<https://share-intra.philips.com/sites/STS20151202191512/_layouts/15/start.aspx#/Shared%20Documents/Forms/AllItems.aspx>

# **DYNAMIC DESIGN**

## **Use Cases**

### **Complete buying cycle flow**



### **Use case: Initialization Module**

This use case occurs during application init, post user registration. The following sequence of operations is performed during the process. If the user registration is not successful, initIAP must not be called.

1. Application creates User object
2. JanRain Registration is carried out, which generates JanRainID
3. The generated username and the JanRainID is passed to the InAppPurchase init module

### **Use case: fetching shopping cart detail**

Once the initIAP is successful, the vertical app shall request for the current cart details to show the same in the number on the shopping cart icon. Should not be called if the initIAP is not successful.

1. initIAP is successful, so that the oAuth access token is already fetched
2. request for the user carat detail based on the cart number is fetch
3. should return value