|  |  |  |
| --- | --- | --- |
| June 12  2015 | IndusInd Bank Mobile Application | |
|  | |  |

**Technical Design Document**

**Version 1.0**

**© Copyright. Kony Solutions, 2011. All rights reserved.**

*This manual contains proprietary information, which is protected by copyright. The information in this manual is furnished under a software license or non-disclosure agreement. The information may be used or copied only in accordance with the terms of the applicable agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Kony Solutions.*

**Document Revision History**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description | Version | Revised by |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Process / Document Owner** |  | | |
| Document ID |  |  |  |
|  | Name | Date | Signature |
| Approved by |  |  |  |
| Reviewed by |  |  |  |
| Prepared by |  |  |  |

**Reference Materials**

|  |  |  |  |
| --- | --- | --- | --- |
| Document | Version | Document Use | SharePoint Link |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[To update the table of contents, select the message below and press F9]

[1. Introduction 7](#_Toc426492127)

[1.1 Purpose 7](#_Toc426492128)

[1.2 Project Overview 7](#_Toc426492129)

[1.3 Scope 7](#_Toc426492130)

[1.3.1 Platform Scope 7](#_Toc426492131)

[1.3.2 Integration Scope 7](#_Toc426492132)

[1.3.3 High Level Functional Scope 8](#_Toc426492133)

[1.4 Assumptions 8](#_Toc426492134)

[2. High Level System Overview 8](#_Toc426492135)

[2.1 Kony Development Model 8](#_Toc426492136)

[2.2 KonyOne Server Architecture 9](#_Toc426492137)

[2.3 Mobile Application Working Flow 11](#_Toc426492138)

[2.4 Services Flow 11](#_Toc426492139)

[3. Application Architecture & Design 12](#_Toc426492140)

[3.1 Client 12](#_Toc426492141)

[3.1.1 Forms 13](#_Toc426492142)

[3.1.2 Modules 13](#_Toc426492143)

[3.1.3 Kony API’s 13](#_Toc426492144)

[3.1.4 Skins/Themes 14](#_Toc426492145)

[3.2 Server 14](#_Toc426492146)

[3.2.1 Connectors 14](#_Toc426492147)

[3.2.2 Service Definition 15](#_Toc426492148)

[3.3 Application life cycle Events 15](#_Toc426492149)

[3.3.1 Pre-App Init 15](#_Toc426492150)

[3.3.2 Post-App Init 16](#_Toc426492151)

[3.3.3 Deeplink Function 16](#_Toc426492152)

[4. Functional Modules 16](#_Toc426492153)

[4.1 Registration 16](#_Toc426492154)

[4.1.1 Forms, Modules & Actions 16](#_Toc426492155)

[4.1.2 Form Navigations 18](#_Toc426492156)

[4.1.3 Diagrams 18](#_Toc426492157)

[4.2 Login 21](#_Toc426492158)

[4.2.1 Forms, Modules & Actions 21](#_Toc426492159)

[4.2.2 Form Navigations 22](#_Toc426492160)

[4.2.3 Diagrams 22](#_Toc426492161)

[4.3 Forgot Password 25](#_Toc426492162)

[4.3.1 Forms, Modules & Actions 25](#_Toc426492163)

[4.3.2 Form Navigations 26](#_Toc426492164)

[4.3.3 Diagrams 26](#_Toc426492165)

[4.4 Account Overview 27](#_Toc426492166)

[4.4.1 Forms, Modules & Actions 27](#_Toc426492167)

[4.4.2 Form Navigations 28](#_Toc426492168)

[4.4.3 Diagrams 28](#_Toc426492169)

[4.5 Payment & Transfer 29](#_Toc426492170)

[4.5.1 Forms, Modules & Actions 29](#_Toc426492171)

[4.5.2 Form Navigations 30](#_Toc426492172)

[4.5.3 Diagrams 31](#_Toc426492173)

[4.6 Mobile/DTH Recharge 31](#_Toc426492174)

[4.6.1 Forms, Modules & Actions 31](#_Toc426492175)

[4.6.2 Form Navigations 32](#_Toc426492176)

[4.6.3 Diagrams 32](#_Toc426492177)

[5. Caching 32](#_Toc426492178)

[5.1 Client Side Caching 32](#_Toc426492179)

[5.1.1 Application Launch 33](#_Toc426492180)

[5.1.2 User Details 33](#_Toc426492181)

[5.1.3 Account Details 33](#_Toc426492182)

[5.2 Server Side Caching 34](#_Toc426492183)

[5.3 Persisting data in Device DB 34](#_Toc426492184)

[5.3.1 Last login User 35](#_Toc426492185)

[5.3.2 Profile Picture 35](#_Toc426492186)

[5.3.3 Users using the Device 35](#_Toc426492187)

[6. Exception Handling 35](#_Toc426492188)

[7. Logging 36](#_Toc426492189)

[7.1 Client Side Logging 36](#_Toc426492190)

[7.2 Server Logging 36](#_Toc426492191)

[7.3 Application Logging 37](#_Toc426492192)

[8. Custom Services 37](#_Toc426492193)

[9. Service Definition 38](#_Toc426492194)

[10. Session Management 38](#_Toc426492195)

[11. Appendix 39](#_Toc426492196)

# Introduction

## Purpose

The purpose of this document is to capture high level design details on IndusInd Mobile application. The document includes details on architecture & design considerations, Functional & Non-functional requirements, development & Integration approach and deployment strategy. The application is developed using Kony platform.

## Project Overview

IndusInd is building banking application for their customers on mobile platform. This is B2C application and the application helps their customer perform banking operations on their smart phones. Users of this application can perform payment & transfers, bill payments, view account balances, perform service requests, etc.

IndusInd Mobile application comes with rich user experience. Apart from the basic banking needs, it also gives wow feature like having video calling with customer, shake to get new offers, etc.

## 1.3 Scope

### 1.3.1 Platform Scope

Following platforms will be in scope for IndusInd Mobile Application:

**Native & SPA**:

Android Phone

Android Tablets

IPhone

IPad

Windows Phone

Windows Tablets

**Note**: The details on device versions and related OS details are listed in Technical Specification Document.

### 1.3.2 Integration Scope

The data integration between IndusInd Mobile application and backend systems will happen through the following:

1. Soap Services

2. Java Services

### 1.3.3 High Level Functional Scope

Please refer IndusInd bank mobile application – Functional specification document. On a high level, the below listed modules will be in scope.

* Registration & Login
* NLI
* Account Summary
* Credit Cards
* Payment & Transfers
* Bill Payments
* Service Requests
* Complaints, Feedback & Ratings
* Lead Management
* Notification, Alerts & Offers
* Profiler
* Segmentation & Menus
* Settings
* Admin & Reporting

## 1.4 Assumptions

The following assumptions will be considered for IndusInd Mobile Application.

