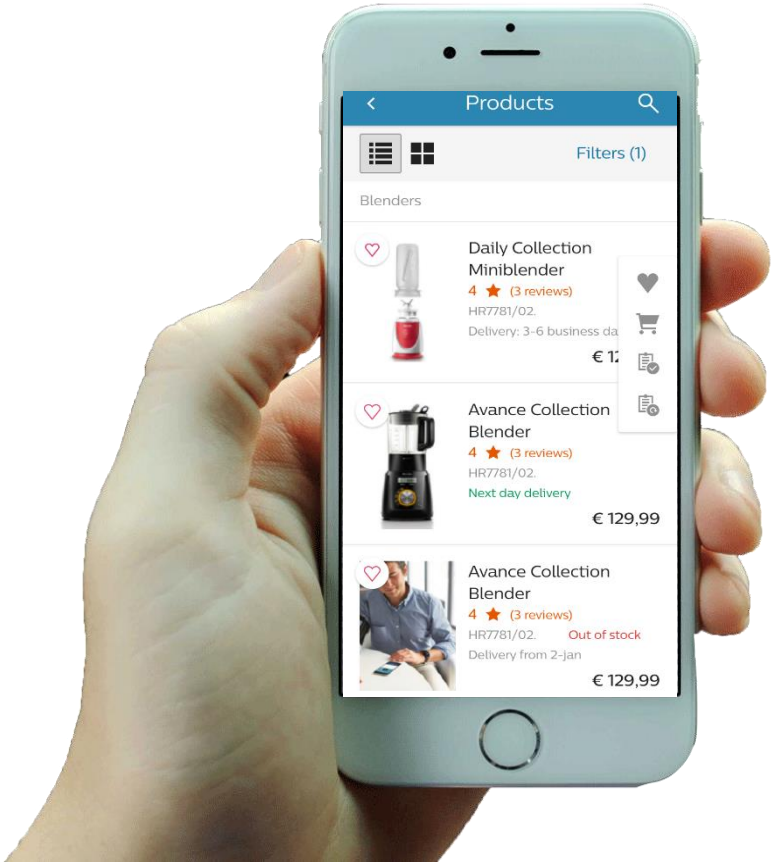
PHILIPS



In-App Purchase

IOS INTEGRATION DOCUMENT

ENTERPRISE MOBILE SOLUTION (EMS)

PHILIPS

Document Configuration Management

Document Identification

Name of Project	In-App purchase
Name of Report	In-App purchase iOS Reference Guide
File Name	IAP_iOS_Integration_Document_V1.0

Document Change Control

Version	Date Released	Change Notice
1.0	2-Dec-2019	Finger Lime – Release Integration Document

PHILIPS

Contents

PHILIPS

INTEGRATION

The easiest and preferred way to use these components is using Cocoapods (version 1.5.3 or newer). In your Podfile, you have to add a source line for the Philips internal pod spec repository. So a minimal Podfile would look like this:

```
source
'https://PhilipsAgile:d3khntuxv5dta2r7pl6gahfpvidja5gtqp7civmm2dzwunwl5p
pa@dev.azure.com/PhilipsAgile/8%2E0%20DC%20Innovations%20%28IET%29/_git/
mobile-ios-podspecs-release'

pod 'InAppPurchase', '#Pod_Version'

ex: #Pod Version: 1906.0.1575037646
```

INITIALIZATION

The InApp Purchase component can be initialized using the following method:

```
var iapInterfaceHandler:IAPInterface!
var iapInputSettings:IAPLaunchInput?

iapInputSettings = IAPLaunchInput()
iapInputSettings?.cartIconDelegate = self

let iapAppDependencies = IAPDependencies()
iapAppDependencies.appInfra = appInfraHandler

let appSettings = IAPSettings()
iapInterfaceHandler = IAPInterface(dependencies: iapAppDependencies,
andSettings: appSettings)
```

Note: Please refer IAPInterface section for more reference

Basically, vertical app needs to declare like the above. This has three main parts like IAPSettings, IAPLaunchInput and IAPDependencies. The object of first and third type should be passed while initializing IAPInterface which basically initializes the configuration.

