Attachment 2:

Mobile Application Technical Design Document

Table of Contents

I. Purpose 3
II. Architectural Requirements / Design 3
A. Kony Mobile Fabric 3
B. JFS Systems of Record 4
C. Communication Modal 4
Services 4
D. Deployment Environment 5
E. Logging 5
F. Error Handling 5
G. Error handling on Device Side 6
H. Binary Generation and Security 6
I. Impact on App Performance 6
J. Data Store 7
III. Use Case and Component Requirements / Design 8
A. App version Check 8
B. Maintenance and Availability of Servers/Services 8
C. Login 8
D. Enabling Touch ID for authentication 9
E. Encryption Algorithm and Keys 9
F. Registering for Push Notifications 9
G. Sending Push Notifications Based on Preference 11
H. Idle Time Out 11
I. Logout 11
J. Analytics 11
K. Home Screen 12
L. App Menu 12
M. Mobile One Time Password Protection Mechanism 12
N. Customer alerts and county bulletins badge 12
O. Payment history 12
IV. References 13
V. Technology Standards 14

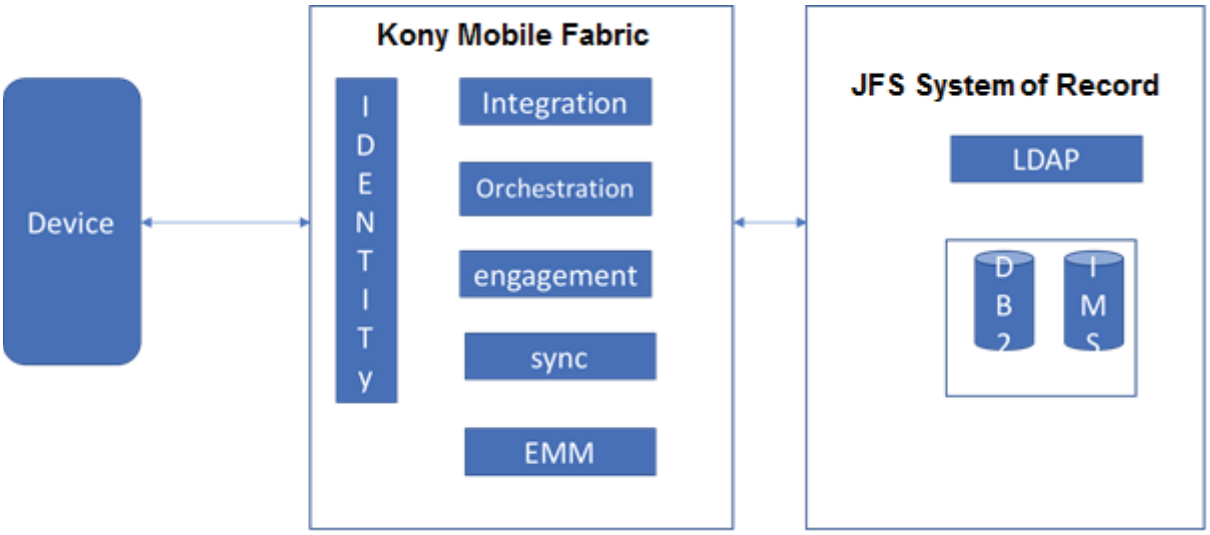
I. Purpose

The purpose of the document is to capture all the software design requirements in scope for the project to be delivered from the business and technical stand point for the SETS Web Portal – mobile application.

This document also focuses on capturing the software system requirements including architecture, system design and external interfaces.

II. Architectural Requirements / Design

A. Kony Mobile Fabric



Kony Mobile Fabric provides enterprise security and complex system integration services and allows developers to focus on building application experiences. This is accomplished by providing a powerful set of services to handle identity, integration, orchestration, data sync, and messaging. When these services are configured within the Kony Mobile Fabric, they can easily be incorporated into a mobile application using any third-party application development tool using our SDKs or direct REST API interface.