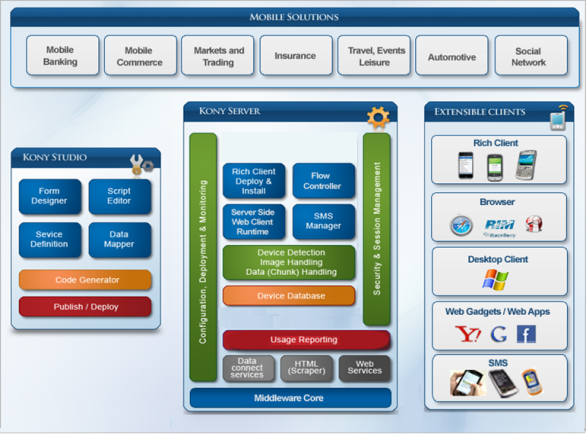
* Mobile application will interact with enterprise system using one of combination of services like SOAP, XML, REST, etc.
* All the data communication with external systems will happen through secured channel.
* IndusInd Mobile application will work in ‘English’ locale.
* Application to work in online mode and only limited functionality will work in offline mode.
* All the web services used will be given by IndusInd.
* IndusInd to help with test data for mobile application.

# High Level System Overview

## 2.1 Kony Development Model

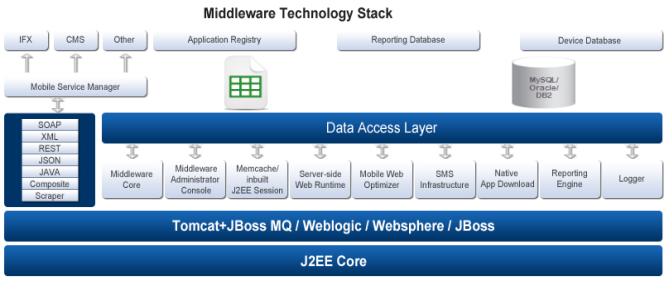
The below diagram depicts the high level development model on KonyOne platform. KonyOne offers a fully integrated application delivery mechanism, involving application development, testing, deployment, and delivery to the devices. This mechanism simplifies the development effort and helps achieve the requirement is minimal time. KonyOne platform consists for mainly three components.

* A development Studio that is used to design and develop the mobile applications.
* A Client Runtime component for each major device platform that enables the same mobile application to execute directly on the device.
* An Application Server that provides server-side functionality for the Mobile Website channel. It also provides the custom data integration and device support services.



## 2.2 KonyOne Server Architecture

KonyOne Server provides a comprehensive and fully integrated platform for deploying sophisticated mobile applications targeted for B2C or B2E. KonyOne Server is built over J2EE Core and provides complete access to enterprise data using SOAP, XML, REST, JSON, Java and other standards. The below diagram gives a high level overview on the KonyOne Server and internal components.

****

The details on the Kony One Server components are listed below:

* **Middleware Core**

Middleware/Server Core decides the process flow to be initiated and acts a mediator managing that process flow once a HTTP/HTTPS request is sent to the Server. Server Core consists of Security Filter, Flow Controller, Data Controller, Service Delegate and JSON Processor components.

* **Application Registry**

Application Registry consists of the application service definition file for each application.

* **Data Connectors**

The KonyOne Server provides out of the box support for standard interfaces such as SOAP, XML, JSON, REST, Scraper and Java API to back end data sources. Based on the Datasource, application developer will use appropriate service definition editor to extract the data from the external services using the XPath.

* **Memcache / In-built J2EE Session**

Memcache is used for Tomcat**/JBoss** based environments and In-built J2EE HTTP Session Management is used for WebLogic and Websphere environments. In both cases, session data is placed in the memory.

* **Server-side Web Runtime**

This component ensures that the business logic executed on the server is exactly same as the one that is executed on the client device (in case of the Native Apps).

* **Mobile Web Optimizer**

The Mobile Web Optimizer detects the device using the device detection capability, backed by the Device Database. Based on the device type, screen resolution and form factor, the Mobile Web Optimizer serves the appropriate Screen Layout, Markup(HTML5, XHTML/HTML) and Assets to the mobile web browser.

* **Logger**

The logger component is responsible for collecting the metrics from the messaging queue and persisting them to a RDBMS.

* **Device Database**

Device profile data is maintained and updated by a device group at Kony that monitors upcoming devices and platform releases. Device parameter data is obtained from manufacturers, where possible, and also by actual physical testing on the devices.

* **Native App Downloader**

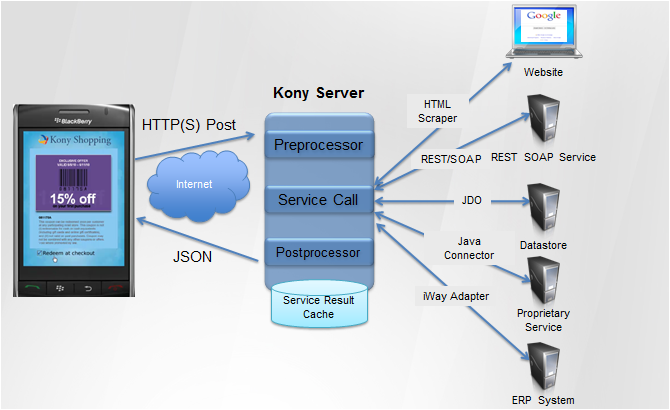
This component enables the Native App binaries to be directly downloaded from the KonyOne Server. This is applicable only for platforms (non-iPhone) which allow applications to be installed outside the app store.

* **Reporting Engine**

The KonyOne Server provides the capability to define and provide application-specific reports like Number of Visitors, Number of Page Views, Number of Native Apps Service Requests, Device Summary, etc.,

## 2.3 Mobile Application Working Flow

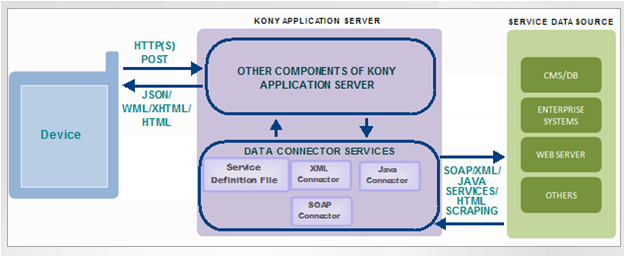
The below diagram depicts the request flow between mobile application and enterprise system. Kony platform has capability to consume any type of services including JSON, SOAP, REST, etc. and serve the mobile requests. If there are proprietary services from client then that can be consumed via Java services.



## 2.4 Services Flow

The attached diagram below depicts the data flow between device and enterprise systems via Kony Server. Kony Server has the following capabilities:

* Has different connectors to support enterprise systems.
* Few of the connectors are XML Connector, Java Connector, Soap Connector, etc.
* These connectors are responsible to request the enterprise systems and parse the response.
* All the data flowed between device and Kony Server will be in JSON format.