IAPSettings

IAP don't have any settings to be initialized. So only default initialization of IAPSettings is required to be passed while creating IAPInterface object.

IAPLaunchInput

IAPLaunchInput class is responsible for initializing the settings required for launching. It has a public init() method which accepts PhilipsUIKitDLS theme. By default, it takes the default theme.

```
public init?(theme: UIDTheme =
UIDThemeManager.sharedInstance.defaultTheme)

//Also it has a delegate of IAPCartIconProtocol which is used to give
the control back to vertical to handle two purposes like showing cart
icon and cart count. For example:

var iapInputSettings:IAPLaunchInput?

iapInputSettings = IAPLaunchInput()
iapInputSettings?.cartIconDelegate = self
```

IAPDependencies

This class handles the dependency required for IAP. So IAP has two dependency i.e AppInfra and UserDataInterface. So vertical needs to initialize IAPDependencies and set the app infra and user data interface object. This app infra object will be responsible for logging, tagging and some configuration and user data interface object is responsible for all user related data.

```
let iapAppDependencies = IAPDependencies()
iapAppDependencies.appInfra = appInfraHandler

let urInterface = URInterface(dependencies:
userRegistrationDependencies, andSettings: nil)
       self.userDataInterface = urInterface.userDataInterface()
```

IAPCartIconProtocol

Vertical Apps needs to adopt this protocol to handle the visibility of the cart icon and updating the cart count. This protocol has two following methods.

```
func didUpdateCartCount()
func updateCartIconVisibility(_ shouldShow: Bool)
```

LAUNCHING

For launching the IAP component vertical has to call the following method of IAPLaunchInput. Then IAPFlowInput object needs to be initialized with default constructor or with one ctn or ctn list. Based on the landing view vertical would initialize the IAPFlowInput. Last parameter “ignoredRetailersList” is optional. Proposition can pass empty array or list of items if it wants to ignore any of the third party retailer.

```
open func setIAPFlow(_ inIAPFlow:IAPFlow, withSettings:IAPFlowInput,
ignoredRetailersList:[String] = [])
```

IAP can be launched with 5 Landing Views, namely:

- **iapProductCatalogueView**
  - Launches IAP Catalogue Screen displaying all the Product List downloaded from Hybris according to the given Configuration like Locale and Proposition ID.

- **iapPurchaseHistoryView**
  - Launches IAP Order History Screen displaying all the Order Histories with their details for the logged in User.

- **iapShoppingCartView**
  - Launches IAP Shopping Cart Screen displaying all the Products and Price Details for the Cart of the Logged in User.

- **iapProductDetailView**
  - Launches IAP Product Detail Screen with the CTN passed to IAP and displays all the details of the Product with the Add to Cart and show Retailer List options.

- **iapCategorizedCatalogueView**
  - Launches IAP Catalogue Screen displaying all the Product List downloaded from Hybris according to the given Configuration like Locale and Proposition ID. The difference between this Landing View and iapProductCatalogueView is that while launching IAP with this Landing View, we have to pass list of CTNs and only those Products will be displayed in the screen.

Example:

```
let iapSeetingsInputHelper = IAPFlowInput(inCTNList: self.ctnArray)
self.iapInputSettings?.setIAPFlow(.IAPProductCatalogueView,
withSettings: iapSeetingsInputHelper)

//for getting the respective view controller
let productCatalogueVC =
self.iapHandler.instantiateViewController(self.iapInputSettings!) {
(inError) in
  //error handling
}
self.navigationController?.pushViewController (productCatalogueVC!,
animated: true)

There is an enum of IAPFlow which represents the landing view.
@objc public enum IAPFlow: Int {
        case IAPProductCatalogueView
        case IAPShoppingCartView
        case IAPPurchaseHistoryView
        case IAPProductDetailView
        case IAPBuyDirectView//NA as this is retailer flow only
        case IAPCategorizedCatalogueView
    }
```

IAPInterface

IAPInterface is the interface class for interacting with the InApp Purchase component. It has two methods for initializing and launching. Following are the two methods:

```
/*For initializing IAP*/
public init(dependencies: UAPPDependencies, andSettings settings:
UAPPSettings?)

/*For launching IAP*/
public func instantiateViewController(_ launchInput: UAPPLaunchInput,
withErrorHandler completionHandler: ((Error?) -> Void)? = nil) ->
UIViewController?
```

Note: please refer initialization and launching section for more reference.

