|  |
| --- |
| Philips |
| PRX CLIENT LIBRARY |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version History** | | | | |
| **Version** | **Date** | **Author** | **Section** | **Changes** |
| 0.1 | 05 Oct 2015 | Deepthi Shivakumar | All | Initial Draft version |
| 0.2 | 12 Oct 2015 | Deepthi Shivakumar | All | Added sub classes overview as per Matthijs suggestion |
| 0.3 | 13 Oct  2015 | Deepthi Shivakumar | All | Created standard template. |
|  |  |  |  |  |

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

CONTENTS

1. DEFINITIONS & ABBREVATIONS 3

2. INTRODUCTION 3

2.1 Purpose 3

2.2 Scope 3

2.3 Target Audience 3

2.4 References 3

3. ARCHITECTURAL ANALYSIS 4

3.1 Overview 4

3.2 System Context 4

3.2.1 Client to PRX library communication 4

3.2.2 PRX to Volley/AF Networking library communication 4

4. DESIGN 4

4.1 Design Overview 4

4.1.1 Client 5

4.1.2 RequestManager 5

4.1.3 Responsehandler 5

4.1.4 Product/ProductSummary/ProductAssets 5

4.1.5 PrxDataBuilder 6

4.1.6 NetworkWrapper 6

4.1.7 ResponseData 6

4.1.8 ResponseListener 6

4.2 Decomposition / Sub classes 6

5. Notes 7

# DEFINITIONS & ABBREVATIONS

|  |  |
| --- | --- |
| CDP | Connected Digital Propositions |
| CC | Consumer Care |

# INTRODUCTION

This document provides an architectural and top level design for PRX client library developed on Android and iOS. The various building blocks of the library are depicted in the document.

## Purpose

This document serves as the base document for designing the individual component and implementation.

## Scope

This document covers the design aspects of PRX client library which can be integrated across all apps.

## Target Audience

*PIC – CDP development team*

## References

|  |  |
| --- | --- |
| PRX API doc | PRX\_API\_v4.13.doc |

# ARCHITECTURAL ANALYSIS

## Overview

The main functionality of this library is to download any data related to product present on PRX. It can be used by consumer care, registration and different applications. This library can be reused by other projects with minimal development changes as a generic network component as well.

## System Context

The following diagram shows the Eco-System of PRX client library.

Registration

Consumer Care

Application

PRX client Library

Figure 3.2: PRX client Eco-System

### Client to PRX library communication

Client can be either an application or consumer care component or registration component.

PRX client library exposes classes and APIs to clients to send a request and get a response. Library also helps clients to customise the requests.

### PRX to Volley/AF Networking library communication

Library classes will communicate with volley library in case of android and AF Networking in case of IOS for all network calls.

It also receives response and is responsible for processing it and delegating further to clients.

# DESIGN

## Design Overview

The figure 4.1 below shows the top level structure of PRX client library.

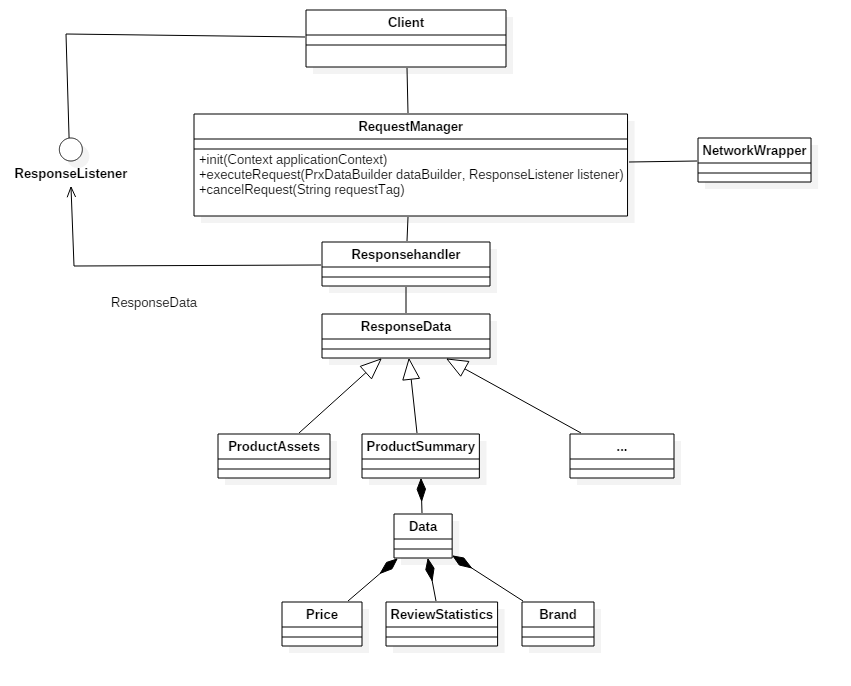
**

Figure 4.1: High level software decomposition

### Client

It can be an application, consumer care component or registration component.

