Table of Contents

[1 DOCUMENT INTRODUCTION 3](#_Toc500425131)

[1.1 Purpose 3](#_Toc500425132)

[1.2 Scope 3](#_Toc500425133)

[1.3 References 3](#_Toc500425134)

[1.4 Terminology & Abbreviations 3](#_Toc500425135)

[2 Overview 4](#_Toc500425136)

[3 Integration 4](#_Toc500425137)

[3.1 Installation 4](#_Toc500425138)

[Pre-Requisite 4](#_Toc500425139)

[Maven Repository 4](#_Toc500425140)

[3.2 Library Integration 4](#_Toc500425141)

[3.3 Configuration 4](#_Toc500425156)

[4 Usage 5](#_Toc500425157)

[4.1 Launching my account uApp 5](#_Toc500425158)

[Listener and Callbacks 6](#_Toc500425159)

[4.2 6](#_Toc500425160)

[5 Revision History 7](#_Toc500425161)

[6 Approval 7](#_Toc500425162)

# DOCUMENT INTRODUCTION

## Purpose

This document describes the Integration document for My Account.

## Scope

The scope of this document is limited to the My Account micro app, which is to be used by application. This document does not describe the application as a whole, nor does it describe in detail any cloud services that may be related to the functionalities covered in the My Account component.

This document include an extensive API description;

## References

| **Reference** | **Identification** | **Title / additional remarks** |
| --- | --- | --- |
| [REQUIREMENT] | MYA000001 | Software Requirement Specification |
| [TECHNICAL DESIGN] | MYA000002 | Technical Design MYA |
| [INTEGRATION DOC] | MYA000009 | Android Integration guidelines |
| [PLATFORM TECH DESIGN] | PLL000002 | Platform technical design |
| [PLATFORM SRS] | PLL000001 | Platform Software Requirement Specification |

## Terminology & Abbreviations

| **Terminology & Abbreviations** | **Description/Definition** |
| --- | --- |
| CDPP | Connected Digital propositions and platform |
| app | Mobile application |
| App Infra | Mobile application infrastructure library |
| CoCo | Common Component |
| OS | Operating system |
| MYA | My Account |
| uApp | Micro app |

# Overview

My Account is a micro app which provides centralized place for user profile management and settings.

Currently user has no centralized space to manage all related information in one place, features are spread across multiple CoCo’s :Product registration, User registration, Consumer care and Data services. This creates a centralized place for user related settings, consents, orders, subscriptions etc.

# Integration

## Installation

My account can be integrated using artifactory

## Pre-Requisite

* Android Studio 1.5 or higher
* A device running Android version 5.0 or newer

# Maven Repository

allprojects {  
 repositories {  
 maven { url **'http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-jcenter-remote'** credentials {  
 username = **"readonly"** password = **"123qweasdzxc"** }  
 }  
 }  
}

## Library Integration

My accounts can be integrated by adding the library to the build.gradle file as compile(**‘com.philips.cdp:MyAccount’)**

Source code can be found [here](http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/CDP2 Portfolio/_git/mya-android-my-account).

## Configuration

Profile and Settings menu in the my account is configurable. These items can be configured in the AppConfig.json file as follows. This configuration is optional and in the absence of this configuration My account will display the default items i.e. My details under profile tab and country and privacy settings under settings tab. Propositions can add extra items to this list

    "mya": {

        "profile.menuItems":["MYA\_My\_details"

                             ],

        "settings.menuItems":["MYA\_Country",

                              "MYA\_Privacy\_Settings",

                             ]

    }

Group name for My Account is “mya”. Under this group we can specify the items in profile.menuItems and settings.menuItems as array of strings. These strings will be the localized key for the menu item title appears in the My account pages. The same keys will be passed back to proposition when user tap on this menu item via callback listener.