The following are the five major services offered by Kony MobileFabric:

- **Identity**: Authenticate and authorize application users including Salesforce, Active Directory, SAP, or other third-party identity providers that support Security Assertion Markup Language (SAML.)
- **Integration**: Securely connect the application to any back-end data using a variety of connectors for standard services such as REST, SOAP, and JSON end points and for enterprise connectors such as Salesforce and SAP. Custom connectors can also be built

using Java code to handle any atypical integration requirement. The integration layer will be used to connect to DB2 via the SOAP services exposed to fetch/write information from DB2 and IMS system of record.

- **Orchestration**: Optimize application performance by creating new services which need to be built as server-side composite services and workflows including the ability to execute custom business logic to server the mobile business use cases.
- **Sync**: Enable application to work offline by keeping a copy of relational data structures on the device. Securely synchronize changes between end-user devices and enterprise databases or web service-enabled systems.
- **Messaging/Engagement**: Engage with users over cross-platform push notifications, SMS, and email. This service includes the ability to track the effectiveness of messaging campaigns. Collection of user information and behavior analytics will enable better target messaging, based on user segmentation rules and location defined by geo-boundaries or iBeacons.

All web services specific to SETS provided by ODJFS will be integrated with the Kony Mobile Fabric server. The mobile application invokes these services that are available from the Kony Mobile Fabric.

B. JFS Systems of Record

LDAP: There is currently an LDAP system that is used in the authentication process.

IMS Mainframe system: Case/order information, personal information, health insurance information, employer information, child information, payment information, payment history and other case information all currently reside in the IMS system.

There are SOAP services (see references section) to fetch some of the data, and new services will be developed (see references section – data mapping document).

DB2: SOAP services will be exposed to read/write from DB2. The County Bulletin Board, News Flash and messages are pulled and saved from/to DB2. The connection to DB2 can also be handled using Java services. The current Java services need to be converted as SOAP/REST services for the mobile application usage.

For a list of services that will be needed by the mobile application, please refer to the Data Mapping document in the References section, showing the data elements on the screen to service level mapping.

C. Communication Modal

ODJFS gives access to the services necessary, and documentation/WSDLs/connection details would be shared for development.

Services

Device sends request to Kony Mobile Fabric over https post.

Kony Mobile Fabric Server interacts with the web service and posts the request/query over https.

Kony Mobile Fabric Server processes the request and makes appropriate call and generates the return data from the end point (which could be SOAP/RESTful services) and sends back the result object as a JSON/SOAP back to the device.

Sample JSON Response:

Sample JSON response looks as follows with name-value pairs:

{"Status":"SUCCESS","opstatus":0,"cacheid":"10545b6a6-362e-455a-afa2-9bd8dc579f5e"}

In the above response, “Status” is the name of the parameter and “SUCCESS” is the value for the same. “opstatus” is the generic Kony Specific output status, and will have error codes specific to Kony platform. Cache id is the session key for the user’s session in the server and is passed from the application to server to every subsequent service call. Other data are in session and are not passed to device.

D. Deployment Environment

The deployment environment is maintained by ODJFS. The environments that will be used for this application are

DEV – for development purpose. The entire dev team has access to this environment and be able to deploy server side artifacts.

SYS – to be used by the QA team to test the application.

UAT – used by ODJFS and its business team to validate the builds.

PROD – the production servers.

The Mobile Fabric installation itself is a separate contract between JFS and Kony. Documentation from that contract can be referenced for further information.

E. Logging

In Java services where applicable, exceptions that are caught are logged in error mode in the KMF. The exceptions will not contain any data related to the user. It will have only the exception stack trace. The Mobile Fabric runs in error mode to log only errors and above.

