February, 2020

Revision History

Date	Document Version	Description of Releases and Updates
02/25/2020	1.4	**Updated for Kony Fabric V9 Foundation Release Fix Pack One**
16/12/2019	1.2	**Updated for Kony Fabric V9 Foundation Release Hot Fix**

Date	Document Version	Description of Releases and Updates
09/30/2019	1.1	**Updated for Kony Fabric Fabric V9 Foundation Release (M2)** Console • Added support to simplify the object model definition - field names and types • Added support to Generate Data Model by importing an Excel file. • Added support to Enumeration Data Type for Object Services • Added support to Cloud access control to Fabric environments • Added support to rotate appkeys: Separate App Key/App Secret for Native and Web Channels Identity • Added support to Sign-Up for Kony User Repository • Added support to enable integrity check for identity services to enable server to server communication • Added support to configure Customer 360 identity adapter • Added support to configure MFA for the Custom identity adapter • Support to download the reconfigurations made to an app before publishing. • Added support for the parameter prompt as part of OIDC authorization to improve the logout flow while using the OAuth Provider. • Added support to create the app service document based on the alias hostname.

Date	Document Version	Description of Releases and Updates
07/23/2019	1.0	**Updated for Kony Fabric V9 Foundation Release (M1)**

Date	Document Version	Description of Releases and Updates
07/23/2019	1.9	**Updated for Kony Fabric V8 SP4 JulyFP** Console • Support for Generating Separate App Key/App Secret for the Web and native channels. Kony Enterprise App Store • Support for Hosting Web apps on EAS.
06/10/2019	1.8	**Updated for Kony Fabric V8 SP4 JuneFP** Console • Support for Manage Service Profiles (Export/Import service profiles and exchange them between environments. Kony Enterprise App Store • Support to configure Display Name of apps while uploading app binaries. • Support to EAS source app for Native.

Date	Document Version	Description of Releases and Updates
04/29/2019	1.7	**Updated for Kony Fabric V8 SP4 AprilFP** Console • A new option has been added to write custom pre-processors and post-processors by using Rules. Previously, only the Java and JavaScript code options were available in Console. • Support for Clustered mode in MongoDB Adapter has been added. • The Kony Fabric landing page has been simplified and enhanced. • Audit Logs have been enhanced with support for pagination and export options. Integration Server • Performance enhancements in Object Services • Usability Enhancements to Service List • Added ability to Export and Import - Configuration Parameters • Ability to Export and Delete - Orphan Storage Schemas has been added. • Out-of-the-box support for batching records has been added as part of Database Adapter for MySQL. • A sample app for managing data in Storage Objects has been developed to facilitate adding and editing data in Storage Objects without using APIs. • Updated Java Adapter and Java Preprocessor and Postprocessor to add the JVM time zone restriction. • Updated the Custom Code Invoking topic to add

Date	Document Version	Description of Releases and Updates
02/25/2019	1.6	**Updated for Kony Fabric V8 SP4 FixPack**
02/04/2019	1.5	**Updated for Kony Fabric V8 SP4** Console and Runtime Integration Server • Added support for Mock JSON as a new adapter type for integration services • Added support for Audit logs for Fabric User Activity • Added support for SAP JCO adapter as a new adapter for integration services • Added support for pre-processor for Orchestration Services • Added ability to test Orchestration Services • Added Orchestration services support for Concurrent Looping • Added ability to add sample data for Storage Service • Added support for reconfiguring schema name in Relational Database connector • Added ability to pin your favorite Apps for Quick Access • Added ability to export list of users • Added support for import and export of custom reports and custom dashboards through MFCLI.

Date	Document Version	Description of Releases and Updates
02/04/2019	1.5	Kony Enterprise App Store • Added support for simplified Enterprise App Store (EAS) Service built directly into Kony Fabric runtime server; Users can access EAS via mobile/tablet device web browser; Supports to use built-in user repository or connect to existing enterprise user repository
		Identity • Enhanced support for identity reconfiguration to allow reusing same identity service with configuration changes across different clouds • Enhanced built-in user store by adding new feature for access levels • Support for Forgot Password for User Repository Identity Service • Enhanced custom identity service by providing a hook that would be invoked after identity session creation
		Kony SDK • Added support for ability to add field level metadata in Objects in object service. The metadata can then be accessed from client app from object model via custom functions registered with generated models to run various custom logic like encryption, conversion

Table of Contents

1. Preface

Kony Fabric is a Mobile Back-end as a Service (MBaaS) provider that helps developers build native and web apps for mobile. Various back-end services are easily integrated with the application irrespective of whether the application is built using JavaScript, PhoneGap, iOS, or Android frameworks.

Kony Fabric allows you to define the back-end to build native mobile apps for iOS, Android, and HTML5-based apps for modern browsers. Kony Fabric ensures that developers build mobile applications quickly by focusing on core areas and obtaining secured back-end services instantly. Kony Fabric has multiple features that can be used - Identity, Integration, Orchestration, Objects, Logic, and Engagement Services. These features can be accessed through a common, centralized console.

For successful authentication with users, and to access the centralized features of Kony Fabric, Kony recommends that you install the following Kony Fabric features:

- Kony Fabric Identity and Console
- API Developer Portal
- Kony Fabric Integration
- Kony Fabric Engagement Services

Kony Fabric supports the following back-end services for your applications:

- **Identity**: This feature allows you to define the type of authentication used for granting access to your application. Kony Fabric supports the following authentication services:
    - Enterprise Identity: Microsoft Active Directory, Open LDAP, Salesforce, Security Assertion Markup Language (SAML), Kony SAP Gateway, OAuth 2.0, Okta, Custom, and OAuth Provider.
    - Social Identity: Google, Instagram, Microsoft, BOX, Facebook, LinkedIn, Amazon, and Yahoo.

- **API Developer Portal**: This feature allows you to directly access your Kony Developer Portal in Kony Fabric Console.
- **Integration**: This feature allows you to define various back-end services for your application. You can define the following integration services:
    - Technology Adapters: XML, SOAP, JSON, Java, JavaScript, and APIProxy.
    - Business Adapters: Kony SAP Gateway, MuleSoft, AWSAPIGateway, Relational Database, MongoDB, RAML, OpenAPI (Swagger), Salesforce, and IBM MQ.
- **Orchestration**: Service orchestration is the coordination or integration of several services and exposing them as a single service. This feature allows you to create two types of orchestration services. They are:
    - Composite: Allows you to run two or more services concurrently or sequentially.
    - Looping: Allows you to run a single service in a loop until the loop ends or an exit criteria is met.
- **Objects**: Allows you to create app models for LOB objects, storage objects, and Service-Driven Objects.
- **Offline sync**: This feature allows you to define the synchronization services for your application. Sync supports only Web Services, except SAP Sky.
- **Logic**: The logic services feature in Kony Fabric helps you import and integrate Node.js services (APIs) directly into Kony Fabric for developing server-side and networking applications.
- **Engagement**: This feature allows you to send push notifications, email, SMS and passes to subscribed applications.

> ***Note:*** For more details, take a look at our hands-on tutorial for An Overview of Kony Fabric and Kony Cloud.

1.1 Product Compatibility Chart

Kony Fabric and Visualizer both support each other for the current release version and one previous version.

Product Version	Compatible With
Kony Fabric N (For example V8)	Visualizer N (For example V8) Visualizer N-1 (For example, V8 and 7.3)
Visualizer N (For example V8)	Kony Fabric N (For example V8) Kony Fabric N-1 (For example, V8 and 7.3)

1.2 Purpose

The document helps you familiarize with the Kony Fabric and provide procedural information to perform various tasks required to build your application.

1.3 Intended Audience

This document is intended for developers who would like to turn their applications into enterprise-grade applications using Kony back-end services.

1.4 Formatting Conventions Used in This Guide

The following formatting conventions are used throughout the document:

Conventions	Explanation
`Monospace`	• User input text, system prompts, and responses • File path • Commands • Program code • File names
Italic	• Emphasis • Names of books and documents • New terminology
Bold	• Windows • Menus • Buttons • Icons • Fields • Tabs • Folders
URL	Active link to a URL.
Note:	Provides helpful hints or additional information.
Important:	Highlights actions or information that might cause problems to systems or data

1.5 Related Documents

Document	Purpose
Kony Fabric Installation Guide Windows	This document explains how to install Kony Fabric and additional software on your Windows computer.
Kony Fabric Installation Guide Linux	This document explains how to install Kony Fabric and additional software on your Linux.

1.6 Contact Us

We welcome your feedback on our documentation. Write to us at techpubs@kony.com. For technical questions, suggestions, and comments, or to report problems on Kony's product line, contact support@kony.com.

2. Accessing Kony Fabric Console - Cloud

Before you use various Kony Fabric services, you must register with Kony.

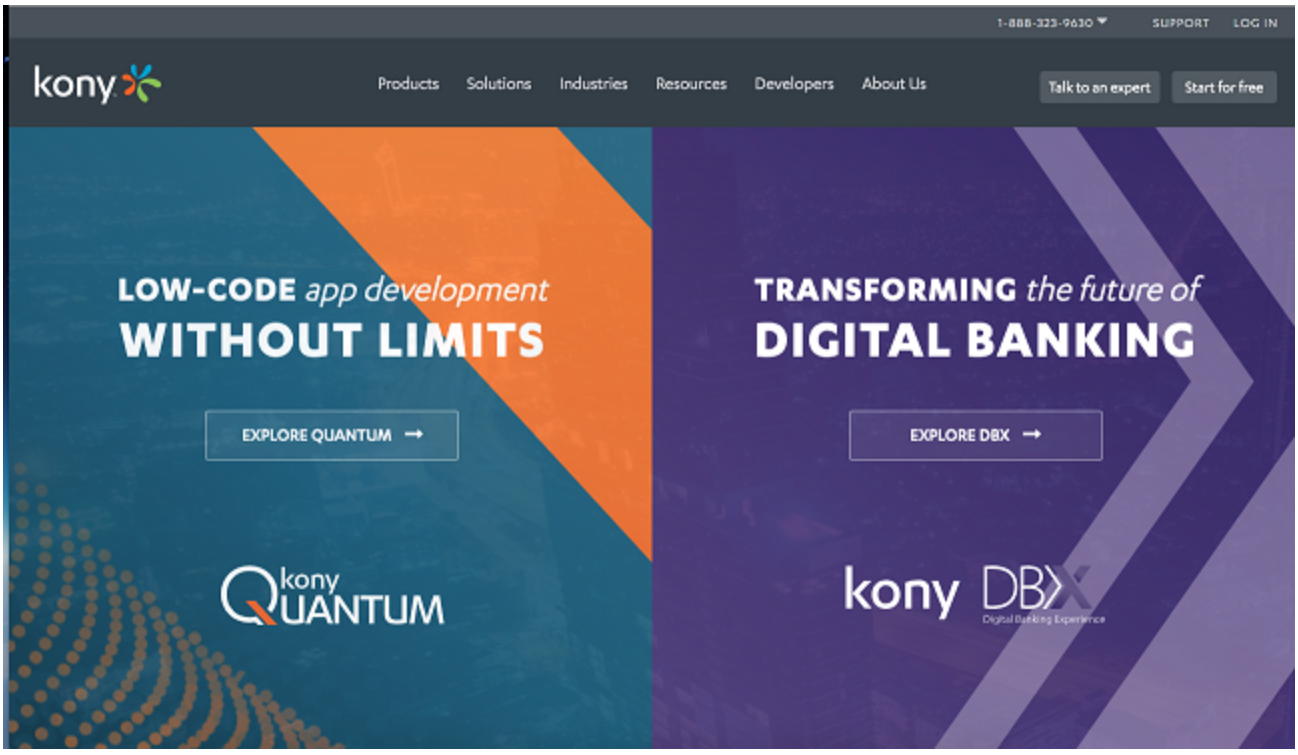
To access Kony Fabric, follow these steps:

1. Create a Kony Fabric Account
2. Access Kony Fabric Portal

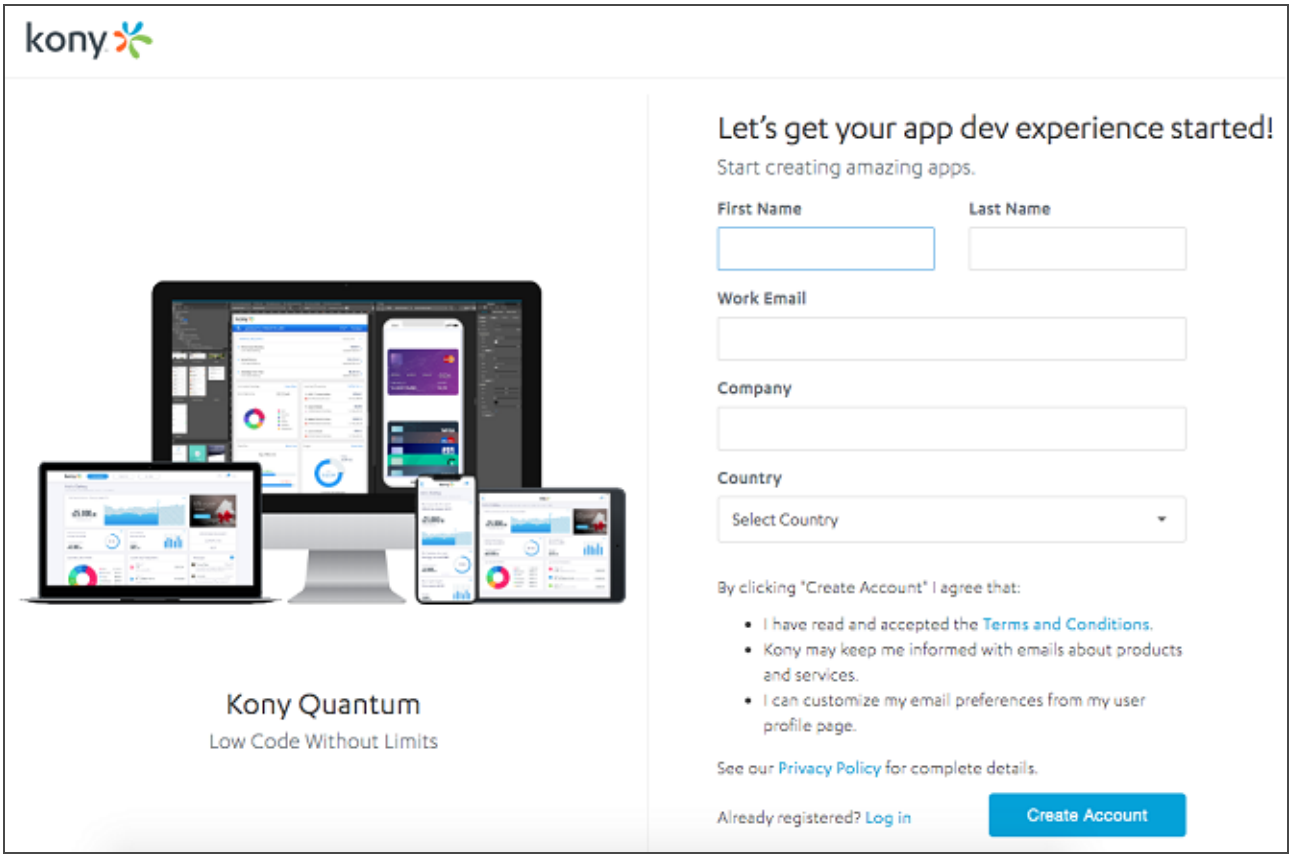
2.1 Create a Kony Fabric Account

If you do not have a Kony account, follow these steps to create a Kony Fabric account:

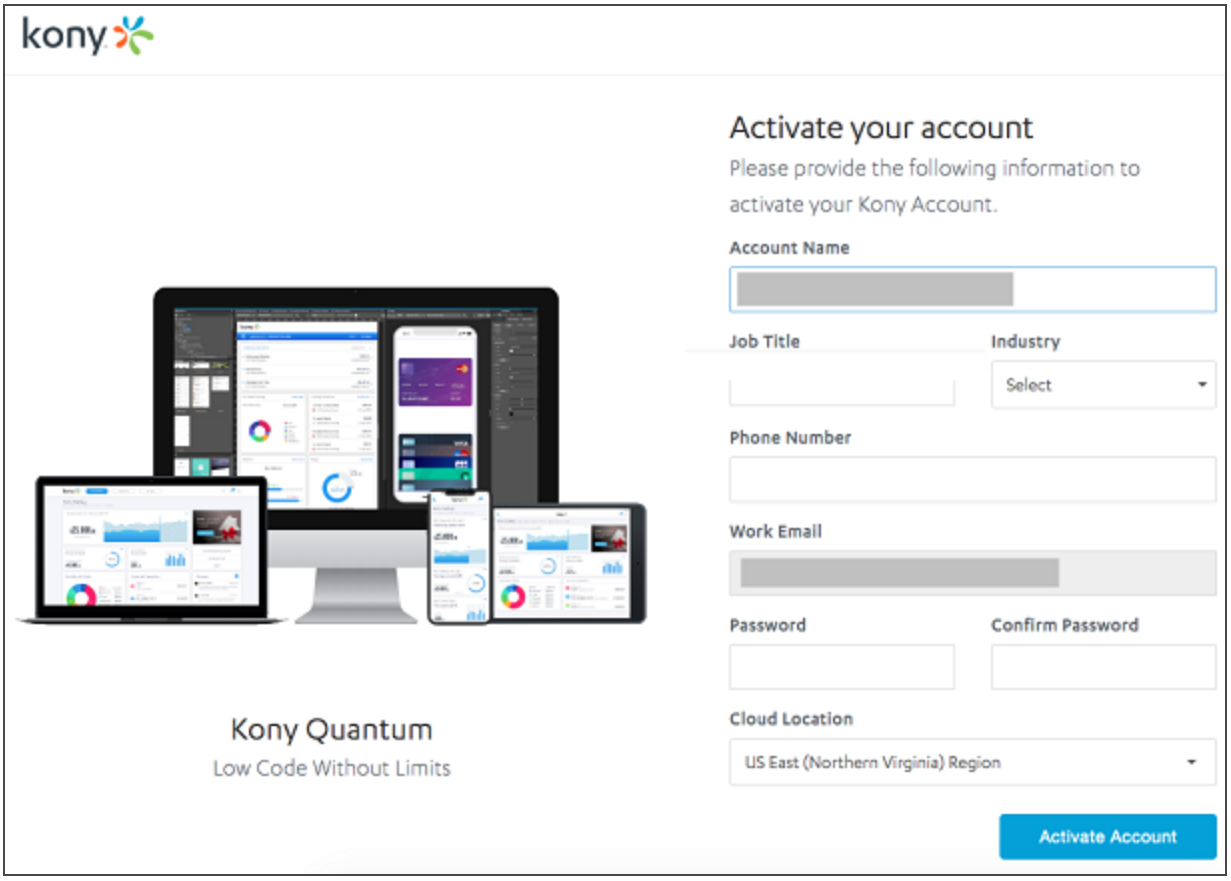
1. Go to kony.com, and then click **Start for free** from the top menu.



The **Let's get your app dev experience started!** page is displayed.

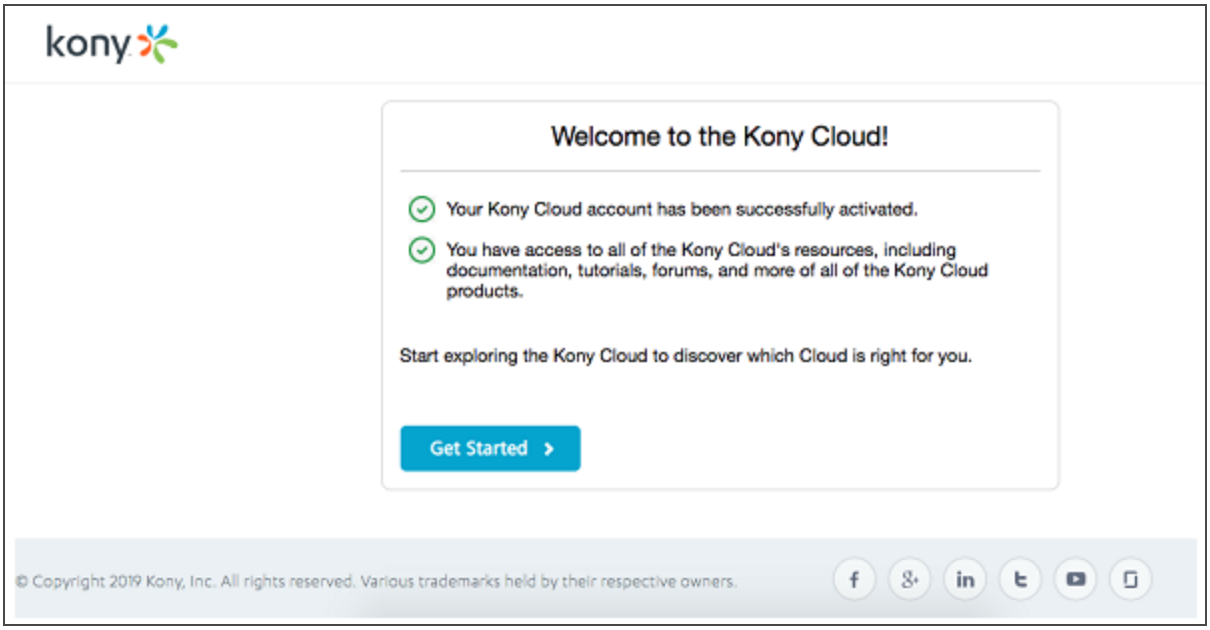


2. Provide the required details.
3. Click **Create Account**.
4. An email is sent to your registered email ID an activation link. Click the link, or copy the link on the browser. The **Activate Your Account** page appears.



5. Complete the required details in the activation page to create a Kony Fabric cloud account.
6. Click **Activate Account**.

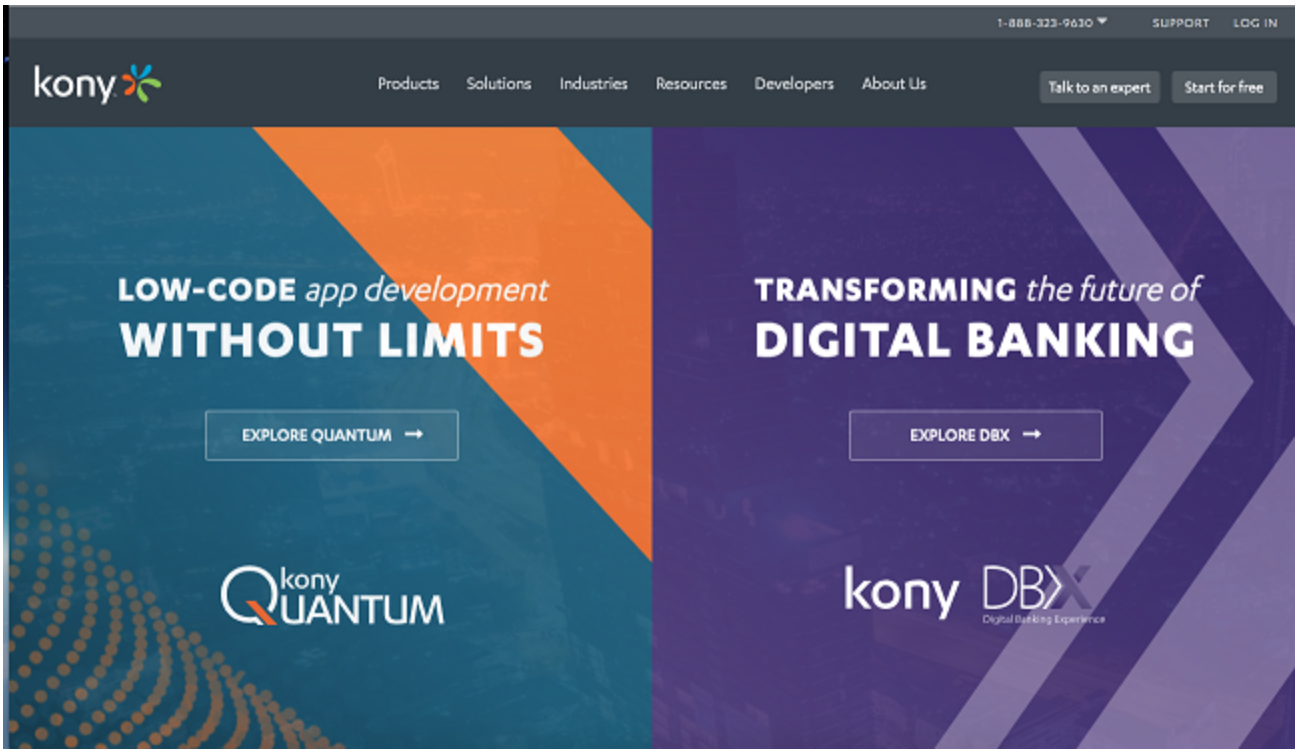
   Welcome to the Kony Cloud! page appears.



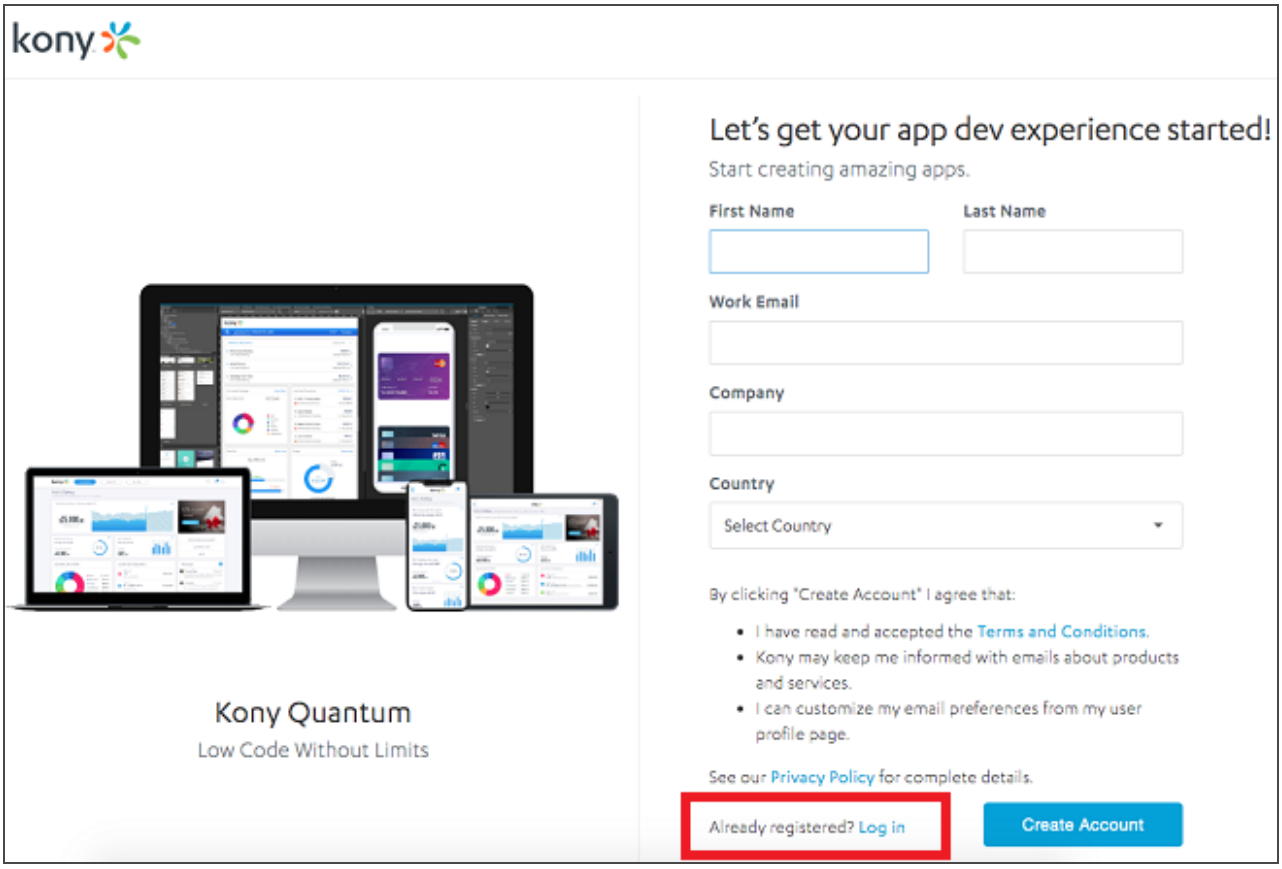
7. Click **Get Started**.
8. Enter details in the **Sign in to your Account** page.
9. Click **SIGN IN**.

If you have a Kony Account, follow these steps for creating a Kony Fabric cloud account:

1. Go to kony.com, and then click **Start for free** from the top menu.



The **Let's get your app dev experience started!** page is displayed.