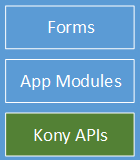
# Application Architecture & Design

The main objective of the project is to provide customers with an enhanced banking features on mobile phones and tablets including Android, iOS & Windows. On a high level, the application architecture can be considered on following aspects:

* Client
* Server

## 3.1 Client

The Kony Studio will be primarily used during the development. There are individual components involved which will be used to build the client side application. All the required components built for the application are described below on high level.



**3.2 Alternate Solution**

**N/A**

### 3.1.1 Forms

* These are the UI components for Kony mobile/tablet applications.
* Each form represents a view in mobile application.
* The display and rendering of any form would be applied at runtime by the requisite app modules.
* The dynamic skinning and theming information for all the forms would be picked up from a configuration file.
* These forms comprises of different widgets and containers.

**Example**:-

There will be lot of forms used to accomplish the IndusInd Mobile Application requirement. Few of the examples were listed below.

frmLogin – The form will be used for login purpose.

frmOTP - The form will be prompting user to input one time password.

frmRegister – Allows end customer to register with IndusInd Mobile application.

**Note**:-

All the form names in application will have ‘frm’ as prefix to identify the given object is referring a form as listed in example above.

### 3.1.2 Modules

* The modules will have all the Java Script files.
* They drive the application flow.
* All the client side validation and the business logic will be built using JS.
* Modules form the core in almost all of the functionality exposed by the Mobile application.

**Example**:-

Lot of modules will be created to accomplish the application requirement. Few of the examples were listed below.

login.js – The JS module will have functionality for application login and captures all the business scenarios.

globals.js - The JS module will have definitions on all the global variables used in the application.

constants.js – The JS module will include and lists all the constants required in application.

### 3.1.3 Kony API’s

* The Kony API’s will be used in modules.
* These API were cross-platform built to suffice a business need.

**Example**:-

Kony has defined many API’s to fulfill the requirements. These API’s were classified into many such as Application API, OS API, Math API, storage API, crypto API, etc. Few of the examples were listed below.

kony.net.invokeServiceAsync – This API is used for connecting Kony Server over network and process service request

kony.crypto – These API’s were meant for encrypting/decrypting data.

kony.store – These API’s were used to persisting data in device DB.

### 3.1.4 Skins/Themes

* Skins and Themes play important role in UI/UX.
* The suggested styles for forms and widget will be defined using skins.
* The rendering in terms of background/foreground, fonts, etc. will be defined here.

## 3.2 Server

The Kony Server plays prominent role in optimizing the request/responses. The Kony Server provides various connectors to help serve the request and process the response from enterprise systems. The below attached diagram depicts the important components in Kony Server. For more details on other components, please refer the high level system architecture.



### 3.2.1 Connectors

* The connectors are used for connecting external entities like web services, XML services, JSON Services, etc.
* Connectors comes built-in with Kony Application Server.

### 3.2.2 Service Definition

Service Definition is an XML file defining the following information:

* URL of the Web Service to be invoked.
* Protocol of the Web Service to be invoked.
* Authentication mechanism.
* Mapping information for the input and output fields.
* Configures pre/post processor to extend the base functionality.

**Note**:

Apart from the above listed components, there are many more components in Kony Server which were meant for different purposes.

## 3.3 Application life cycle Events

Kony application has life cycle events which will be automatically invoked when the application is launched, one after the other. All these events are optional and will be invoked only if they were defined.

With respect to IndusInd Mobile Application, following events will be used. Details were specified on the functionalities to be invoked during application life cycle.

### 3.3.1 Pre-App Init

IndusInd mobile application will receive push notification messages based on transactions performed. To receive the notification message and display their details, application should register for notifications. Use the following code snippet to register.

*kony.push.setCallbacks(object);*

The object passed in the call back API has 6 key value pairs.

**onsuccessfulregistration**: Invoked when registration with notification servers is success

**onfailureregistration**: invoked when registration with notification server is failed

**onlinenotification**: invoked when the notification is received when the app is online

**offlinenotification**: invoked when the notification is received when app is offline.

**onsuccessfulderegistration**: Invoked when deregistration with notification servers is success

**onfailurederegistration**: invoked when deregistration with notification server is failed

### 3.3.2 Post-App Init

The properties required to govern IndusInd mobile application will be stored in Kony Server. These property values will be loaded once by Kony Server and will be cached in server memory.

Application to invoke service call to load these configurable properties in device memory. Invoke the call to fetch the configurable properties file from post app init. Use the below given API to load these values.

*kony.net.invokeServiceAsync()*

**Note**: IndusInd Mobile application will have idle timeout set. Use the API specified below to set the idle time out for mobile application. The value for idle timeout will be from configurable properties.

*kony.application.registerForIdleTimeout(<timeout>)*

### 3.3.3 Deeplink Function

This event will be used to change the landing form based on certain criteria. Not applicable for IndusInd mobile application for now.

# Functional Modules

## 4.1 Registration

This module will be used for registering a user with in the IndusInd Mobile Application. The below details explains the use case for registration module.

### 4.1.1 Forms, Modules & Actions

The below grid lists the forms used in registration module. Accordingly, required modules for registration is listed in the grid.

|  |  |  |
| --- | --- | --- |
| **Form Names** | **modules** | **Description** |
| frmRegistration | registration.js | registration options |
| frmRegistrationNetBanking | registration.js | registration using netbanking |
| frmRegistrationViaCard | registration.js | registration via credit/debit card |
| frmRegistrationViaSocialProfile | registration.js | registration via HRV |
| frmRegistrationViaSocialProfile1 | registration.js | registration via HRV |
| frmOTPValidation | otp.js | otp validation |
| frmMPINSwipeValidation | registration.js | complete registration |
| frmDeviceSetup | common.js | set primary device |
| frmPrimaryAccountSetup | common.js | set primary account # |
| frmPreferredCard | common.js | set primary credit card # |

|  |  |  |
| --- | --- | --- |
| **Form: frmRegistration** | | |
| **Events** | **Method Name** | **Description** |
| onCircularClick | checkRegisteredCount | Function to identify number of users registered on particular device. |

|  |  |  |
| --- | --- | --- |
| **Form: frmRegistrationNetBanking** | | |
| **Events** | **Method Name** | **Description** |
| onClickOfSubmit | validateNetBankingDetails | function to validate the netbanking details and navigate to OTP form |

|  |  |  |
| --- | --- | --- |
| **Form: frmRegistrationViaCard** | | |
| **Events** | **Method Name** | **Description** |
| pre-show | preShow\_frmRegistrationViaCard | perform any UI logic, if required. For example, making the image as circle. |
| onClickOfRoundIconOnCard | changeCardUI | The function to change the UI for debit or credit card |
| onClickOfSubmit | validateCardDetails | Validate the card details and navigate to OTP form. |