### RequestManager

It provides set of public APIs for placing requests from client and also talks to Network wrapper class for performing network operations.

### Responsehandler

Handles response like invoking respective builders to build the response. It also invokes listener/blocks.

### Product/ProductSummary/ProductAssets

Model data for each request type.

### PrxDataBuilder

It is an abstract class which acts as base class for all data builders. For each type of request, we create builder class which is responsible for forming the right URL and also it is responsible for creating response object by routing to corresponding model class. It has abstract methods called getRequestUrl () and getResponse ().

### NetworkWrapper

A class which performs HTTP get, maintains request queue, handles caching etc.

It is responsible for interacting with any third party libraries that we use for performing network operations.

In android, we will be using Volley.

In IOS, we will be using AF networking.

### ResponseData

An abstract base class for all response model classes. It is a parent class which is used while returning back the response to clients after which clients will typecast to respective model class. All model classes also implements abstract method called parseResponse for parsing the response.

**Example:** To get product summary, we build ProductSummaryBuilder where in it overrides two abstract methods. Constructor of each builder will take necessary parameters required to build the right URL, for example ctn number. When client executes request, it creates a builder object and passes it to RequestManager along with listener object. Client also needs to call setLocale on this object.

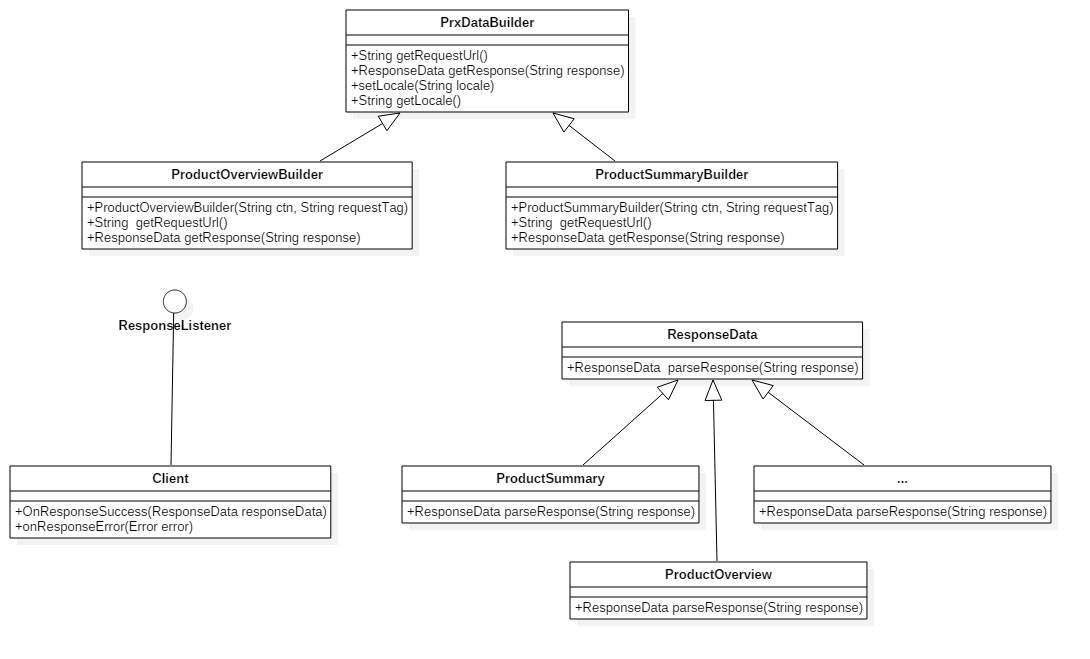
### ResponseListener

It is an interface which has two methods. On successful response we return onResponseSuccess (ResponseData data) and on any error we return onResponseError (Error error).

* **ResponseData** – Basically it returns exact model class like ProductSummary which is sub class of ResponseData. Clients need to create listener object for each request and hence when they get response they will typecast to respective model class data.
* **Error** – It is an enum which has error codes with hardcoded string messages which helps clients displaying appropriate log messages. It can be either an exception name with message details or a predefined error code with hardcoded string message.

## Decomposition / Sub classes

The layered design is decomposed into following major paradigms.



# Notes

1. For now it is thought to be a single queue which holds requests from all clients be it cc, registration or an app.
2. If we need to cancel all requests, it is with respect to one queue and hence it may not be valid to cancel all requests from 3 clients if we get request from one of them. Still need to analyze bit on request tagging and cancelling all/multiple requests on single/multiple queue.
3. In Volley library, it has defined its own interface which gets invoked once we get response from server. Since it’s an implementation detail for external clients, we need to implement volley listener inside prx component and then invoke client side interface callbacks after parsing the response.

We may need another similar interface implementation from IOS side.

1. Volley provides disk based cache by default with request queue. We can also create new request queue by customizing cache by specifying size, type (disk or LRU based) etc.