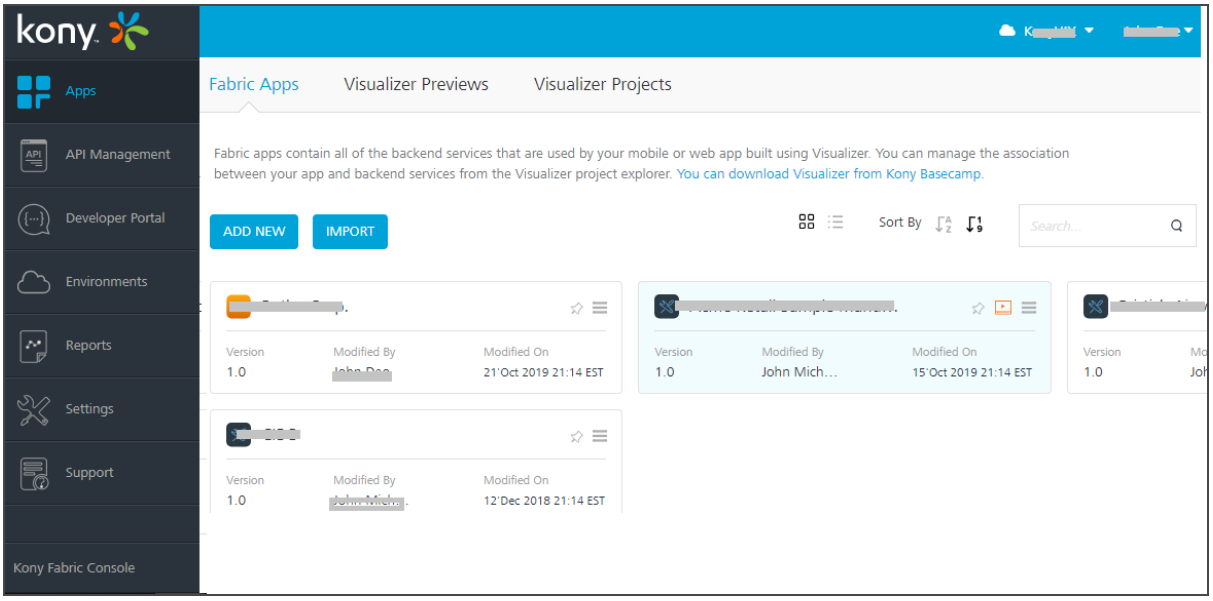
2. Click on **Already registered? Log in.**
3. Enter details in the **Sign in to your Account** page.
4. Click **SIGN IN**.

2.2 Access Kony Fabric Portal

To access your cloud account, follow these steps:

1. Go to https://manage.kony.com.
2. Provide your Kony account login credentials, and then click **Sign in**.

3. After validating your credentials, you are directed to your Kony Fabric account (Kony Cloud).



From Kony Fabric Console, you can navigate to the following:

- **Apps**: For more information on **Applications**, refer to Adding Applications.
    - The **Visualizer Previews** page lists the test live previews that you performed in a particular Cloud account. Kony Visualizer supports the **Run Live Preview** option that you can use to preview a prototype of your Visualizer application.
      For more information on How to user Live Preview in Kony Visualizer, refer Live Preview
    - The **Visualizer Projects** page lists the projects that you published to a particular Cloud. The Project tab in Kony Visualizer contains the **Export > Cloud Project** option.
      For more information on How to share a project on the Cloud, refer Publish your project to the cloud
- API Management: Configure and manage (create, edit, and delete) app services (identity, integration, and orchestration) without linking or configuring them within an app.

- "Kony Developer Portal" on page 1003: Allows you to create a Portal for exposing APIs created using Kony Fabric. Developers from internal and external partner teams can access the portal created to explore and test the APIs.
- **Clouds (Environments)**: For more information on Cloud/Environments (Kony Cloud), refer to Environments - Kony Cloud.
- **Reports**: For more information on **Reports**, refer to http://docs.kony.com/konylibrary/konyfabric/custom_metrics_and_reports/default.htm.
- **Settings**: Allows you to invite users associated with account roles such as admin, billing, and member.
- **Support**: Displays links to the latest tutorials and articles and Developer resources from Kony Base Camp Library. Click here for more information.

> ***Note:*** The release version of a console is displayed at the bottom left corner in the console menu pane. The release version is in the following format:
> `<Major_version> <servicepack> <hotfix> <DEV/QA>`
> For example: `V8 SP1 HF4 DEV`

2.3 Accessing Kony Fabric Standard

Kony introduces a Kony Fabric Standard edition which is a subset of Kony Fabric runtimes made available at a lower cost.

To access Kony Fabric Standard, follow these steps:

1. You need to contact the Kony sales representative to enable a plan.
   - Kony sales representative discusses regarding the suitable plan and the plan information will be passed to the Kony Accounts team.
   - Accounts team will create the cloud plan JSON with free SKU (i.e. access to Kony Fabric Workspace) and links the same to the customer's account.

2. On selecting the Kony Fabric Standard edition, customers can access the following:
    - Accessing Kony Fabric Workspace/Console and
    - Kony Fabric Standard

2.3.1 Kony Fabric Workspace/Console

Kony Fabric Workspace/Console is available as a part of Kony Fabric Standard edition but no environments are available. Links to use the trial version or to purchase the SKU are displayed. You can view, create apps and use all the features of Kony Fabric Console except the features that involve runtime.

2.3.2 Kony Fabric Standard

The standard SKU of Kony Fabric Standard excludes the following features:

- Integration (Kony LOB Adapters are not available i.e Salesforce and SAP adapter)
- Sync
- Engagement
- Analytics (custom)
- App Management
- IoT Services

Following features are available to access in Kony Fabric Standard:

- Identity (All Providers)
- Integration (Tech Adapters only)
- Orchestration
- Object Services
- API Management

- Analytics (Standard Reports only)
- Node.js runtime

> ***Note:*** Be default, a customer having access to Standard SKU cannot view the custom reports. If a customer has the provision to access two cloud environments, one with the Enterprise SKU and the other with the standard SKU, they can view/access the custom reports.

2.3.3 Error Messages

If a customer tries to access the features that are not available, following error messages are displayed,

- If you try to access the features which are not available, the following message is displayed during the publish time "**Your application contains sky services that are not supported by the type of environment you have selected. Please select a different environment or contact support**".
- If the server is not provisioned for the feature accessed, the following error message is displayed "**Server Connection failed**".

3. Environments - Kony Cloud

By default, at least one Cloud/Environment is configured for your Kony Cloud account. Environments can include at least one server or a combination of all servers, such as Kony Fabric Integration, Kony Fabric Engagement, Kony Fabric Sync, and Kony Fabric Management. You can view your clouds in the **Environments** page in Kony Fabric console for Cloud.

For more details on Kony AppFactory such as how to configure your first app using your source code repository, build and deploy apps, deliver apps, refer Kony AppFactory User Guide.

> ***Important:*** Ensure that your environments include all required servers that are part of an app.
>
> For example, if your environment contains only Kony Fabric Sync, and you try to publish an app with Kony Fabric Engagement, the system throws an error.

From the **Environments** (Kony Cloud) page, you can navigate to the existing Clouds in your account:

- **Clouds**: Following consoles are available for each of the cloud account:
  - **App Services**: For more information, refer Appendix - App Services.
  - **Log Services**: For more information, refer Log Services.
  - **Kony Fabric Sync**: For more information, refer http://docs.kony.com/konylibrary/sync/kony_sync_console_user_guide/Default.htm.
  - **Kony Fabric Messaging**: For more information, refer http://docs.kony.com/konylibrary/messaging/kms_console_user_guide/Default.htm.

> ***Note:*** You can view all applications published to a Runtime Environment. Refer Applications Published to a Runtime Environment

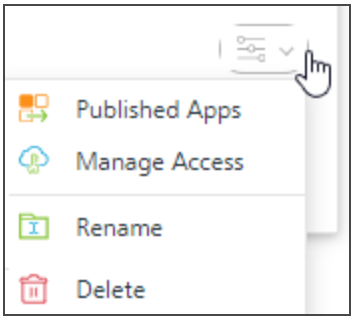
3.1 Managing Cloud Environments

You can manage user access to the existing clouds in the **Environments** page in Fabric console such as viewing published apps to a cloud, managing features of a cloud, renaming a cloud, and deleting a cloud.

1. From the left pane in your Kony Fabric Console, click **Environments**. This displays the list of environments configured for your Kony Fabric Cloud account.

2. Click the **More Options** button of an environment.

   > ***Note:*** The **More Options** button in the **Environments** page is available only if you have the Admin access.

3. Perform the following actions for a cloud environment:

Published Apps	Helps you to view the list of apps published to an environment and EAS and the runtime status of the apps. Click here for more information.
Manage Access	Helps you to manage environment access for all the existing users in a particular account. Click here for more information.

Rename	Helps you to rename this cloud environment.
Delete	Helps you to delete this cloud environment.

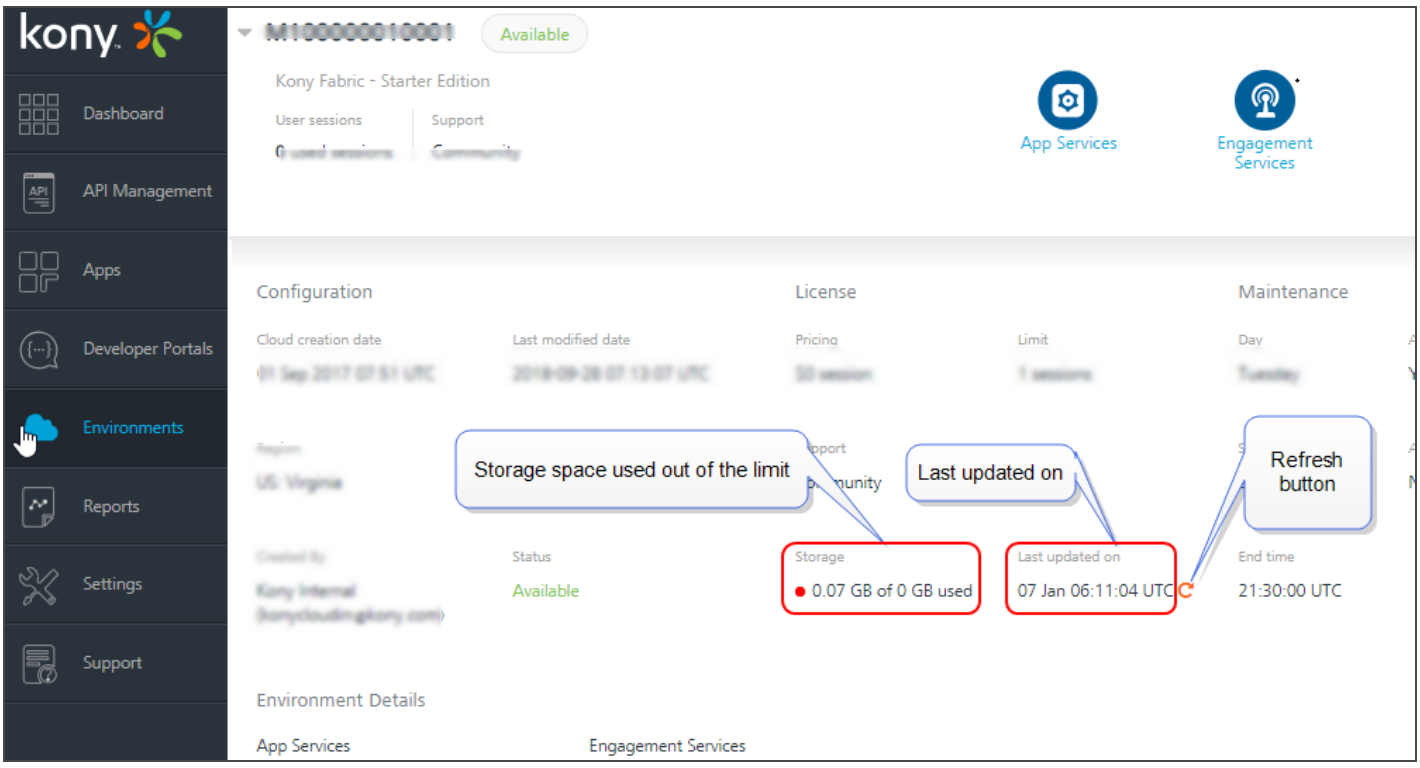
3.2 Limiting the Size of Storage Services - Environments in Cloud for Starter Edition

The **Limiting space for storage service** feature enables to limit the total storage used at a tenant level (Multi-tenant) for users in Cloud environments for Kony Apps Server. This feature is enabled for Kony Fabric Starter Edition.

By default, all Cloud users in Kony Starter Edition are provisioned with 1 GB size limit for storage services on App Servers. So, users can use the default storage in the run-time server to deploy the resources such as Web Apps, Native Apps, Storage Objects, and Runtime Services (for custom JARS and data in DB)

The **Environments** page displays the users' usage of storage resources for app server in the Kony Fabric, as follows:

- **Storage** displays the amount of storage used and the allowed total storage limit for that environment.
- **Last updated on** displays the last time when it was measured.
- A user can check the latest status of the storage occupied and limit by clicking the **Refresh** button.



The following are the notification messages for space for storage services:

- When the storage space occupied by a tenant reaches 90%, a warning notification mail will be sent.
- When the storage space occupied by a tenant reaches 100%, an error notification mail will be sent.
- Once the space occupied by a tenant reaches 100%, apps publish will be blocked and an error message will be sent. All create/update calls for storage services will also be blocked for the tenant.
- If the storage space had reached 90% or 100%, then space is cleared up later, and then increased it again to reach 90% or 100%, corresponding notification mail will be sent again.