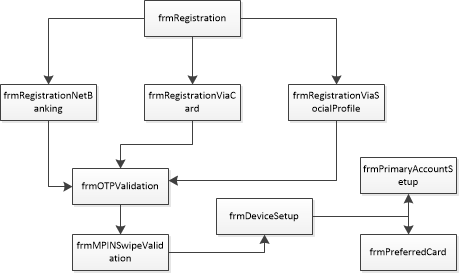
|  |  |  |
| --- | --- | --- |
| **Form: frmOTPValidation** | | |
| **Events** | **Method Name** | **Description** |
| init | N/A |  |
| pre-show | preShow\_frmOTPValidation | Function to request server for sending OTP. |
| post-show | postShow\_frmOTPValidation | Function to register with SMS Manager for reading SMS. |
| onClickOfSubmit | validateOTP | Function to authorize the user based on given OTP. |
| onClickOfResend | resendOTP | The function will be used to invoke to custom service and resend the OTP |

|  |  |  |
| --- | --- | --- |
| **Form: frmMPINSwipeValidation** | | |
| **Events** | **Method Name** | **Description** |
| init | N/A |  |
| pre-show | preShow\_frmMPINSwipeValidation | To execute logic before the form is shown. |
| post-show | postShow\_frmMPINSwipeValidation | to execute the logic after the form is shown |
| onClickOfSubmit | createUser | function to register and create mobile banking user. |

|  |  |  |
| --- | --- | --- |
| **Form: frmPrimaryAccountSetup** | | |
| **Events** | **Method Name** | **Description** |
| init | N/A |  |
| pre-show | preShow\_frmPrimaryAccountSetup | function to invoke the service and fetch related accounts. |
| post-show | postShow\_frmPrimaryAccountSetup | to execute the logic after the form is shown |
| onClickOfSubmit | setPrimaryAccount | function to set primary account for given user. |

### 4.1.2 Form Navigations

This section gives a high level understanding on the navigation between one form to another form.

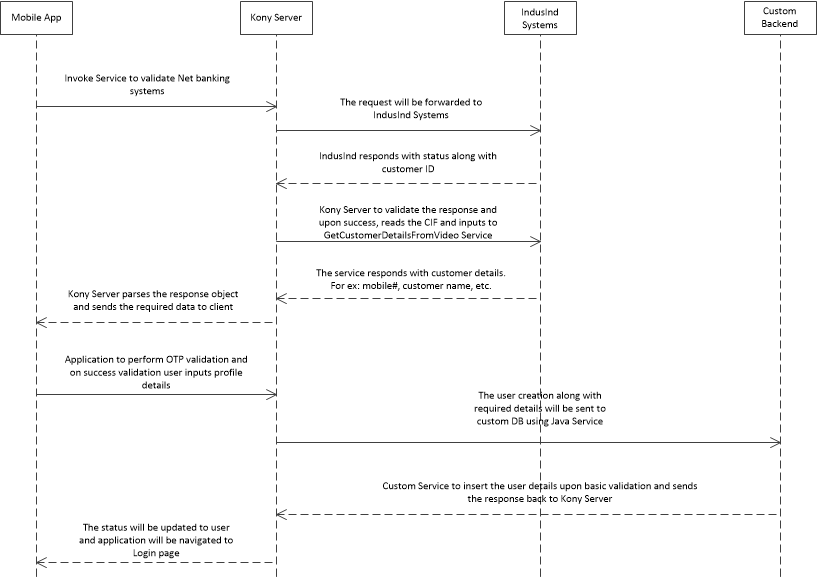


**Note**: The device back button will be disabled for all the forms.

### 4.1.3 Diagrams

**Registration using Net Banking**:

The below sequence diagram explains the positive flow on user registration process with mobile application using Indus Net banking System.



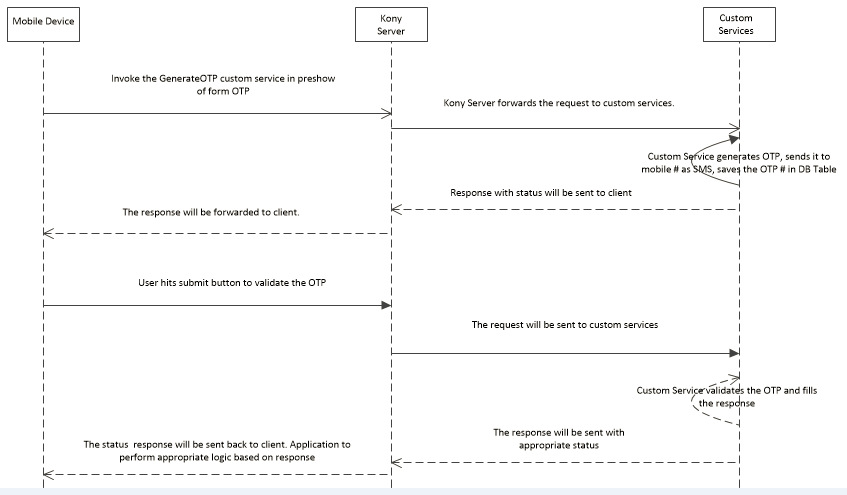
**OTP Implementation**

The below diagram details on

1. Flow diagram on OTP behavior

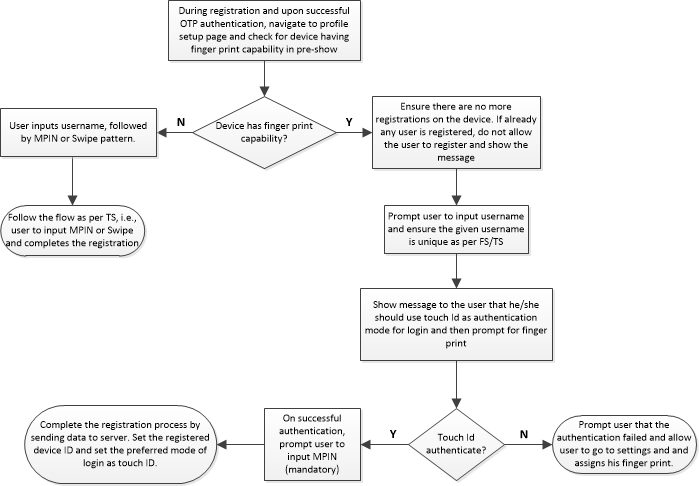
2. The sequence diagram depicting the process.





**Registration having Finger Print Capability**:

The below sequence diagram depicts the process flow when user has device having touch Id capability.



## 4.2 Login

This module will be used for login to mobile application. The below details explains the use case for login module.