IAPInterface has exposed few public interface methods for integrating InApp Purchase component.

Fetch cart count:

```
func getProductCartCount(_ success:@escaping (Int)-
>(),failureHandler:@escaping (NSError)->())
```

Fetch complete product list from Hybris:

```
func fetchCompleteProductList(_ completion:@escaping (_
withProducts:Array<String>)->(), failureHandler: @escaping (NSError)-
>())
```

Enabling shopping cart or not:

```
func isCartVisible(_ success:@escaping (Bool)->(),
failureHandler:@escaping (NSError)->())
```

Note: “isCartVisible” API is a mandatory method after init. This method handles the service discovery calls and based on that cart icon visibility will be decided. So if the user changes the country then this method needs to be called again to decide the visibility of the cart icon. So after initialization of InApp Purchase, and before calling any other methods, proposition has to call “isCartVisible” method at least once, and post country change.

PHILIPS

Navigation

Using the parent controller's navigation controller, it will push the InApp Purchase on the same navigation stack. So the vertical app should have a navigation controller.

Configuration

The proposition has to keep the AppConfig.json file.

AppConfig.json

There is a group named “IAP” for InApp Purchase. There is one key named “propositionId” which basically represents the vertical proposition name. For example, Tuscany proposition has id “Tuscany2016”.

```
{
 "UserRegistration": {
        "JanRainConfiguration.RegistrationClientID.Development":
"8kaxdrpvkwyr7pnp987amu4aqb4wmnte",
        "JanRainConfiguration.RegistrationClientID.Testing":
"g52bfma28yjbd24hyjcswudwedcmqy7c",
        "JanRainConfiguration.RegistrationClientID.Evaluation":
"f2stykcygm7enbwfw2u9fbg6h6syb8yd",
        "JanRainConfiguration.RegistrationClientID.Staging":
"f2stykcygm7enbwfw2u9fbg6h6syb8yd",
        "JanRainConfiguration.RegistrationClientID.Production":
"9z23k3q8bhqyfwx78aru6bz8zksga54u",
        "PILConfiguration.CampaignID": "CL20150501_PC_TB_COPPA",
        "Flow.EmailVerificationRequired" : true,
        "Flow.TermsAndConditionsAcceptanceRequired" : true,
        "Flow.MinimumAgeLimit" : { "NL":12 ,"GB":14,"default": 16},
        "SigninProviders.default": ["myphilips","facebook"]
        "SigninProviders.CN": ["myphilips","sinaweibo"]
    },
    "IAP": {
        "propositionId" : "Tuscany2016"  // Proposition Specific ID
needs to be mentioned
    },
    "appinfra": {
        "appidentity.micrositeId": 77000,
        "appidentity.sector": "b2c",
        "appidentity.appState": "Staging",
        "appidentity.serviceDiscoveryEnvironment": "Production",
        "servicediscovery.platformMicrositeId": 77000,
        "servicediscovery.platformEnvironment":"Production"
    }
}
```

PHILIPS

TAGGING

Tagging is completely handled by InAppPurchase with AppInfra tagging API's, please refer the AppInfra integration document for any clarification.

SERVICE DISCOVERY

InAppPurchase has two flows i.e Hybris and Non-Hybris (i.e only retailer flow). InAppPurchase has implemented with service discovery feature to decide Hybris or retailer flow. Based on app infra service discovery implementation, InApp Purchase flow will be decided.

---------- End of Document----------