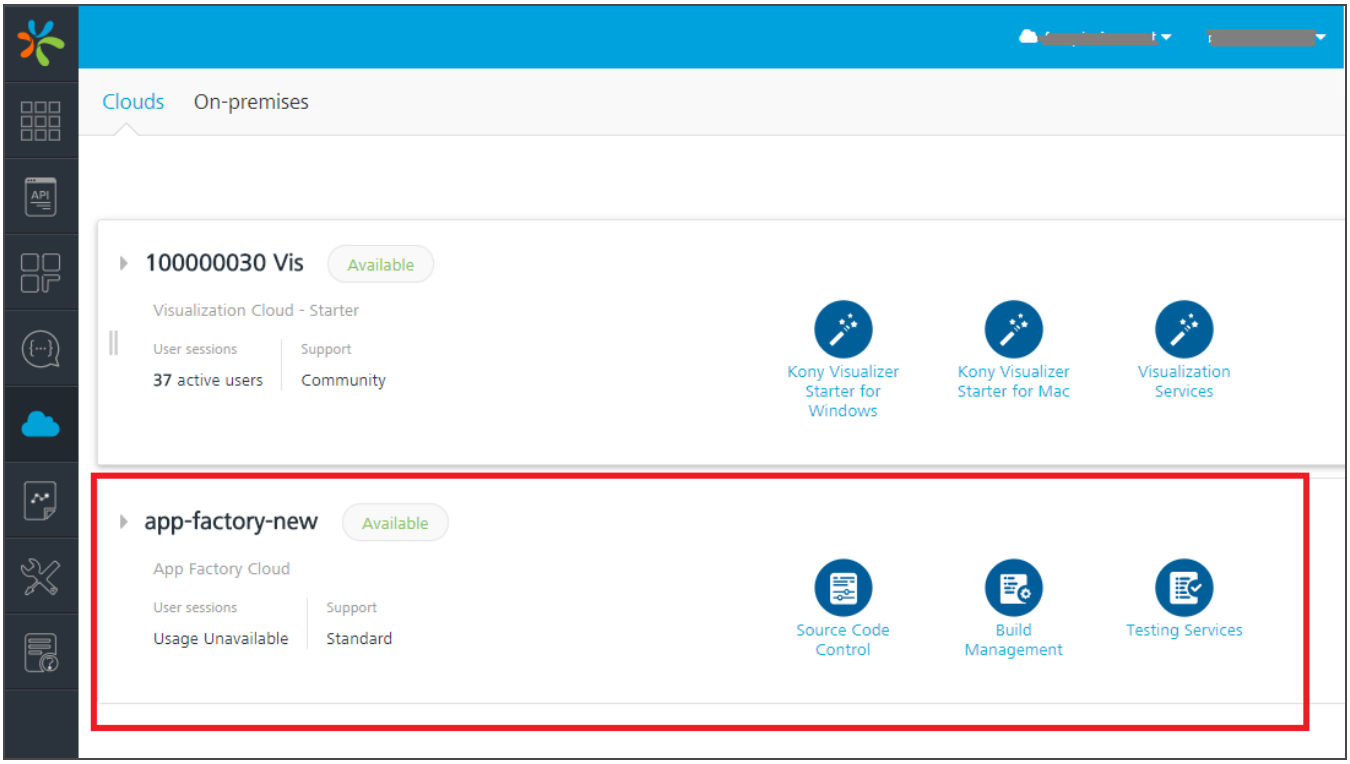
> ***Note:*** The storage limit can be increased based on user's request in case the user wants to upgrade his resources for Starter Edition. For example, A user buys SMB edition and wants to increase the limit of the storage DB, he can request Kony for upgrading storage size limit.

3.3 Kony AppFactory Cloud

The Kony AppFactory is a combination of agile methodologies and tools that allow teams to deliver apps at scale. Kony AppFactory® allows building omni-channels mobile applications with a few clicks. The AppFactory runtime automates SDLC process of your product lifecycle and reduces Time to Market by cost effective automatic solution of product release management. Finally, deploy the application to a specific environment and test the application in that environment using automated testing on real devices.

Using Kony AppFactory Cloud, you can view the list of app builds, configured apps, and test results corresponding to the apps. You can enable Kony AppFactory Cloud in Kony Cloud account. For more details on how to enable Kony AppFactory, contact support@kony.com.

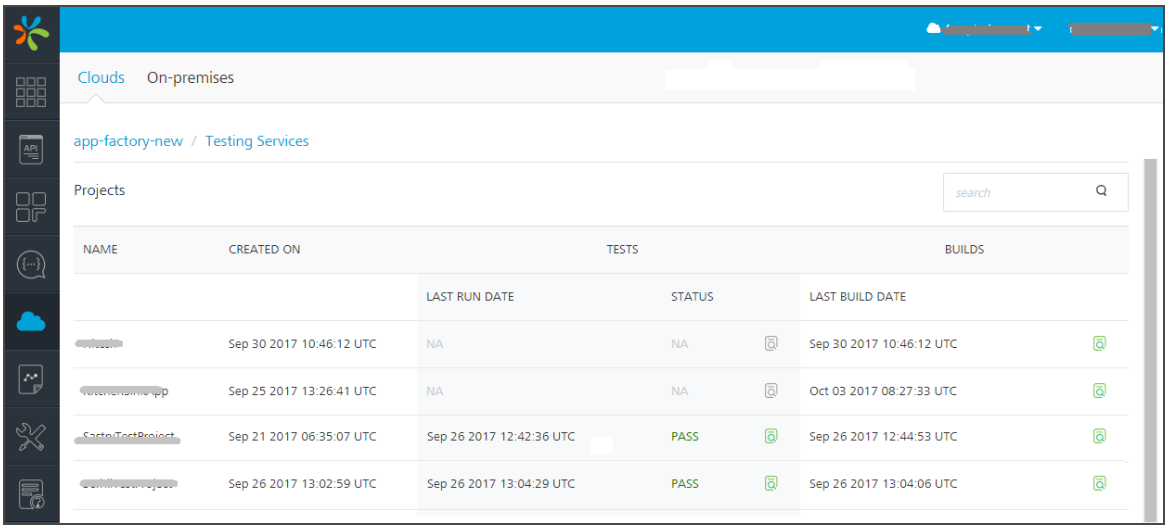
Once Kony AppFactory Cloud is enabled, you can view the Kony AppFactory Cloud along with other clouds in the **Environments** page in Kony Fabric Console.



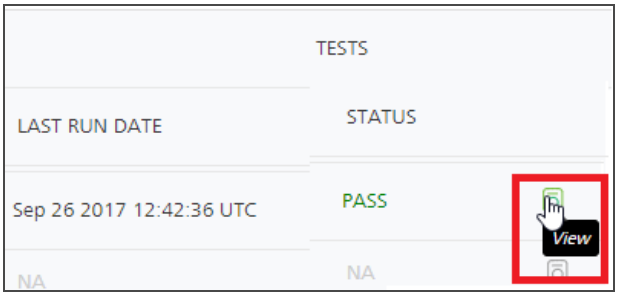
You can perform the following tasks through Kony AppFactory Cloud in Kony Fabric Console:

- Click **Source Code Control** to navigate to your GitHub page for code check-ins and develop apps.
- Click **Build Management** to navigate to your source automation server (for example, Jenkins) to automate the software development process.
- Click **Test Services** to view results of build execution and test execution.

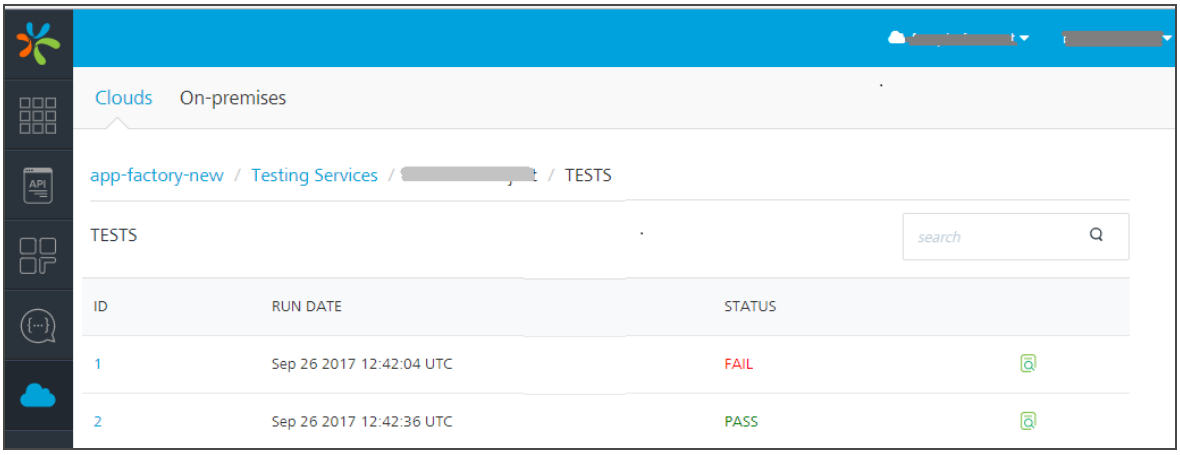
  The **Testing Services** pages detail app names, test project creation date, test results and statuses, and build statuses of your apps.



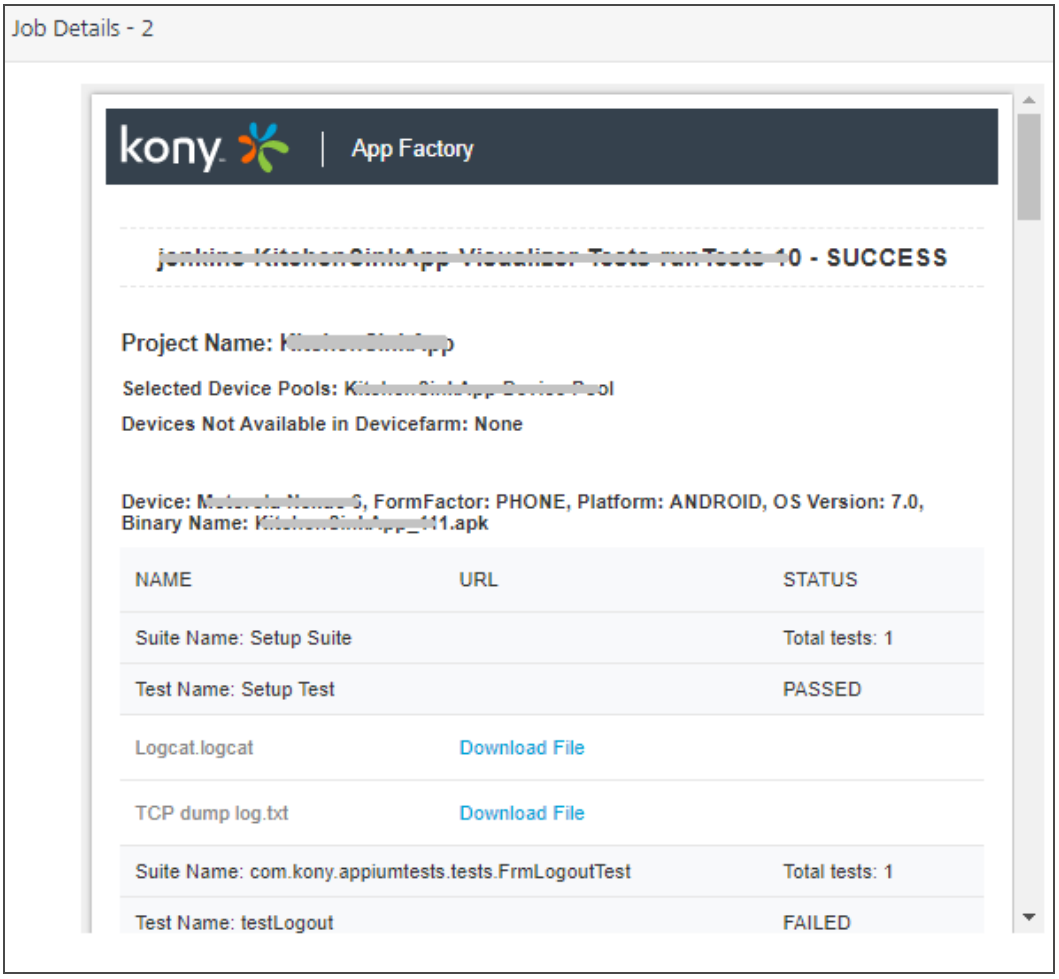
  - The **TESTS** page displays a list of test results for a project/app. For viewing the details, you can click **View** button.



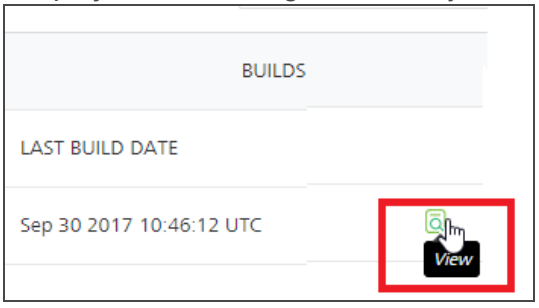
    Test results will retrieve stored assets, videos, and logs.
    Click **View**. The following screen appears with details:



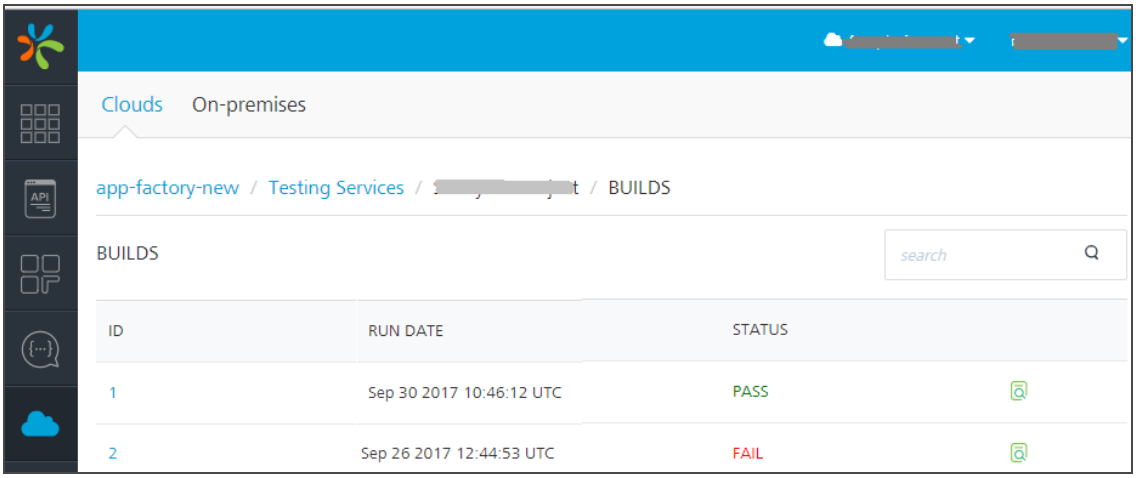
- To view the test results of an app, click **View** in the **TESTS** screen. The following screen appears with sample details:



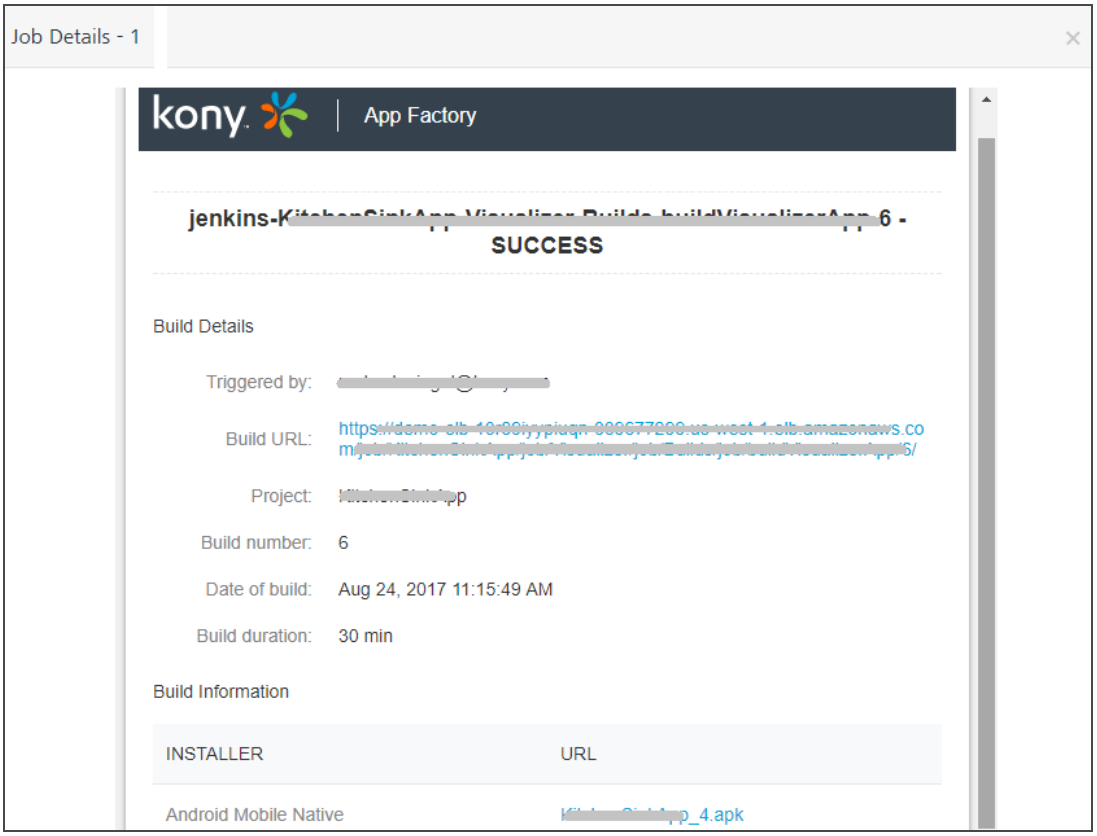
- The **BUILDS** page displays a list of builds triggered and build status for selected channels in a project. For viewing the details, you can click the **View** button.



Click **View**. The following screen appears with details:



- To view the results of a build or app, click **View** in the **BUILDs** screen. The following screen appears with sample details:



4. Account Settings for Default User Access Control Lists (ACLs) on New Apps and Services

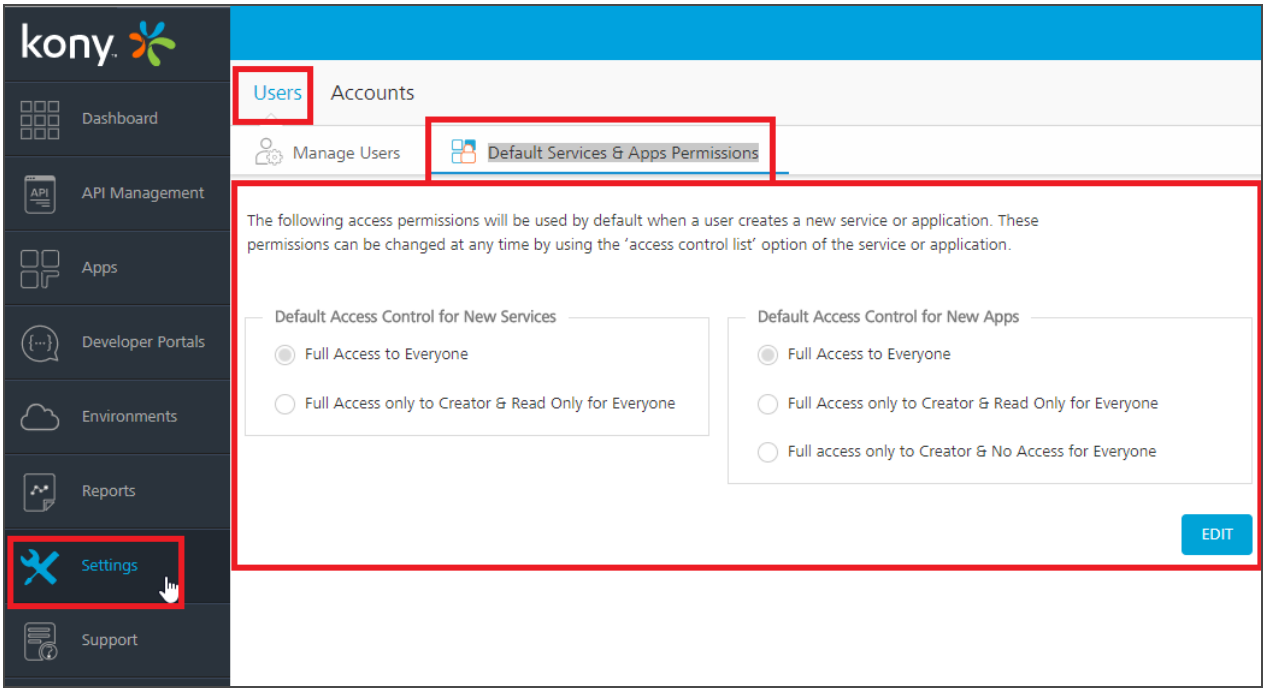
As an Admin/Owner account role, you can configure the default access control for the new apps and services. So, when a new user is invited or automatically added to an account using the master account setting, these default access control settings are applied to the user.

By default, the following access permissions are used when a user creates a new service or application.

Default Access Control for New Services	Default Access Control for New Apps
• Full Access to Everyone • Full Access only to Creator & Read Only for Everyone	• Full Access to Everyone • Full Access only to Creator & Read Only for Everyone • Full access only to Creator & No Access for Everyone
You can edit the **Default Access Control for New Services** and **Default Access Control for New Apps** by clicking the **EDIT** button.	

> ***Important:*** When you edit the default access control permissions, the changes will apply to the apps and services created post change only and not the ones created before the edit.

You can access the **Default Services & Apps Permissions** under **Settings** > **Users** tab.



Note: Use the Apps Console Access Control page to control the access to an application.

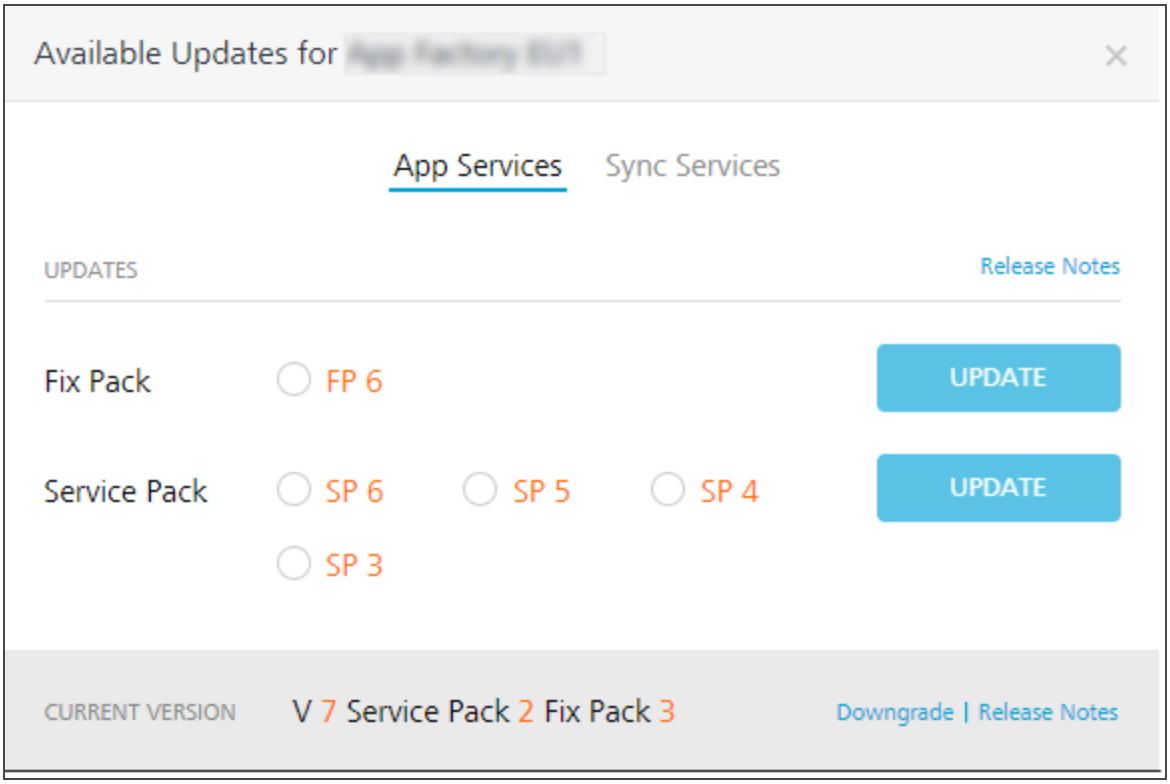
Note: Use the Services Console Access Control page to control the access to a service.

5. Kony Fabric Updates on Cloud

Kony is constantly making improvements to Kony Fabric, which you can install as updates to your PROD environments/Clouds. As an Admin, you can view the latest available Fix Pack version as well as all the Service Packs of the current Cloud's Major version.

5.0.1 How to Update a Fix Pack or Service Pack

1. From the left pane in your Kony Fabric Console, click **Environments**. By default, the **Clouds/Environments** tab is selected and displays the list of clouds or environments configured for the Kony Fabric account.
2. Click the **Settings** button of an environment.
3. From the **Settings** context menu, select the **Check for Updates**. The **Available Updates** dialog appears.



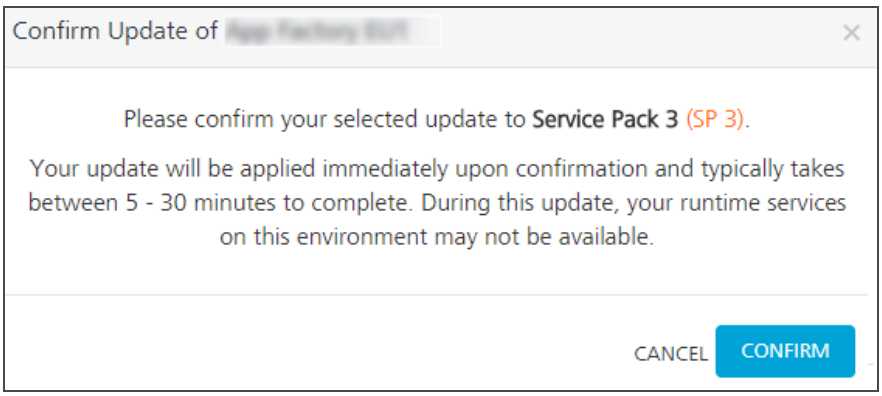
You can switch between the **App Services** and **Sync Services** tabs to view the available *Fix Pack* and *Service Pack* updates for the current environment version. Updates are displayed in the following format:

App Services	Sync Services
UPDATES	**UPDATES**
• Fix Pack version ***Note:*** Only the latest version of a Fix Pack is displayed.	• Fix Pack version ***Note:*** Only the latest version of a Fix Pack is displayed.
• Service Pack versions	• Service Pack versions
Current Cloud Version	

4. You can select the available latest fix pack version or a service pack version of app services or sync services and then click **UPDATE**.

   Note: You can select one update option at a time and update it.

   The **Confirm Update** dialog displays the following message and prompts for you to confirm the update.
   For example, *Your update will be applied immediately upon confirmation and typically takes between 5 - 30 minutes to complete. During this update, your runtime services on this environment may not be available.*



5. Click **CONFIRM**. The process of downloading the updates begins.

6. Accessing Kony Fabric Console - On-premises

Before you use various Kony Fabric services, you must create a superuser.

> *Important:* **Kony Visualizer and Kony Fabric Version Compatibility**
> Kony Visualizer and Kony Fabric support each other for the current release version and the immediate previous version. For example, with Kony Visualizer V8, Kony Fabric V8 and V7.3 will be compatible and vice versa.

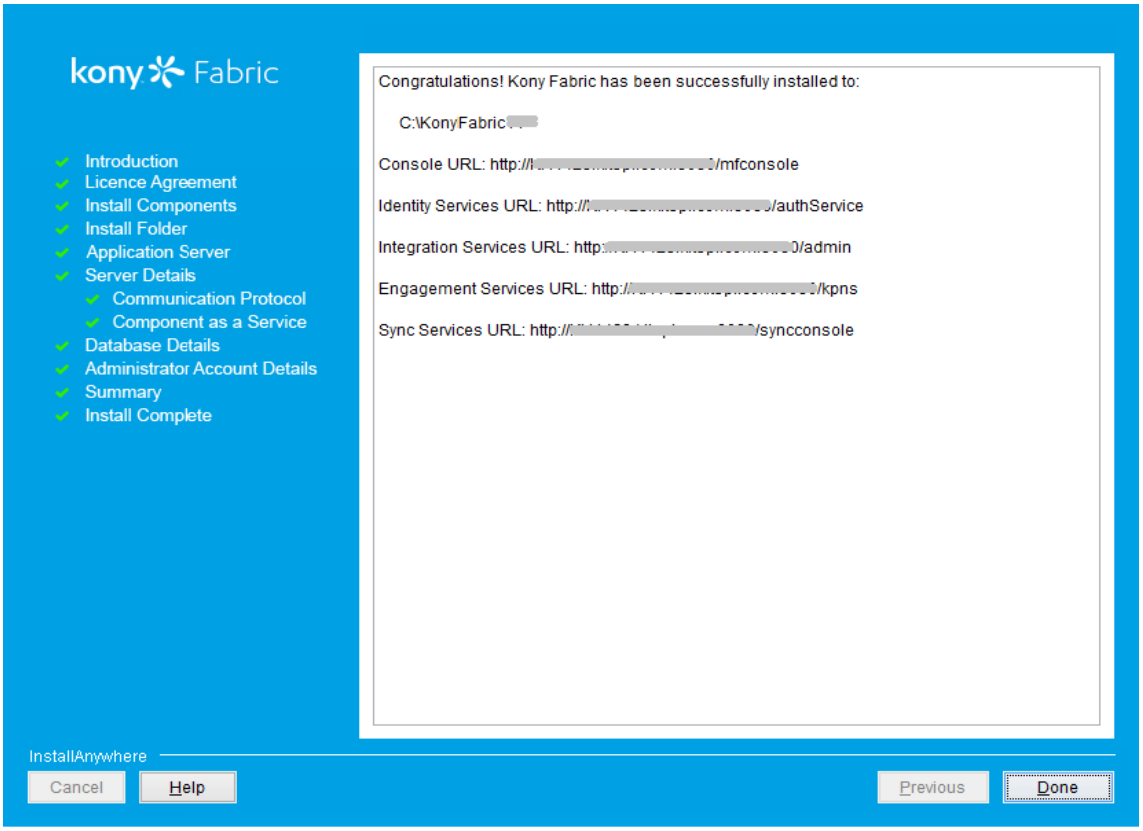
To access Kony Fabric, follow these steps:

1. How to Get Started With Kony Fabric Console
2. How to Log In to Kony Fabric Console

6.1 How to Get Started With Kony Fabric Console

> *Note:* If you have installed Console and Identity Service along with one or more Kony Fabric components such as **Integration**, **Engagement** and **Sync** services on **Tomcat** or **JBoss** on a **single node**, you can directly log in to Kony Fabric Console as you have already created your super administrator account.
>
> While installing Kony Fabric components with above combination, the **Administrator Account Configuration** window helps you to configure your super administrator account.