F. Error Handling

The Kony Mobile Fabric server (referred as KMF) interacts with the ODJFS back-end systems based on the request that is being sent from the device. On the KMF, the return code is checked for the response that is returned from the back end ODJFS Systems. Based on the return code, the appropriate Error messages are displayed to the user on the device. When there are error response/unhandled exceptions that are identified by the Mobile Fabric, the system logs the operation status (called opstatus)

After invoking a service, if the response from KMF contains the key 'message', then the value for that key is the error message that needs to be shown in the application.

Error messages below to be used if there is no error message returned from service:

Message to display:

Service is currently unavailable, please try again shortly.

Thank you.

Scenario:

Check is made and determines that the user does not have Wi-Fi or Cellular service?

Message to display:

Your data service is not available. Please try again when you have service.

G. Error handling on Device Side

Device errors are handled at the application level in JS. This will be displayed as the generic error message based on the verbiage that will be provided by ODJFS.

Any error other error 'message' will be a generic error message that will be used throughout the project: "The service is currently unavailable. Please try again."

H. Binary Generation and Security

There is an option in Kony Visualizer to build the application in protected mode. Protected mode offers several security features that secure the binary at build time by including multiple self-protection security mechanisms. Additional security mechanisms are provided through the use of White Box Cryptography to protect application business logic and source code.

If an application attack is observed, the security mechanism exits the application. For more information on protected mode please refer to the following:
http://docs.Kony.com/Konylibrary/Visualizer/Visualizer_user_guide/Default.htm#ApplicationSecurity.htm

I. Impact on App Performance

While enabling security features in your application ensures attacks are prevented, the application's start-up time may slow. The following image provides insight on the performance if the *Protected Mode* option is enabled.

Application Bootup Time Comparisions in milli seconds (iOS)			
Device Model Specs & OS	**Release Mode**	**Protected Mode**	**Delta time between Release Mode and Protected Mode**
iPad 3	2026	3794	**1768**
iPhone 5c	1220	2503	**1283**
iPhone 6+	624	1208	**584**
Application Bootup Time Comparisions in milli seconds (Android)			
Device Model Specs & OS	**Release Mode**	**Protected Mode**	**Delta time between Release Mode and Protected Mode**
Samsung Galaxy S5 (Android OS v5.0)	1520	3440	**1920**
Samsung Galaxy S3 (Android OS v4.1.2)	2905	4075	**1170**
Micro Max Android One (Android OS v6.0)	1830	3739	**1909**

J. Data Store

A data store is a storage area on the mobile device that is capable of holding persistent data for the mobile application. The data store is available on the mobile device at a location that is dependent on the underlying platform. The data store is not directly exposed to the users.

The underlying platform of the mobile device maintains the integrity of the persistent data stored in the following scenarios when:

- the application accesses the data
- the device reboots
- the battery is changed
- the application upgrades:
    - **Native Clients** - data stored earlier is not deleted when an application upgrades.
    - The actual storage used to store this data depends upon the underlying Operating System:

Operating System	Storage Space	Accessible to other applications
iPhone	Within a file outside the application sandbox. The storage area is shared by multiple applications.	Yes

Android/Android Tablet	Within a file in the application sandbox. The storage area is unique to the application.	No

The previous stored data is lost (permanently) in the following cases:

- the application is deleted from the device
- The *Kony.ds.delete* API is used to delete the data.

III. Use Case and Component Requirements / Design

A. App version Check

During application launch, a service is invoked to check for an update available for the version of the application that sits on the device. The service returns a flag that prompts if the upgrade is optional/mandatory. For a mandatory upgrade, the application will quit on pressing ok button. The user has to upgrade the application from the store and can relaunch the application. For an optional upgrade, the user can choose to upgrade or dismiss the alert and continue with current version of the application.

This call also gives back the public key back to the mobile application, which will be used to encrypt sensitive data on the mobile device.