### 4.2.1 Forms, Modules & Actions

The below grid lists the forms used in registration module. Accordingly, required java script files for login module is listed in the grid.

|  |  |  |
| --- | --- | --- |
| **Form Names** | **modules** | **Description** |
| frmAlreadyRegisterdUserMPIN | registration.js | registration options |
| frmAlreadyRegisteredUserSwipe | registration.js | registration using netbanking |
| frmChangeUser | registration.js | registration via credit/debit card |
| frmMpinLogin | registration.js | registration via HRV |
| frmSwipeLogin | registration.js | registration via HRV |

The below grids has details on the events/actions which will be perform on each of the form.

|  |  |  |
| --- | --- | --- |
| **Form: frmAlreadyRegisterdUserMPIN** | | |
| **Events** | **Method Name** | **Description** |
| onClickOfSubmit | loginWithMPIN | function to login |

|  |  |  |
| --- | --- | --- |
| **Form: frmAlreadyRegisteredUserSwipe** | | |
| **Events** | **Method Name** | **Description** |
| onClickOfSubmit | loginWithSwype | function to login using swipe pattern |

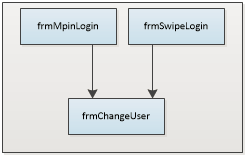
|  |  |  |
| --- | --- | --- |
| **Form: frmMpinLogin** | | |
| **Events** | **Method Name** | **Description** |
| pre-show | preShow\_frmMpinLogin | to identify the last logged in user from device DB |
| onClickOfChangeUserIcon | N/A | Just navigate to frmChangeUser form |

|  |  |  |
| --- | --- | --- |
| **Form: frmSwipeLogin** | | |
| **Events** | **Method Name** | **Description** |
| pre-show | preShow\_frmSwipeLogin | to identify the last logged in user from device DB |
| onClickOfChangeUserIcon | N/A | Just navigate to frmChangeUser form |

|  |  |  |
| --- | --- | --- |
| **Form: frmChangeUser** | | |
| **Events** | **Method Name** | **Description** |
| init | N/A |  |
| pre-show | preShow\_frmChangeUser | read the device DB and fetch all the user details who logged in on the current device. |
| post-show | postShow\_frmChangeUser | perform image loading for users |

### 4.2.2 Form Navigations

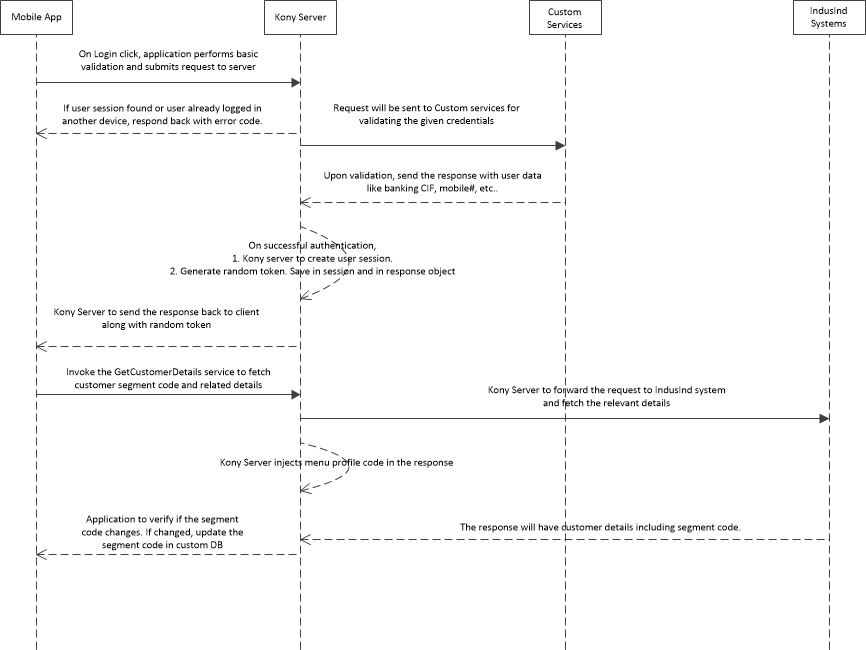
A simple representation on form navigations for Login module.



### 4.2.3 Diagrams

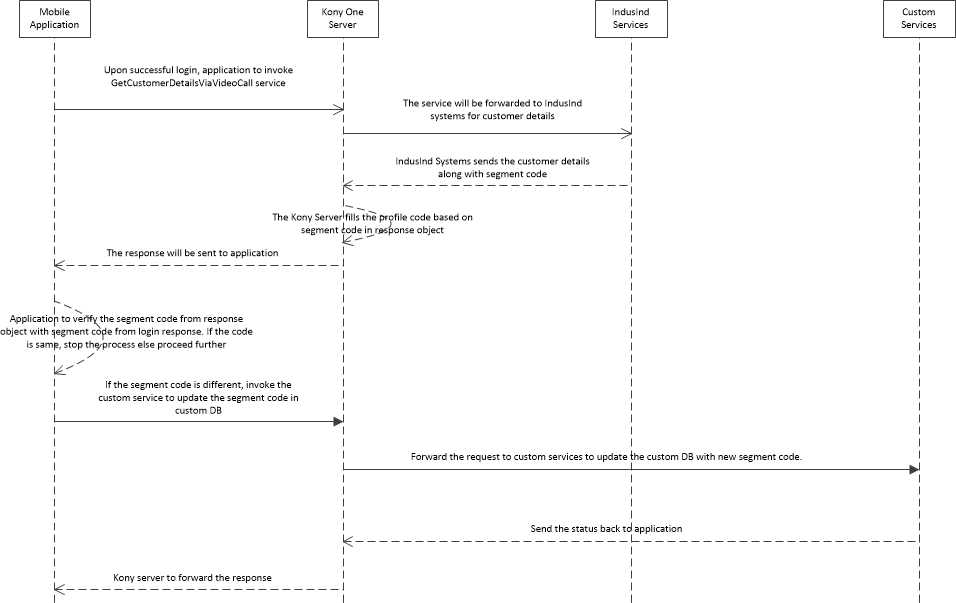
**Login**:

The below sequence diagram explains the login process.



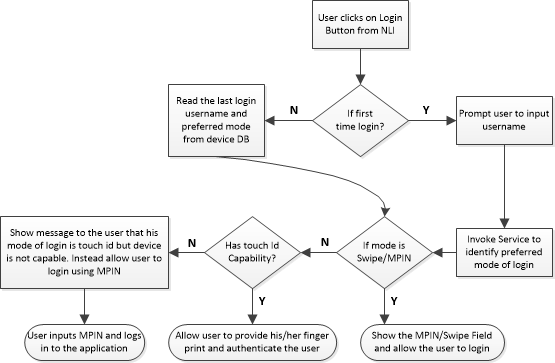
**Updating User segment in Custom DB**:

The below sequence diagram explains the process of updating the customer segment code in custom DB.



**Login having Touch Id capability**:

The below sequence diagram explains the flow when device has touch id capability.