After Kony Fabric is installed, you need to configure identity services, and create your administrator account. Based on the installation, you will see the list of URLs in the **Install Complete** window shown below:

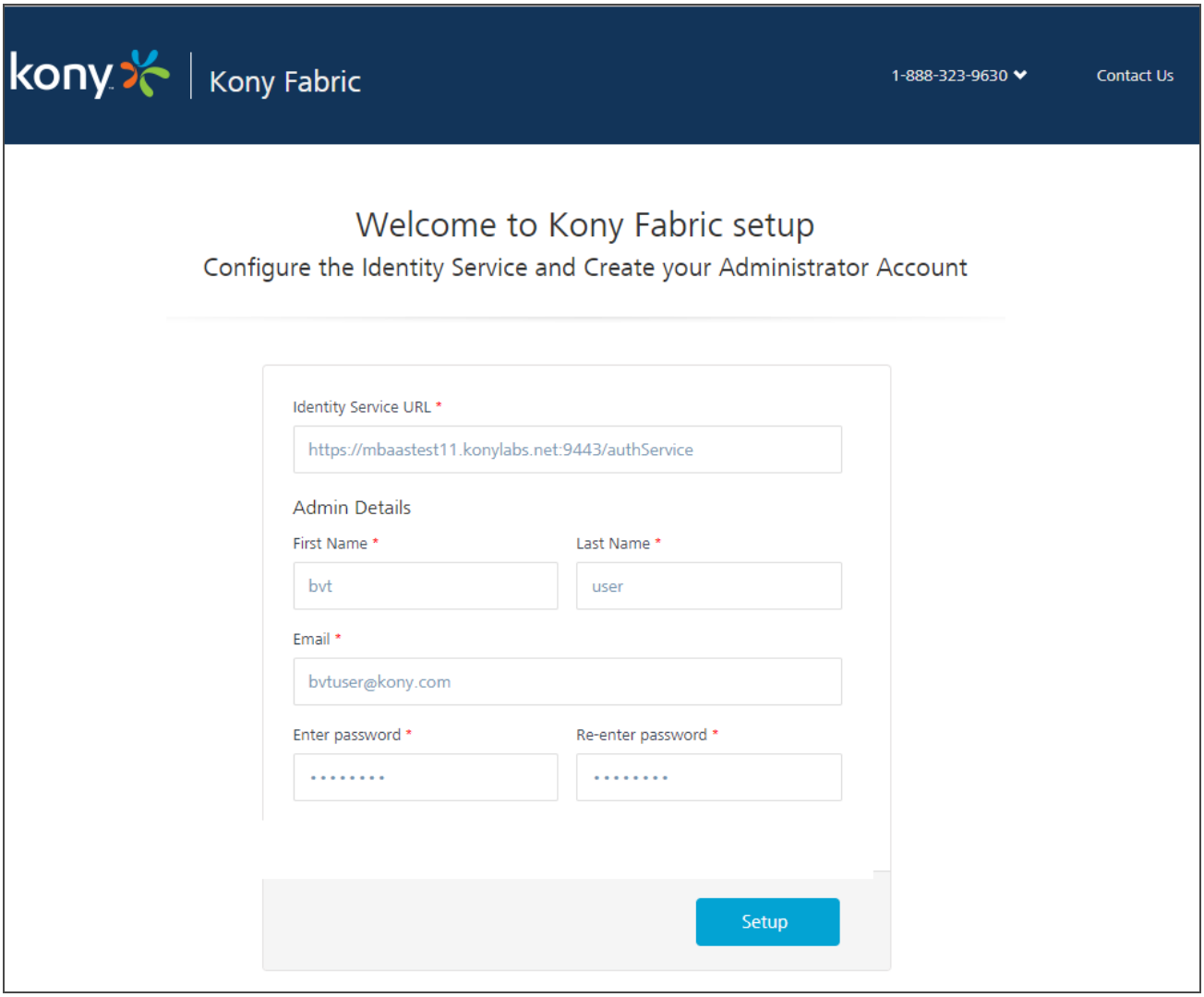


To launch Kony Fabric Console, follow these steps:

1. From the **Install Complete** window, copy the **Console URL**, and run it in your web browser.

   > ***Note:*** Bookmark the URL for quick access.

   The **Welcome to Kony Fabric setup!** page appears if you have not configured your identity services.



> ***Note:*** Fields marked with an asterisk are mandatory.

2. In **Kony Identity Service URL** text box, enter Identity Service URL from the **Install Complete** page.
3. In the **Admin Details**, enter the following details:
   - **First Name**: Enter the first name of the user.
   - **Last Name**: Enter the last name of the user.

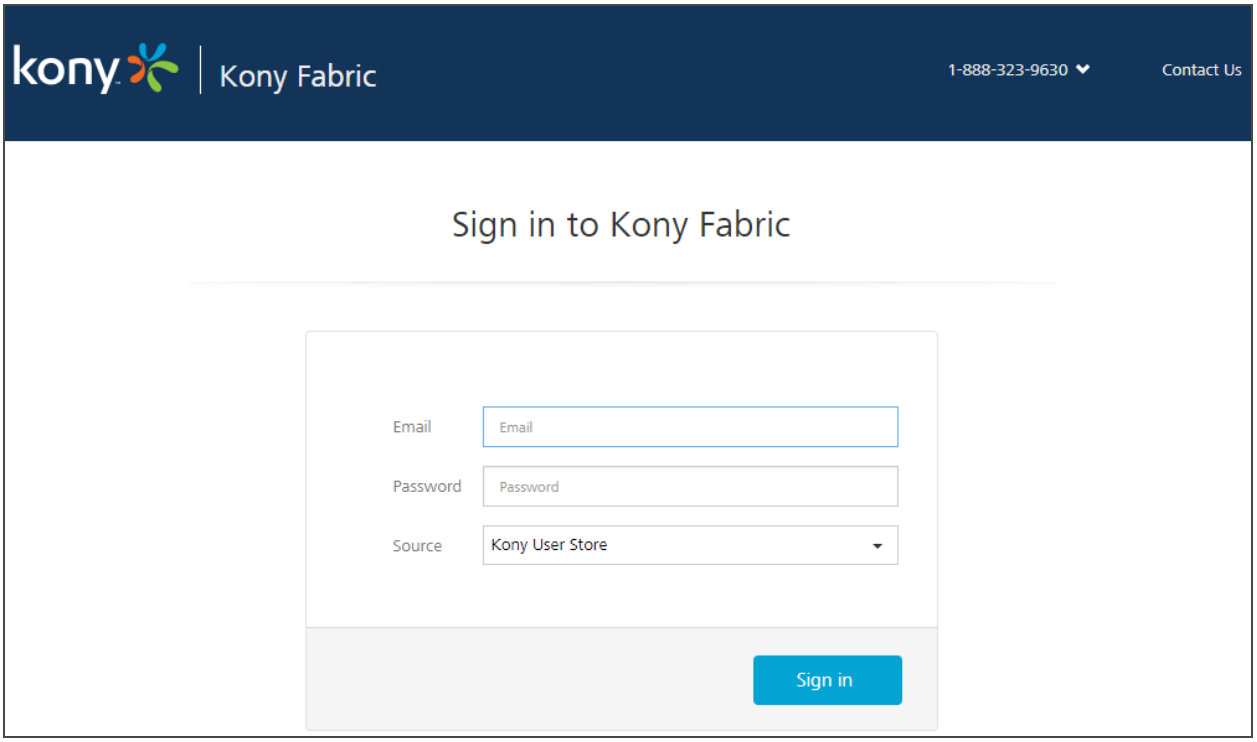
- **Email**: Enter the email address of the user. It can include alphanumeric and special characters that follow standard email address form.
- **Enter password**: Enter the password for the user. It can be a combination of alphanumeric and special characters.
- **Re-enter password**: Retype the password to ensure the user's identity.

4. Click **Setup**.
   Once the details are validated for one-time configuration, the system will:
   - Associate your credentials with Kony Fabric identity services and authorization services.
   - Display the **Sign in to your Kony Account** page.

6.2 How to Log In to Kony Fabric Console

If you have configured identity services and created your administrator account (Kony Fabric superuser account), you can log in to the Kony Fabric console. A superuser will have owner permissions by default.

1. Go to **Kony Fabric Console URL** that you have bookmarked in the previous section. The **Sign in to Kony Fabric** page appears.



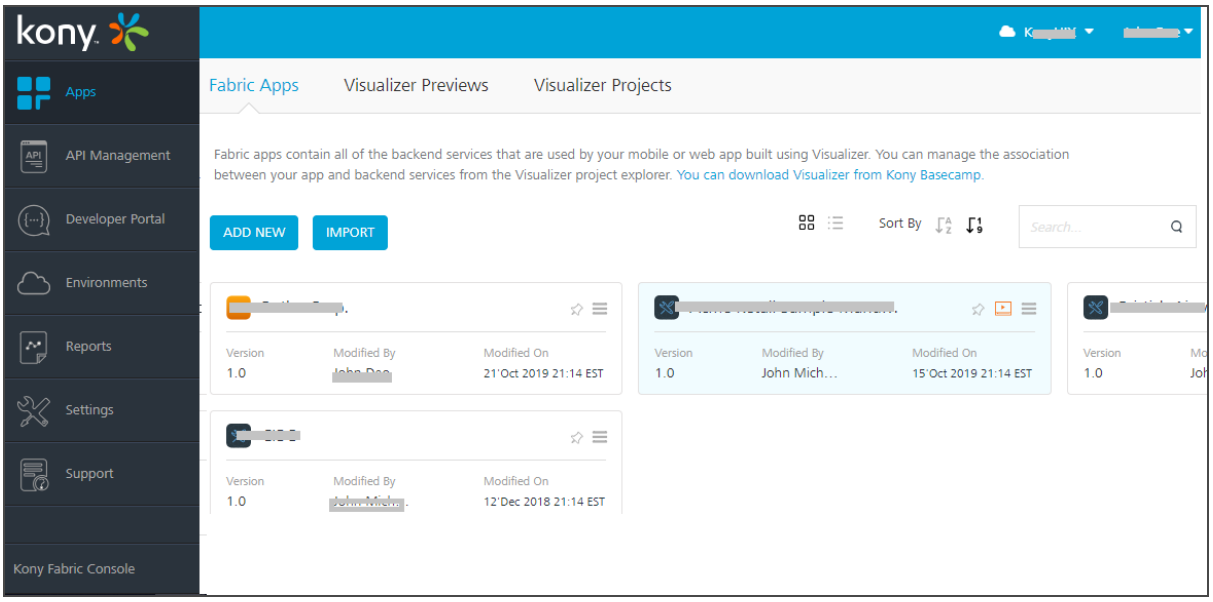
2. Provide your Kony administrator account login credentials (email and password) that you have created.

3. From the **Source** drop-down list, choose the source type. By default, the **Source** lists the Kony User Store.
   The Source lists configured active directories only if you have configured active directories in the Settings > User Management > Identity Providers.

   The **Domain** drop-down list appears only if you choose Active Directory as the source.

   When a username is common across multiple sources and multiple domains in Active Directory, a user is asked to provide the source and domain details for authentication. Because there are users from multiple sources, both the Source and the Domain should be differentiated. A user must provide both the Source and the Domain before authentication occurs.

4. From the **Domain** drop-down list, choose one of the domains of Active Directory.
5. Click **Sign in**.
   After your credentials are validated, you are directed to your Kony Fabric account. By default, the **Apps** page appears.



> ***Note:*** The release version of a console is displayed at the bottom left corner in the console menu pane. The release version is in the following format:
> `<Major_version> <servicepack> <hotfix> <DEV/QA>`.
> For example: `V8 SP1 HF4 DEV`

From Kony Fabric Console, you can navigate to the following:

- **Apps**: For more information on **Applications**, refer to Adding Applications.
  - The **Visualizer Previews** page lists the test live previews that you performed in a particular Cloud account. Kony Visualizer supports the Run Live Preview option that you can use to preview a prototype of your Visualizer application.
    For more information on How to user Live Preview in Kony Visualizer, refer Live Preview

    - The **Visualizer Projects** page lists the projects that you published to a particular Cloud. The Project tab in Kony Visualizer contains the Export > Cloud Project option.
      For more information on How to share a project on the Cloud, refer Publish your project to the cloud

- API Management: Configure and manage (create, edit, and delete) app services (identity, integration, and orchestration) without linking or configuring them within an app.
- "Kony Developer Portal" on page 1003: Allows you to create a Portal for exposing APIs created using Kony Fabric. Developers from internal and external partner teams can access the portal created to explore and test the APIs.
- **Environments**: For more information on **Environments**, refer to Environments.
- **Consoles**: The following consoles are available for each cloud account:
    - **App Services**: For more information, refer to Appendix - App Services.htm.
    - **Kony Fabric Sync**: For more information, refer to http://docs.kony.com/konylibrary/sync/kony_sync_console_user_guide/Default.htm.
    - **Kony Fabric Engagement**: For more information, refer to http://docs.kony.com/konylibrary/messaging/kms_console_user_guide/Default.htm.
- For more information on **Reports**, refer to Kony Reporting and Analytics - Standard Metrics and Reports.