B. Maintenance and Availability of Servers/Services

All the services exposed to the mobile application will have a pre-check on the Kony Mobile Fabric server to see if the time on the server is between 4AM – 6 AM. If so it will return a maintenance flag as true and the mobile application then shows a maintenance screen. The user can then logout and try to login again after the maintenance period. The maintenance period will be a configurable property on the server.eg:

maintenanceFlag = true

startTime = 04:00

endTime = 06:00

Even before the user can login (authenticated), a call to a PING service is made to ensure that the services are available. Internally, the service checks whether IMS is up and running. If the ping service times out then the maintenance screen is shown to the user.

C. Login

When the application is launched and if ‘isTouchEnabled’ (local storage) is not true, the user enters the user name and password.

If the value of ‘isTouchEnabled’ is true, this means that the user has enabled the touch ID feature previously, and local authentication is done to verify the user. If the user is using the application for the first time (known if ‘isFirstTimeUser’ is not true in local storage) then the

enable touch ID page is shown after the user successfully authenticates. If the user enables touch ID, 'isTouchEnabled' is set as true in the local storage.

Services are invoked to validate the credentials of the user as mentioned in the data mapping document in the References section.

D. Enabling Touch ID for authentication

When the user enters user name and password and after successful authentication, the application will present the option to sign in with touch ID (on devices and OS versions that are supported by Kony for touch ID), and 'isTouchEnabled' is saved in local storage with a value true.

If the user logs in and then enables touch ID to ON from the hamburger menu, 'isTouchEnabled' is saved in local storage with value true. (A value of false will be stored if the user turns off the feature or clicks on 'may be later' button after login.)

The user name and password are encrypted and stored in the local storage using the RSA 2048 algorithm.

E. Encryption Algorithm and Keys

When saving the username/password in the application, the encryption method used will be RSA 2048 to encrypt and store the user name and password on the device in the local storage. A public and private key pair will be generated for the use of the app before the implementation begins. The public key will be embedded in the application and also transmitted to the mobile application through the application version check service call to the Mobile Fabric. The private key will be on the KMF server. The user name and password are sent in encrypted format to the KMF server. On the Kony server the user name and password are first decrypted and sent to the login web service.

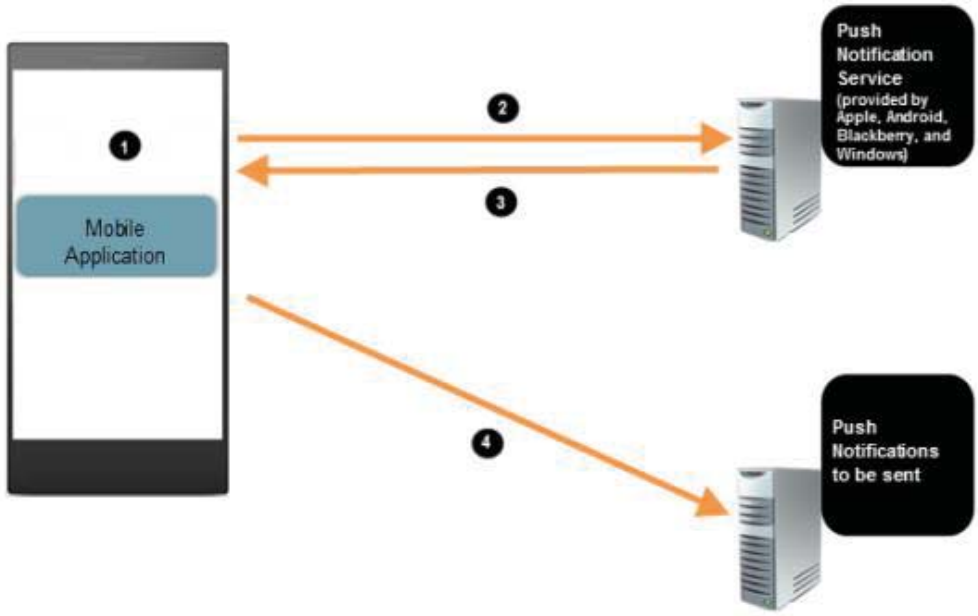
F. Registering for Push Notifications

On successful authentication, the user is registered for push notifications. If it is the first time, then the user is prompted by the underlying operating system (iOS or Android) to allow or not allow the permission for push notifications.