## 4.3 Forgot Password

### 4.3.1 Forms, Modules & Actions

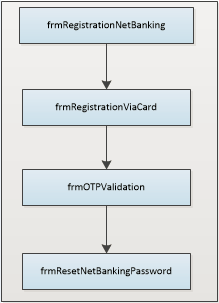
The below grid lists the forms used in forgot password module. Accordingly, required modules is listed in the grid.

|  |  |  |
| --- | --- | --- |
| **Form Names** | **modules** | **Description** |
| frmRegistrationViaCard | forgotPassword.js | validate the debit/credit card |
| frmResetNetBankingPassword | forgotPassword.js | to reset the netbanking password |
| frmOTPValidation | otp.js | otp validation |

|  |  |  |
| --- | --- | --- |
| **Form: frmResetNetBankingPassword** | | |
| **Events** | **Method Name** | **Description** |
| init | frmResetNetBankingPassword\_init | Function to bind events |
| pre-show | frmResetNetBankingPassword\_preShow | perform any UI logic, if required. |
| onClickOfSubmit | resetNetbankingPassword | to reset the netbanking password |

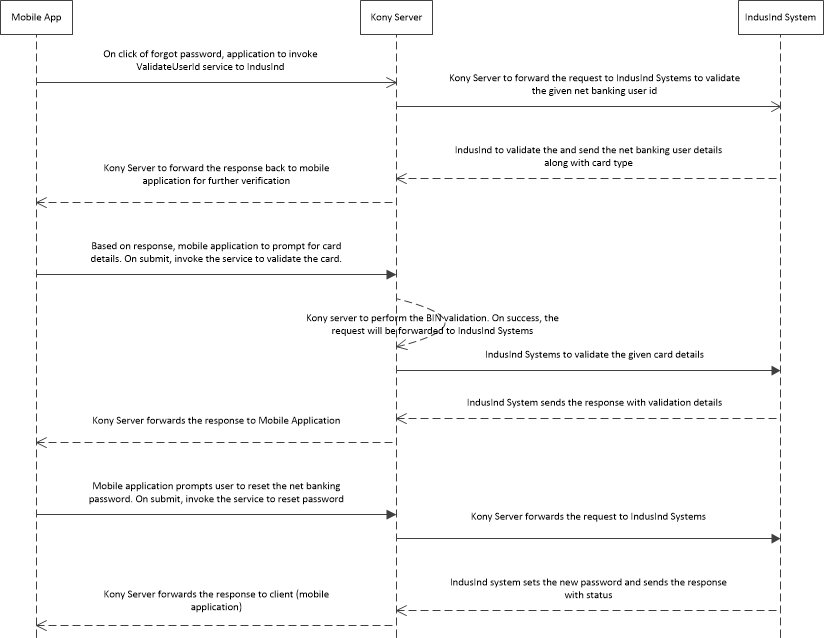
### 4.3.2 Form Navigations

This section gives a high level understanding on the navigation between one form to another form for forgot password module.



### 4.3.3 Diagrams

The below diagram explains the sequence of steps for forgot (net banking) password.



## 4.4 Account Overview

### 4.4.1 Forms, Modules & Actions

The below grid lists the forms used in Account Overview module.

|  |  |  |
| --- | --- | --- |
| **Form Names** | **modules** | **Description** |
| frmAccountOverview | forgotPassword.js | validate the debit/credit card |
| frmAccountsList | forgotPassword.js | to reset the netbanking password |

|  |  |  |
| --- | --- | --- |
| **Form: frmAccountOverview** | | |
| **Events** | **Method Name** | **Description** |
| init | frmAccountOverview\_init | function binding and initialization events |
| pre-show | frmAccountOverview\_preShow | perform any UI logic, if required. |
| onClickOfIOwn | onClickIOwn | Perform the logic to show I OWN tab. |
| onClickOfIOwe | onclickIOwe | Perform the logic to show I OWE tab. |
| onClickOfAccount | onClickOfAccount | Perform the logic to display account details |
| onClickOfDeposit | onClickOfDeposit | Perform the logic to expand the deposits. |
| onClickOfViewStatement | onClickOfViewStatements | navigate to account history page |

|  |  |  |
| --- | --- | --- |
| **Form: frmAccountsList** | | |
| **Events** | **Method Name** | **Description** |
| init | frmAccountsList\_init | function binding and initialization events |
| pre-show | frmAccountsList\_preShow | perform any UI logic, if required. |
| onClickOfCalendar | onClickOfCalendar | navigate to calendar view |
| onClickOfChart | onClickOfChart | navigate to chart view |
| onClickOfFilterIcon | frmAccountsList\_onClickOfFilterIcon | show the filter options |
| onClickOfSubmit | frmAccountsList\_onClickOfSubmit | Invoke the service to fetch the transactions based on given filter criteria |

### 4.4.2 Form Navigations

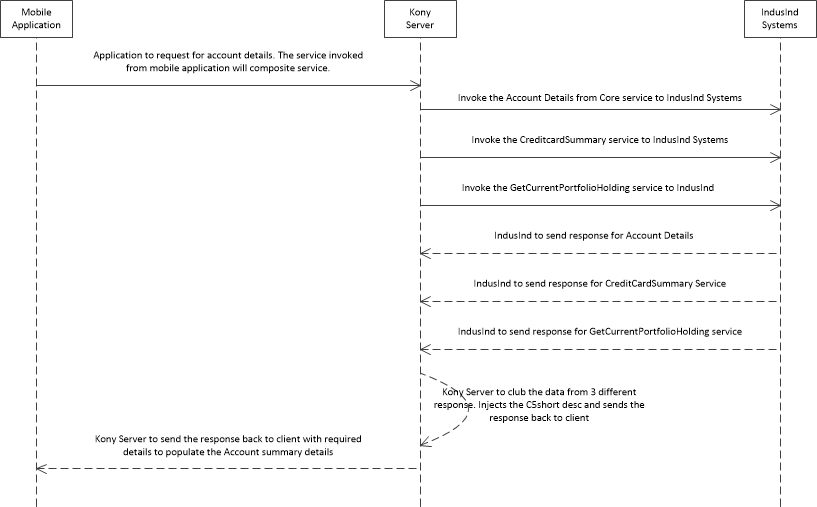
This section gives a high level understanding on the navigation between one form to another form for Account Overview module. The form navigation is very simple for account overview since lot of functionalities were accommodated in same form.

frmAccountOverview **->** frmAccountsList

### 4.4.3 Diagrams

**Account Overview**:

The below diagram explains the sequence of steps to fetch the account summary details. The data received will also have summary and details related data.