  Refer to http://docs.kony.com/konylibrary/konyfabric/custom_metrics_and_reports/default.htm
- For more information on **Settings**, refer to Settings.

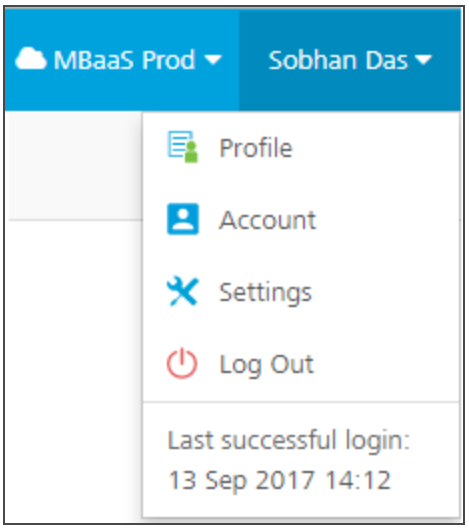
6.3 How to View Your Login History

You can view the history of your logins to Kony Fabric. The history includes both successful and unsuccessful login attempts. The account options menu shows your last successful login. You can access your login activity log by clicking the last successful login in the user account drop-down menu.

To view the history of logins for your Kony Fabric account, do the following:

1. Click the user account options menu.

   The menu displays the history of your last successful login.

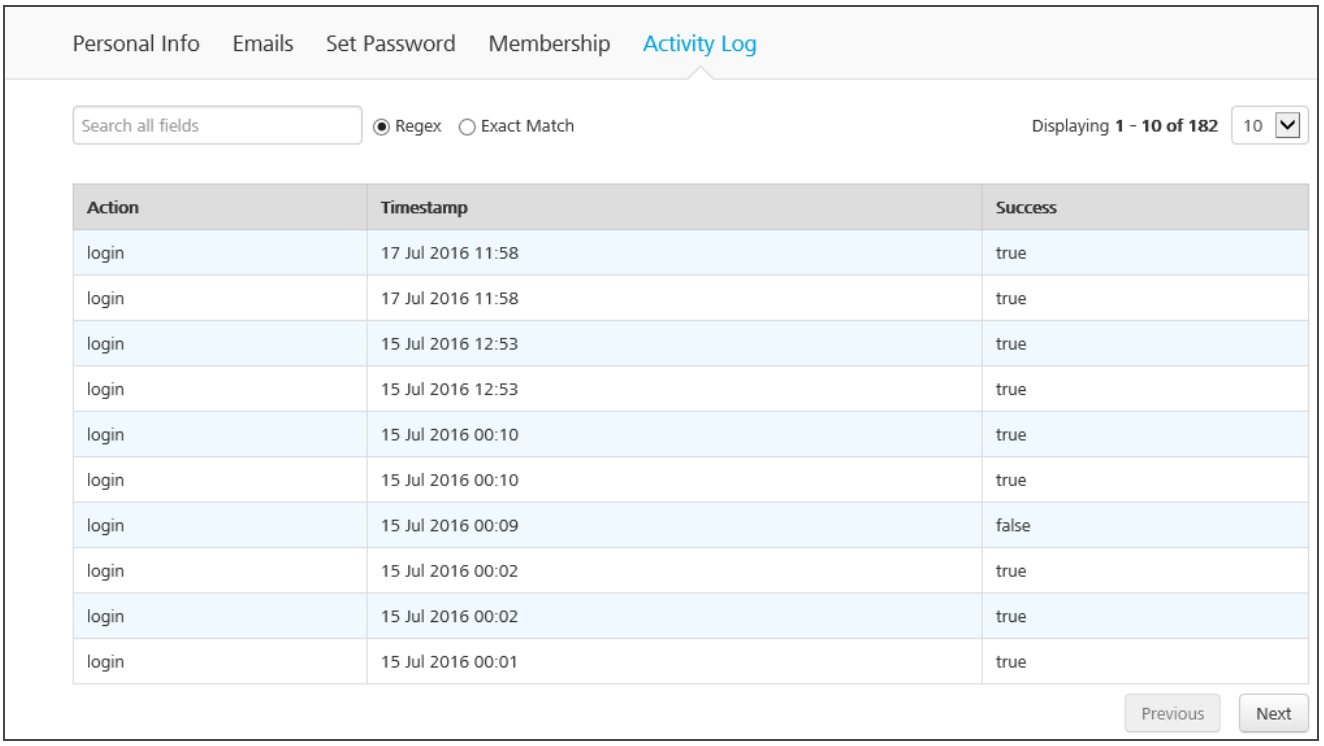
   

2. Click **Last successful login**.

   The login activity profile for your account appears. You can search the login activities by stipulating a regular expression or an exact match.

   You can also access your login activity log by clicking **Profile** in the user account drop-down

menu, and then clicking **Activity Log**.



7. Environments - On-Premises

You need to create an environment to publish your apps. Environments can include at least one server or a combination of all servers, such as Kony Fabric Integration, Kony Fabric Engagement, Kony Fabric Sync, and Kony Fabric Management.

> *Important:* As a user, you must be an admin or owner to access the Environments page and perform different tasks based on your role.

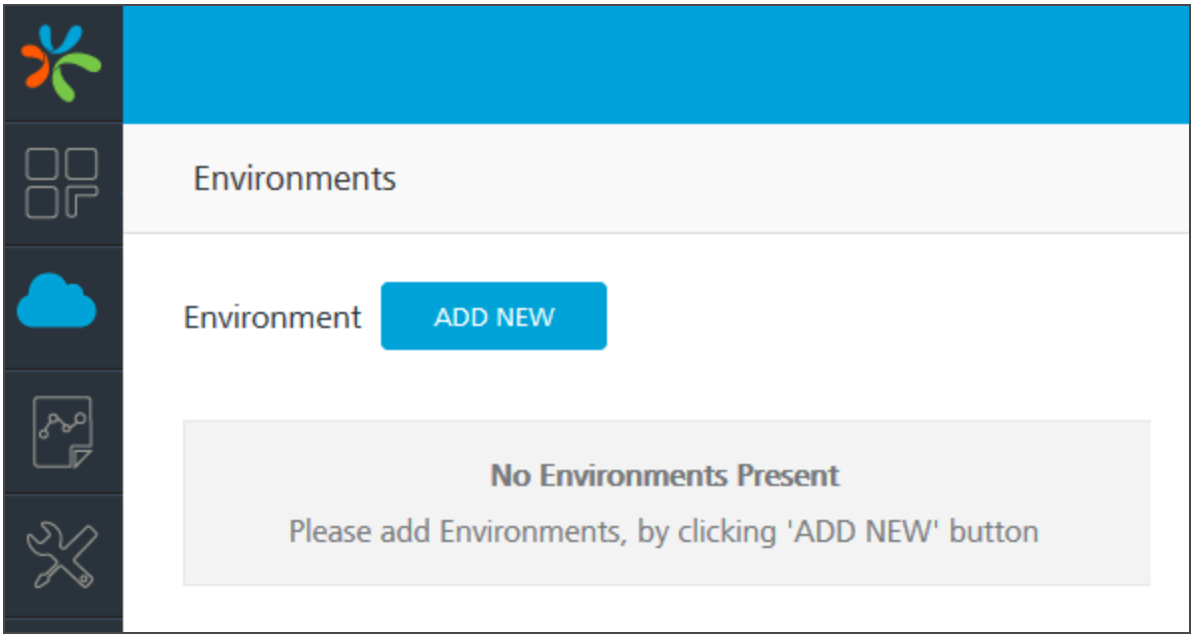
> *Important:* Ensure that your environments include all required servers that are part of an app.
>
> For example, if your environment contains only Kony Fabric Sync, and you try to publish an app with Kony Fabric Engagement, the system throws an error.

7.1 How to Add an Environment

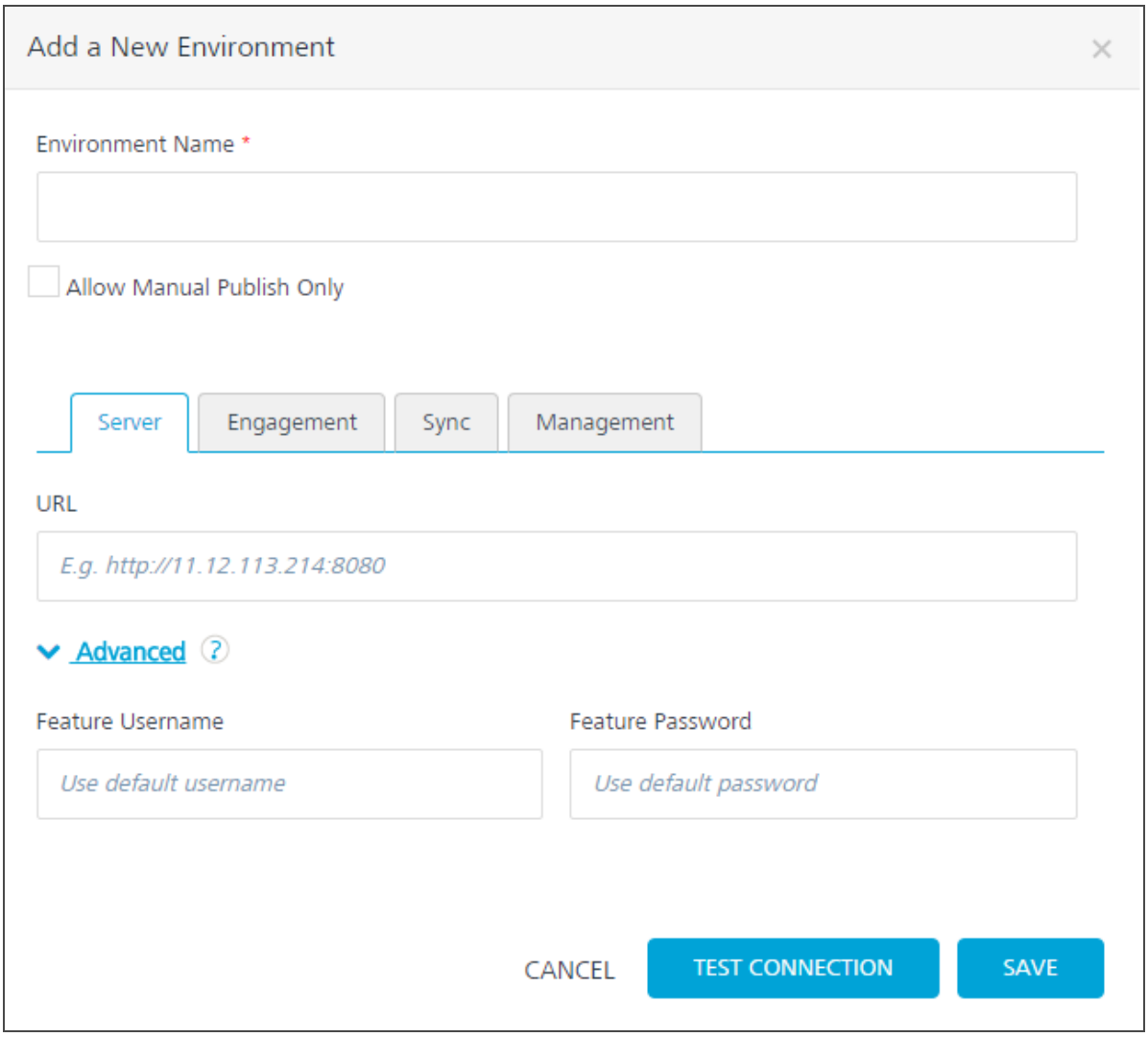
You can add environments with different combinations of servers.

To add an environment, follow these steps:

1. In your Kony Fabric account, in the left-pane, click **Environments**. The **Environments** page appears.



2. Click the **ADD NEW** button. The **Add a New Environment** dialog appears.



3. In the **Environment Name** text box, enter an environment name.

> ***Note:*** Your Environment name can only contain letters, numbers, and hyphens (-). A hyphen cannot appear at the beginning or at the end of a name. A number cannot appear at the beginning of a name. A name should be a minimum of three characters and a maximum of 20 characters long.

4. Select the **Allow Manual Publish Only** check box for not publishing the .war file if you have

uploaded it for **Web** platform under the **Manage Client App Assets** tab. By default, the **Allow Manual Publish Only** check box is cleared.

5. In the services section, follow these steps.

   The **Add a New Environment** dialog contains the Server, Engagement, Sync, and Management tabs. The input values are URL, username, and password. By default, the system will display the **Server** tab.

   a. In the **Server** tab, provide the following details:

      - **URL**: Enter the URL for your Kony Fabric Integration.
        The URL format is: `<http or https>://<server_host>:<server_port>`
        For example, a sample URL: `http://mbaastest30.konylabs.net:53504`
      - Under **Advanced:**
        - **Feature Username**: By default, this field shows the default username of Kony Fabric Integration. You can modify the username, if required.
        - **Feature Password**: By default, this field shows the default password of Kony Fabric Integration. You can modify the password, if required.

   > ***Note:*** You need to modify the username and password only if these credentials are changed via Kony Fabric Server Console.

   a. To configure the Kony Fabric Engagement, click the **Engagement** tab, and provide the following details:

      - **URL**: Enter the URL for your Kony Fabric Engagement.
      - Under **Advanced**:
        - **Feature Username**: By default, this field shows the default username of Kony Fabric Engagement. You can modify the username, if required.

- **Feature Password**: By default, this field shows the default password of Kony Fabric Engagement. You can modify the password, if required.

> ***Important:*** Support for Kony Fabric Engagement is available from Kony Fabric Engagement Version 6.0.1 onwards.