If application is using consent via my account then the following configuration is required

 "hsdp": {

        "appName": "OneBackend",

        "propositionName": "OneBackendProp"

    }

# Usage

## Launching my account uApp

My account confirms to standard interface of micro app so it can be launch as any other micro app as follows

 MyaDependencies uappDependencies = new MyaDependencies(MyAccountDemoUAppInterface.getAppInfra());

// Pass app-infra as dependency

MyaLaunchInput launchInput = new MyaLaunchInput(context, getMyaListener());

launchInput.setConsentDefinitions(createConsentDefinitions(Locale.US)); // pass list of consent definitions required to launch consent screen

MyaInterface myaInterface = new MyaInterface();

myaInterface.init(uappDependencies, new MyaSettings(context));

if (to be launched as Activity) {

ActivityLauncher activityLauncher = new ActivityLauncher(ActivityLauncher.ActivityOrientation.SCREEN\_ORIENTATION\_SENSOR,

dlsThemeConfig,

dlsThemeResId, bundle);

myaInterface.launch(activityLauncher, launchInput);

} else {

myaInterface.launch(new FragmentLauncher(context, fragment\_container\_id, new ActionBarListener() {

@Override

public void updateActionBar(@StringRes int i, boolean b) {

}

@Override

public void updateActionBar(String s, boolean b) {

}

}), launchInput);

}

@NonNull

private MyaListener getMyaListener() {

return new MyaListener() {

@Override

public boolean onClickMyaItem(String itemName) {

return false;

// For specific configurable item added in AppConfig.json, value of parameter itemName is the key provided in AppConfig.json, if proposition is handling it need to return true and handle it as required, if not handled return false.

}

@Override

public DataInterface getDataInterface(DataModelType modelType) {

return LaunchFragment.this.getDataInterface(modelType);

// Based on the model type returned by Myaccounts preposition should pass the corresponding data model.

}

@Override

public void onError(MyaError myaError) {

// onError callback will be called on occurance of any error for example, on user not logged in.

if (myaError == MyaError.USER\_NOT\_SIGNED\_IN) {

}

}

};

}

* Initialize MYADependencies
* Set AppInfra instance of the application to dependency
* Initialize launch input with dependency and settings
* Create MyaInterface instance, invoke init and launch with dependent parameters.
* in case of error correcsponding error callback will called. User should be loggedin to launch My account or it will result in error

## Listener and Callbacks

My account designed to be configurable and as independent as possible from other cocos even though it can be plugged easily

My account listener are declared in MyaListener.class

   boolean onClickMyaItem(String itemName);

   boolean onLogOut();

void onError(MyaError myaError);

* onClickMyaItem listener will be called when the user tap on item in my account. A string is passed which will be the same string that is configured in the AppConfig.json file . If this item is application specific they should handle and return true else they should return false . My account can handle items like MYA\_My\_Details, MYA\_Country.
* My account can handle “Logout” item also . This will be called when the user tap on logout in My account. If my account uapp is pushed/ presented to existing screen logout will return to that screen any other user journey should be handled by the caller.
* In case of error corresponding error callback will called. User should be logged in to launch My account or it will result in error.

## Just in time consent

Just in time, is a widget within My account, which provides an application’s developer, functionality to set a specific consent at the point where it is needed.

To use this widget an application developer must configure JustInTimeConsentDependencies:

public class JustInTimeConsentDependencies {  
  
 public static ConsentHandlerInterface *consentHandlerInterface*;  
 public static ConsentDefinition *consentDefinition*;  
 public static JustInTimeTextResources *textResources*;  
 public static AppInfraInterface *appInfra*;  
 public static JustInTimeWidgetHandler *completionListener*;  
}

1. Consent handler interface – Interface to read and set consent states.
2. Consent definition – The consent which is going to be asked to the user.
3. Just in time text resources – The texts to be displayed on the widget.
4. AppInfraInterface – App Infra provides a range of modules that are the basis for any mobile application
5. JustInTimeWidgetHandler - This provides the necessary callbacks to perform actions based on user input. The interface specifies only two methods. One for when the user gives consent, and one for when the user refuses to give consent. Note that the user actions update the consent values in the backend, thus the application developer does not need to maintain them.

# Revision History

| **Version** | **Date** | **Author** | **Description of Change** | **Reason for Change** |
| --- | --- | --- | --- | --- |
| 0.1 | 2017-12-07 | Yogesh HS | Draft version of MYA Integration doc |  |
| 0.2 | 2018-02-12 | Rebelo, Antonio | Extended document to describe the usage of just in time widget | Functionality added |
| 0.3 | 2018-03-06 | Kok, Erwin | Fixed typos, saved as word document | Wrong file format |

# Approval

| **Name** | **Role / Function** | **Date** (YYYY-MON-DD) | **Signature** |
| --- | --- | --- | --- |
| Deepthi Shivakumar | Architect |  |  |
|  |  |  |  |