## 4.5 Payment & Transfer

### 4.5.1 Forms, Modules & Actions

The below grid lists the forms used in Payment & Transfer module.

|  |  |  |
| --- | --- | --- |
| **Form Names** | **Modules** | **Description** |
| frmTransferHome | TransferHome.js | Payment Transfers Home Java script Module |
| frmBeneficiaryList | BeneficiaryList.js | Beneficiary List Java script Module |
|  |  |  |
| frmSelectBeneficiary | SelectBeneficiary.js | Select Beneficiary Java script Module |
| frmTransferMoney | TransferMoney.js | Payment Transfer Java script Module |
| frmOneTimeTransfer | OneTimeTransfer.js | One Time Transfer Java script Module |
| frmTransferMyOwnAccounts | TransferMyOwnAccounts.js | Transfer My Accounts Java script Module |
| frmTransferConfirmation | TransferConfirmation.js | Transfer Confirmation Java script Module |

|  |  |  |
| --- | --- | --- |
| **Form: frmTransferHome** | | |
| **Events** | **Method Name** | **Description** |
| init | frmTransferHome\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmBeneficiaryList** | | |
| **Events** | **Method Name** | **Description** |
| init | frmBeneficiaryList\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmSelectBeneficiary** | | |
| **Events** | **Method Name** | **Description** |
| init | frmSelectBeneficiary\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmTransferMoney** | | |
| **Events** | **Method Name** | **Description** |
| init | frmTransferMoney\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmOneTimeTransfer** | | |
| **Events** | **Method Name** | **Description** |
| init | frmOneTimeTransfer\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmTransferMyOwnAccounts** | | |
| **Events** | **Method Name** | **Description** |
| init | frmTransferMyOwnAccounts\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmTransferConfirmation** | | |
| **Events** | **Method Name** | **Description** |
| init | frmTransferConfirmation\_init | function binding and initialization events |

### 4.5.2 Form Navigations

This section gives a high level understanding on the navigation between one form to another form for Payment and Transfer module. The form navigation is very simple for Payment and Transfer module.

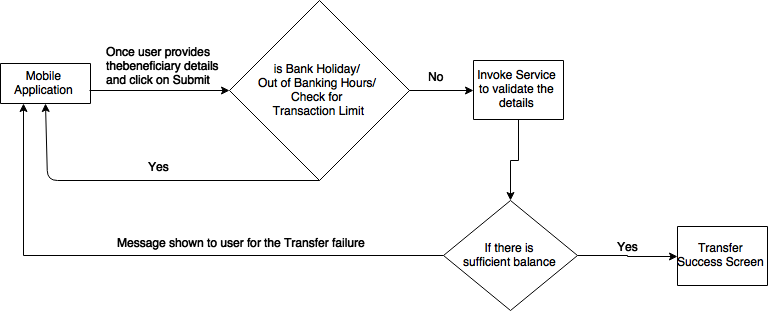
frmTransferHome **->** frmOneTimeTransfer **->** frmTransferConfirmation

frmSelectBeneficiary **->** frmBeneficiaryList **->** frmTransferConfirmation

frmTransferMyOwnAccounts **->** frmTransferConfirmation

### 4.5.3 Diagrams

**Payment & Transfer**:

The below diagram explains the sequence of steps for the Payment and Transfer module. 

## 4.6 Mobile/DTH Recharge

### 4.6.1 Forms, Modules & Actions

The below grid lists the forms used in Mobile/DTH Recharge module.

|  |  |  |
| --- | --- | --- |
| **Form Names** | **Modules** | **Description** |
| frmMobileRechargeHome | MobileRechargeHome.js | Mobile Recharge Home Java script Module |
| frmMobileRecharge | MobileRecharge.js | Mobile Recharge Java script Module |
|  |  |  |
| frmMobileRechargeConfirmation | MobileRechargeConfirmation.js | Mobile Recharge Confirmation Java script Module |
| frmMobileRechargeHistory | MobileRechargeHistory.js | Mobile Recharge History Java script Module |

|  |  |  |
| --- | --- | --- |
| **Form: frmMobileRechargeHome** | | |
| **Events** | **Method Name** | **Description** |
| init | frmMobileRechargeHome\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmMobileRecharge** | | |
| **Events** | **Method Name** | **Description** |
| init | frmMobileRecharge\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmMobileRechargeConfirmation** | | |
| **Events** | **Method Name** | **Description** |
| init | frmMobileRechargeConfirmation\_init | function binding and initialization events |

|  |  |  |
| --- | --- | --- |
| **Form: frmMobileRechargeHistory** | | |
| **Events** | **Method Name** | **Description** |
| init | frmMobileRechargeHistory\_init | function binding and initialization events |

### 4.6.2 Form Navigations

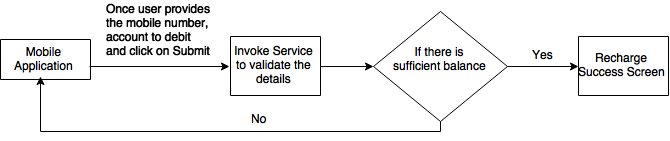
This section gives a high level understanding on the navigation between one form to another form for Mobile/DTH Recharge module.

frmMobileRechargeHome **->** frmMobileRecharge **->** frmMobileRechargeConfirmation

### 4.6.3 Diagrams

**Mobile/ DTH Recharge**:

The below diagram explains the sequence of steps for the Mobile/ DTH Recharge module.



# Caching

Since the launch of application, many of the data being used in mobile requires caching. Most of the data being cached in mobile will pertain to user or static data across application. The caching also applies in server since many of the data fetched from custom DB will be static in nature.

## 5.1 Client Side Caching

The following sub section describes on client side caching. The data cached will be for limited time. The details are listed in each of the sections below.

### 5.1.1 Application Launch

When the application is launched, a custom service ‘**LoadConfigurableProperties**’ will be invoked. The custom service loads the key value pairs from configurable property file.

* Read the response from service.
* The response will have all the key-values defined in configurable property file.
* These static values which will not change.

**Global Variable**:

* Use a global variable (for ex: **gblProperties**) to store the configurable properties.

**Cache Expiry**:

* The data will be cached till application life time.

### 5.1.2 User Details

When user performs login in mobile application, a custom service ‘**Custom\_AuthenticationWithMobileBanking**’ will be invoked. The custom service performs validates the user and on successful authentication, returns user related data.

* Read the response from service.
* The service response includes user details like banking/credit CIF, net banking userid, email Id, mobile #, customer segment code, menu profile code, etc.
* Once the data is processed, these data will be cached in device memory.
* The response will be in the form of JSON object.

**Global Variable**:

* Use a global variable (for ex: **gblUserDetails**) to store the user related data.
* The given global variable will also have CIF linked to the username.

**Cache Expiry**:

* The data will be cached when user successfully logs in.
* The data will be cleared whenever logout happens.

### 5.1.3 Account Details

On successful authentication during login in mobile application, ‘**AccountDetailsFromCore**’ service will be invoked to fetch details on the related accounts. The service response will have all the account associated with the customer and all the accounts associated with linked customer ID.

* Read the response from service.
* The service response will have all the accounts associated with the customer
* Once the request is processed, the account numbers will be cached in device memory.

**Global Variable**:

* Use a global variable (for ex: **gblUserAccounts**) to cache the user account numbers.
* The given global variable will also have account numbers which were linked.

**Cache Expiry**:

* The data will be cached when user successfully logs in.
* The data will be cleared whenever logout happens.

## 5.2 Server Side Caching

This section details on the server side caching. Many of the DB tables maintained in Admin were static and will be loaded once in memory. For example, data related to Banking Hours and Holiday Master are static. Data will be loaded in server memory during server startup and will be cached. These data will be refreshed once every day.

The list of data cached in server memory were listed in Technical Specification document. Following will be the approach to load the static data in memory.