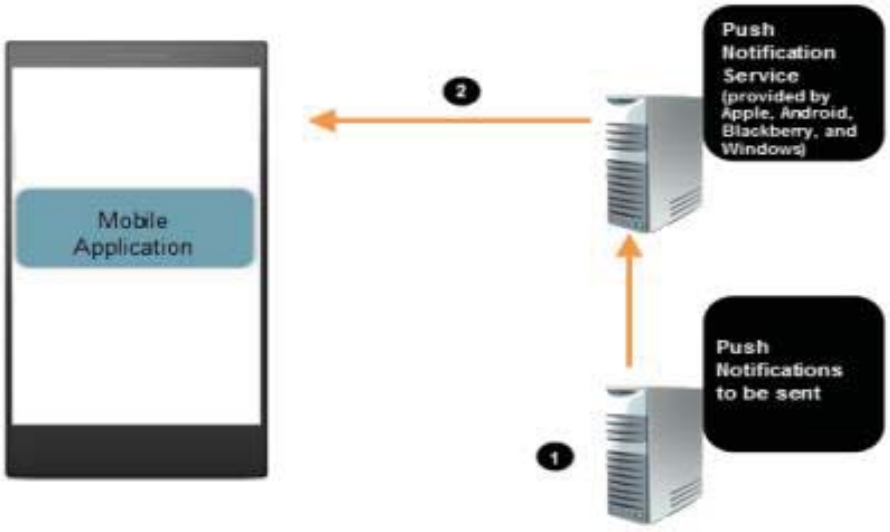
If the user allows, they will be registered for push notifications, otherwise they will not. If the user does not allow and wants to later enable, they have to do so from settings application, selecting the application and enabling notifications.

Sending push notification requires the application to register with a push notification service as the first step and also register with Kony Mobile Fabric engagement server. When a notification has to be sent to the application, it can be submitted to Kony Mobile Fabric which in turn sends to the push notification service.

The diagrams below help explain the flow. The Kony engagement server is the third-party application server shown in the diagrams.



1 - push.setcallback API is called on the device.
2 - push.register API is used to register for Push Notifications with the Push Notification Service provider.
3 - On successful registration, a unique identifier is returned by the Push Notification Service provider.
4 - The application sends the identifier to the third-party application server.



1 - Third-party application server send the Push Notifications to the Push Notification Service provider.
2 - The Push Notification Service provider send the Push Notifications to the device.

For iOS the application registers with APNs (apple push notification server). For Android, it registers with GCM (google cloud messaging server). The application gets the token ID from either apple/google and then calls a service on the Kony engagement server to register the token with the engagement server. The mobile application gets back a Kony subscription ID (KSID in short). In this service call to the Kony engagement server, the email ID of the user is passed that will be used as a reconciliation key on the Kony engagement server. This way, when push notifications are to be sent to the users, the engagement server can use the email IDs rather than KSIDs. Sound, badge or banner is allowed for push notifications.

G. Sending Push Notifications Based on Preference

The user can set push notification preferences within the application.

Jobs will be configured (which are public operations) on the Kony Mobile Fabric:

- To run periodically (either they are scheduled on Mobile Fabric to run at specific intervals of time or the existing batch jobs for the website will invoke these jobs)
- To query and find the list of users (email IDs) to whom the push notifications are to be sent.

It is important to get the list of email IDs to determine to which devices the push notification has to be sent. There have to be services that will return the email IDs of users to whom the push notification has to be sent for particular criteria. For example, to get the email ID of all users to whom notification needs to be sent for payment due; similarly for missing personal information and when there is a new message.

These notifications include alerting users when there is a payment due, when any profile information is missing and when there is a new message to the user.

H. Idle Time Out

The application is configured to have an idle time out of 5 minutes for all authenticated features in the application. After idle time out, log out will be invoked.

I. Logout

When the user logs out, the service logout is invoked and the session on the server is cleared. Any cache/global variables on the device are also cleared.

J. Analytics

The delivered Mobile Fabric analytics will be used to capture metrics. The application will automatically capture form entry, exceptions, and crash logs for iOS and Android.

K. Home Screen

The home screen shows important and relevant information to the user from different service calls. The number of service calls and their orchestration is mentioned in the attached Data mapping excel sheet, in the References section. The data mapping document shows how the data would be mapped if existing services are exposed as SOAP services.

Given the complexity and number of services, we orchestrate the services identifying their execution as either sequential or concurrent. It is recommended to have services that minimize the number of service calls and so not exactly a replacement to existing IMS connectors.

L. App Menu

When the user taps on application icons that will show data to the user (home, messages, pay history), the relevant services are invoked and data is refreshed on the screens. This is to clear any cached data during login or stale data and to then show the latest information to the user.

M. Mobile One Time Password Protection Mechanism

In flows like registration, forgot user id, and forgot password (re-authentication process), the requirement for captcha can be replaced by sending a onetime password to the users email ID or send SMS to the phone.

Two new services will be built in the back end–

- To send OTP (one time password) to the email ID provided or SMS to the phone.
- To validate if the user entered OTP is same as the one originally sent to the user's email ID or the text send to the phone.

N. Customer alerts and county bulletins badge

County bulletins will be fetched from a SOAP/RESTful service call to DB2. Customer alerts are fetched from a SOAP service call currently. On the home screen, there are two badges to be shown:

- Number of county bulletins
- Customer alerts.

When a user clicks on either of these to view the county bulletins or the customer alerts, the badge on the home screen for that item can be cleared and will remain so for the duration of the user session. If the user has the county bulletins/customer alerts, each time the user logs in the badges will be displayed until it is resolved by the county staff.

O. Payment history

The payment history will be shown as a pdf document with up to the 24 months of data. The Adobe ES4 service will be invoked to return base64 representation of the pdf. The mobile

application uses the base 64 to render the pdf. The way it is shown is dependent on the native behavior of the underlying OS. IOS devices usually can show the pdf within the application in browser view. Conversely, in Android the pdf will have to be downloaded and the user should use their native pdf reader to open and view the file. The native behavior cannot be changed and the users on their devices are used to that experience. Once the user opens the pdf file, he/she can use native options to print, email, or share.

P. New Registration

The Mobile Application Registration process will follow the Web Portal Registration process outlined in Attachement 4 Web Portal Registration Enhancement Business Requirements Document.

During the registration process, the user will be prompted to enter their Ohio Driver's License, State ID or Key Number, last 4 of SSN, DOB, First character of user's last name and email address. Once the user has passed field level validation additional information will be required to completed the reqistration process. Customers without an approved Ohio Id will be provided an altrernate registration process.

IV. References

Please see the below documents for more information.

Document Name	Version	Document	Author
Kony SOAP services	1.0	Kony SOAP Services.xlsx	
Data Mapping	1.0	DataMapping.xlsx	

V. Technology Standards

The following table defines the software technology standards that JFS requires the Contractor to use for this project.

IT Capability	Technology Platform
Development Platform	
Front End Development Platform	Kony Visualizer
Backend Development Platform	Kony Mobile Fabric
Build Tools	
Sprint / User stories and Requirements	Jira
Project documentation & collaboration	Confluence
Version Control	Bit Bucket
Continuous Build and Integration	Jenkins
Artifact / Dependency Management	Nexus Repo
Integration and Monitoring	
Application Integration	IBM Integration Bus
Logging and Monitoring	Splunk
Testing	
Automated Testing	HP Unified Functional Tester (UFT)
Mobile Device Management (MDM)	
MDM	Air watch
App Store	
iOS	Apple Store (Public and Enterprise)
Android	Google Play store
Windows	Windows app store