* A servlet class ‘IndusIndMobileServlet’ will be created.
* The servlet will have init function defined where all the static data will be loaded one after the other.
* The servlet will be configured in web descriptor having load-on-startup set to true.

**Note**:

1. The listed details in section 5.0 (mobile & Server caching) are only high level information.

2. During development if we identify more data to be cached, caching will be performed without updating this document.

3. The list of data which will be cached in Kony Server were listed in IndusIndTS-General\_V1.docx.

## 5.3 Persisting data in Device DB

This section details on the data which will be saved in device DB. Please note that the data saved in device DB will only be accessible by application, irrespective of whichever the user is logging in. The following items will be applicable when persisting the data in device DB.

* The data saved in device DB will be specific to the device.
* Any user using the application will have access to device DB.
* The data in device DB can be accessed only from application.
* Use kony.store API to save the data in device DB.
* The data stored in device DB (for ex: username) will be encrypted.

The below section describes the data which will be saved in device DB for IndusInd Mobile application.

### 5.3.1 Last login User

When user logs in to the application successfully, device DB will be updated. The code snippet given below

kony.store.setItem(“LAST\_LOGGED\_IN\_USER”, <username>);

**Note**: The username will be encrypted before persisting in device.

### 5.3.2 Profile Picture

IndusInd Mobile Application will have option to save profile picture. The picture can be captured from gallery or camera. The profile picture will be saved in Custom DB as well as device DB. When the profile picture is changed or updated, use the below code snippet.

kony.store.setItem(<username>, <base64\_rawbytes>);

**Note**: profile picture will be base64 content converted from profile picture rawbytes. Data need not encrypted.

### 5.3.3 Users using the Device

IndusInd Mobile Application allows same device to be used by multiple users and one user can use multiple devices for login. To keep track of users using the device, username will be saved in device DB.

var users = {username1:user1, …., usernameN:usenN};

kony.store.setItem(“USERS\_LOGGED\_IN\_DEVICE”, users);

**Note**: The username will be encrypted before persisting in device.

# Exception Handling

Application to perform check for exception possibilities and handle it in application. Following will be applicable.

* As defined in section 7.2, use the custom API to capture errors.
* Any of the service failed due to any errors should be handled.
* All the response coming from server should have error message along with error code, if applicable.
* The error message as received from server will be displayed to the user.
* Use try/catch statements, wherever required. Do not use try/catch unnecessarily.
* All the exceptions related to server will be captured in server logs.

# Logging

There are different aspects for logging application for mobile application within the KonyOne platform. Each of these are discussed below:

## 7.1 Client Side Logging

Kony has exposed API to log the events performed by user in mobile application. This API commonly works for all the platforms which can be used for debugging purpose. To view these logs, developer can use native tools to view the client side logs. Below is the Kony API to print the contents.

***kony.print(“message”);***

IndusInd mobile to use a simple custom API which makes the debugging simpler for mobile application. The code snippet for custom API is listed below.

*indus.print = {*

*log : function (msg) {*

*kony.print(arguments.callee.caller.name + ": " + msg);*

*},*

*error : function (msg) {*

*kony.print(arguments.callee.caller.name + ": \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");*

*kony.print(arguments.callee.caller.name + ": " + msg);*

*kony.print(arguments.callee.caller.name + ": \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");*

*}*

*}*

Application developer instead of using kony.print API, they should starting using indus.print.log API or indus.print.error API.

**Note**: The final binary which will be uploaded in app store will be generated by setting the property ‘Remove Print Statements’ to ON and in release mode.

## 7.2 Server Logging

KonyOne Server does support 2 modes of logging, i.e., file logging and DB logging. It uses Log4J and does supports different level of logging i.e., DEBUG, INFO, ERROR, etc.

The server logging can be controlled using middleware-log4j.properties. This file will be part of %middleware.home%/middleware/middleware-bootconfig folder. Details on each of the server logging is described below.

**File Logging**:

The file logging is suggested to be used during DEV, SIT and/or UAT. This mode of logging is highly not recommended for production mode.

**DB Logging**:

If this logging is turned on, all the logs captured in Kony Server along with pre/post processor code will be saved in database. To maximize the performance, the DB operations will be performed using separate process by reading log messages from JMS queue. It is recommended to use DB logging than file logging during production.

**Note**:

1. IndusInd mobile application will use DB logging in production & UAT environment.

2. IndusInd mobile application to use file logging in DEV & SIT environments.

3. Kony Server reporting module will use DB (only) to generate usage reports.

## 7.3 Application Logging

Apart from Client Side Logging and Server Side logging, IndusInd mobile will use additional logging mechanism. The data logged using this mode will be used for Admin reporting and audit logs purpose. The following items are applicable for this.

* The logging will be applied in post-processor.
* A base post processor class will be used for this purpose.
* The class will be responsible to push the logs in custom DB.
* All the other post processor required for the application will be extended from base post processor.
* The feature specific post processor will be configured in service definition.
* If no post processor is required, base post processor will be configured in service definition.

# Custom Services

Apart from using soap services, IndusInd Mobile Application has lot of dependencies on custom services. To fulfill the requirement, a custom DB is maintained in Kony Server with 100+ odd tables. Many of these DB tables were independent of each other and were maintained for master data. Few of them are used for admin and reporting purpose.

Following items are applicable for IndusInd Mobile Application:

* All the custom services were based on Java Connectors.
* Most of the custom services used in the application are meant for fetching data from DB.
* One of the custom service will be used for loading configurable properties.

All the custom Services were identified and captured in the document attached below.



# Service Definition

Most of the services used in IndusInd Mobile Application falls under 2 category.

1. SOAP Services

2. Java Services

**SOAP Services**:

These services were exposed by IndusInd. These services talk to IndusInd Backend Systems like Core, Net Banking, DCMS, etc. The attached services has the details on the web services used in different modules across the application.

<TODO: attach the IndusInd Services>

**Java Services**:

To fulfill the IndusInd Mobile Banking requirement, few data is stored in separate database. To serve the mobile application with these data, Java services will be used. The attached services has the details on the Java services used in different modules across the application.

<TODO: attach the Custom Java Services>

# Session Management

The HTTP requests are stateless and to keep track of user identity, session management will be used. The following items will be applicable with regards to session management:

* Session timeout in Kony Server will bet set to 30 minutes.
* The session will be shared across multiple Kony Server instances.
* Memcache will be used to share the session data across server instances.
* User specific data will be cleared from session on logout.

10.1 Data in Session

The following section describes the data being saved in session for mobile application. It also describes the scenarios when session data will be created, updated or deleted.

**Session Creation**:

On successful login, a new session will be created which will be saved in memcache.

**Session Update**:

On every post login services,

* A random token will be generated.
* The token will be updated in session.

**Session Delete**:

The user specific session data will be delete/invalidated in following occasions.

* When user perform logout in mobile application.
* When idle time out is triggered in mobile application.
* When user tries to login simultaneously in mobile application.

**Note**: The idle time out property will be set in mobile application which triggers logout.

# Appendix