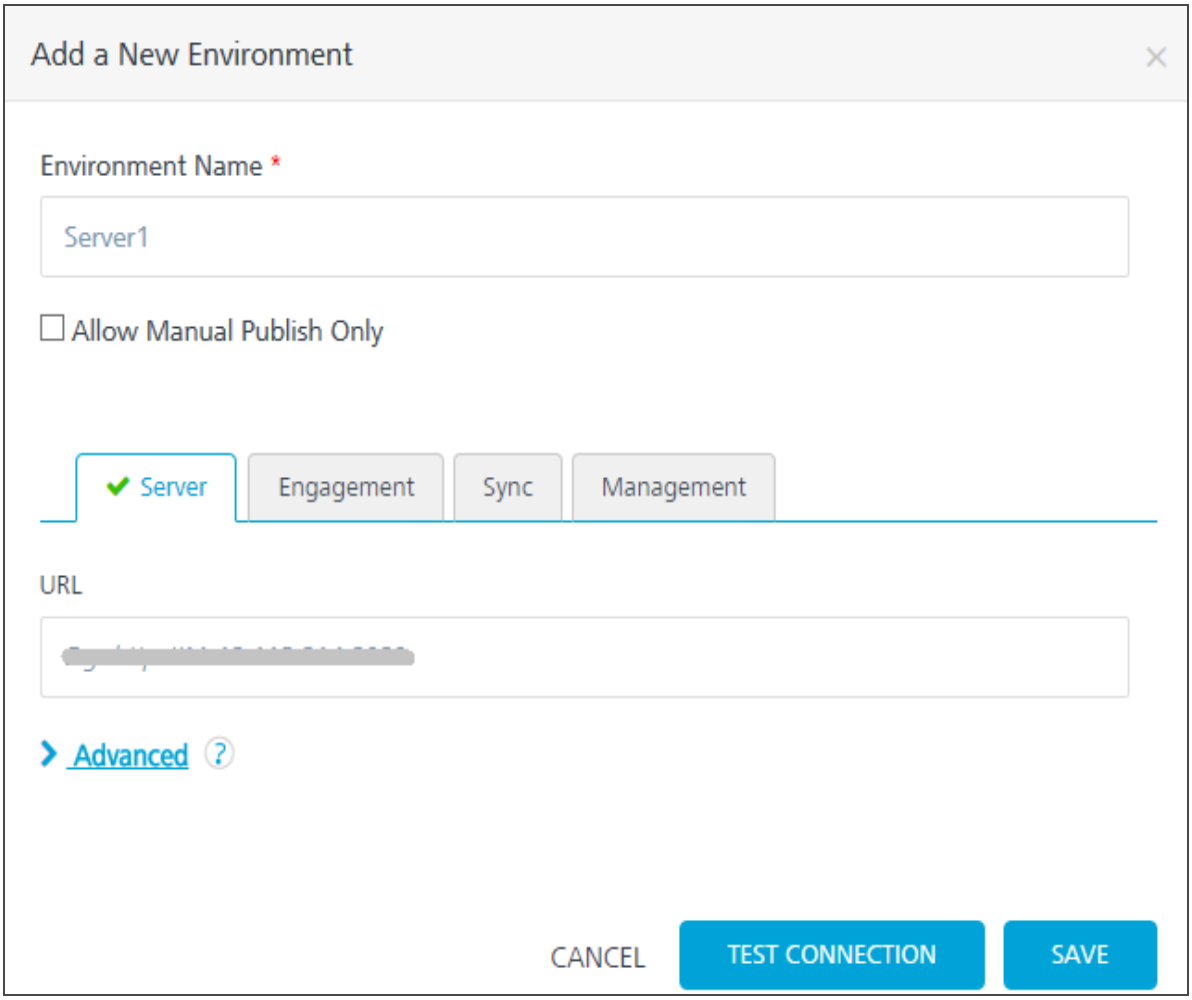
b. To configure the Kony Fabric Sync, click the **Sync** tab, and provide the following details:

- **URL**: Enter the URL for your Kony Fabric Sync.
- Under **Advanced**:
  - **Feature Username**: By default, this field shows the default username of Kony Fabric Sync. You can modify the username, if required.
  - **Feature Password**: By default, this field shows the default password of Kony Fabric Sync. You can modify the password, if required.

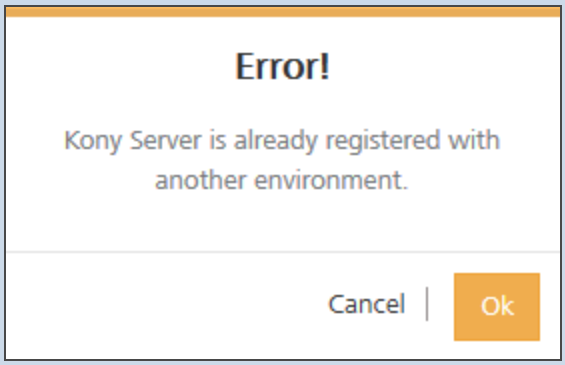
c. To configure the Kony Fabric Management, click the **Management** tab, and provide the following details:

- **URL**: Enter the URL for your Kony Fabric Management.
- Under **Advanced**:
  - **Feature Username**: By default, this field shows the default username of Kony Fabric Management. You can modify the username, if required.
  - **Feature Password**: By default, this field shows the default password of Kony Fabric Management. You can modify the password, if required.

6. Once you enter details, click **TEST CONNECTION**.
   If the server details are correct, the system displays a check mark next to a service, shown below:

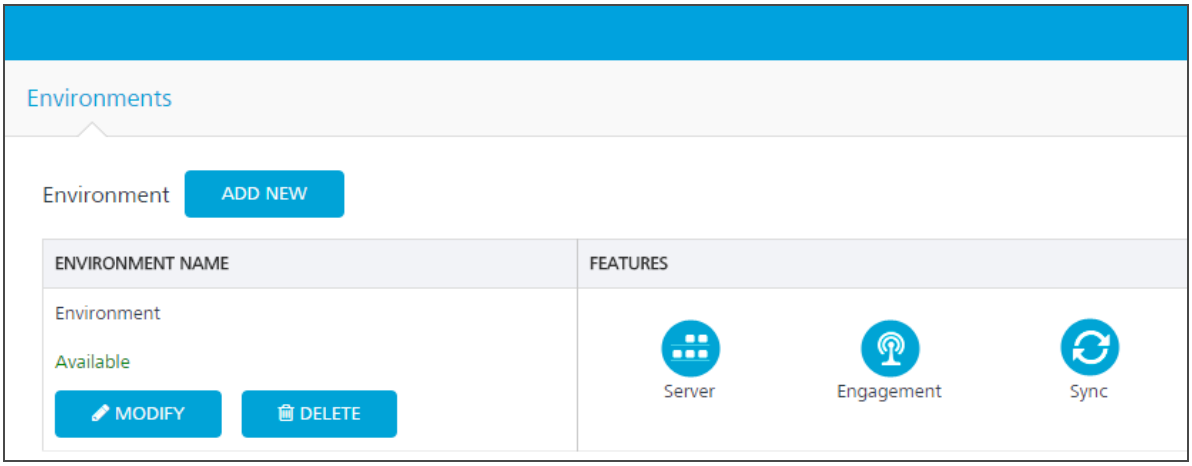


Important: The system allows you to add a unique server URL to only one environment. If a server is already configured with an environment and you try to add the same server to another environment, the system will throw an error, shown below:



Click **OK** to confirm.

7. Click **SAVE** to apply the environment capabilities. The environment is created in the **Environments** page.



The **Environment** list view displays the following columns:

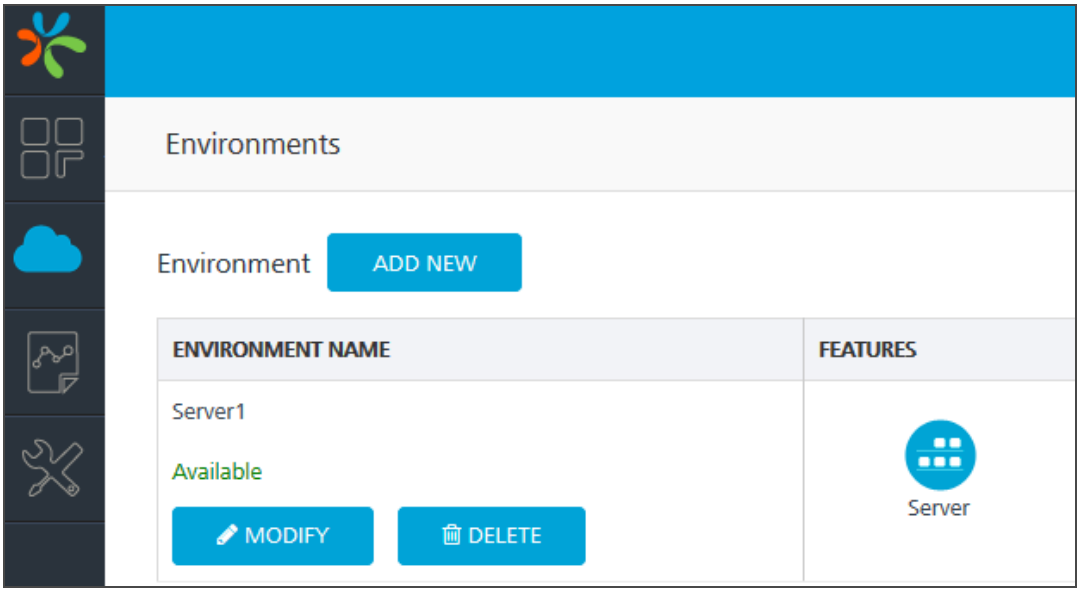
- **Environment Name**: Displays the name of the environments.
- **Administration**: Displays servers configured for an environment such as Kony Fabric Integration, Kony Fabric Engagement, Kony Fabric Sync, and Kony Fabric Management.

7.2 How to Modify an Environment

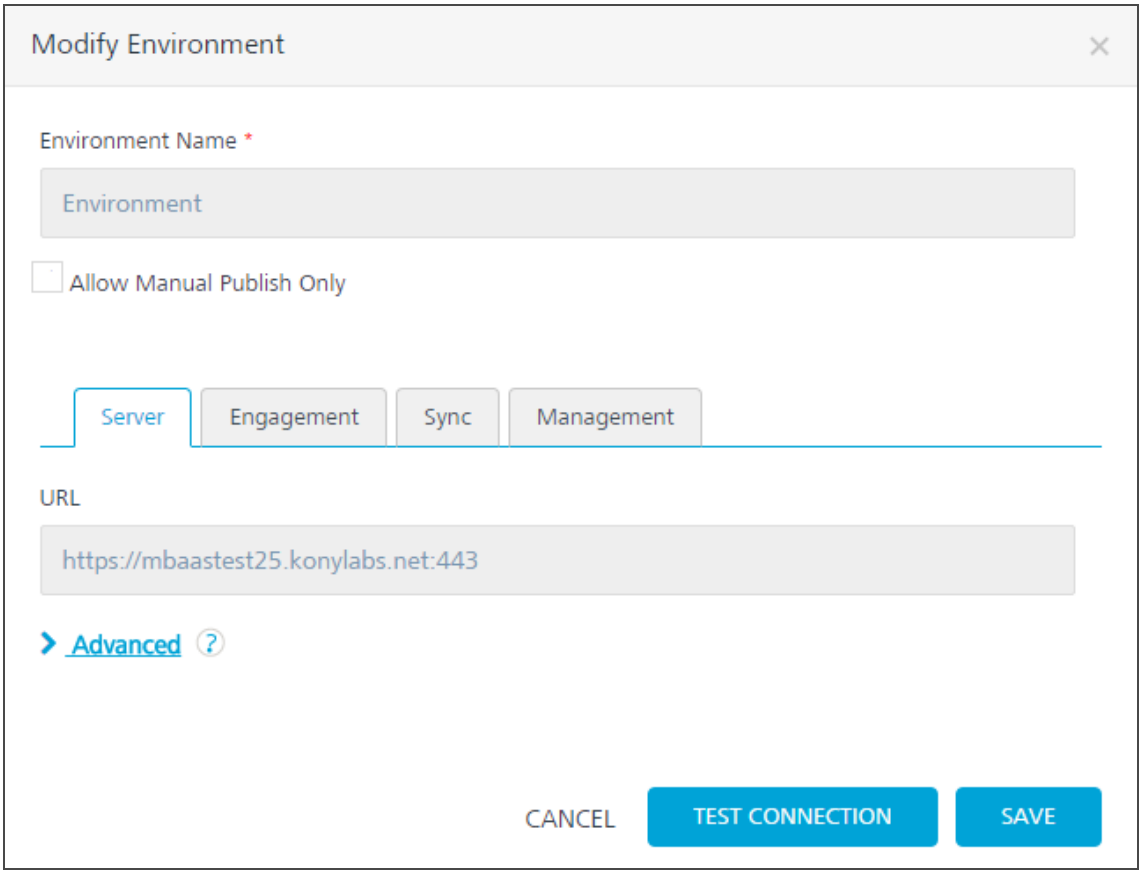
The Kony Fabric Console allows you to view as well as add servers to an environment.

To modify an environment, follow these steps:

1. In your Kony Fabric account, click **Environments**. The **Environments** page appears.
2. Click **MODIFY** for an environment.



The **Modify Environment** page appears, shown below:



You cannot edit the **Environment Name** and **URL** fields.

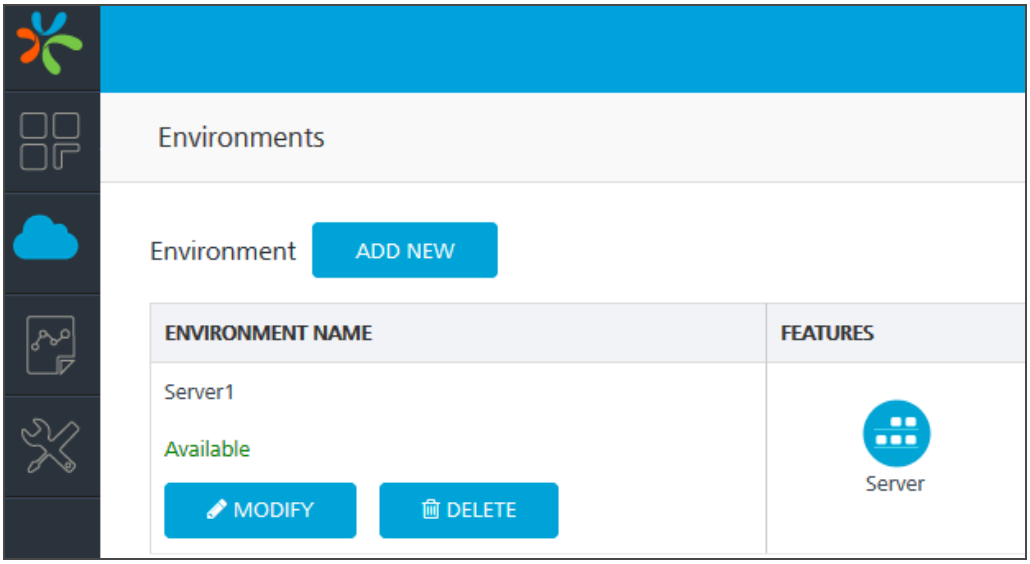
3. You can modify the environment, if required. Select the **Allow Manual Publish Only** check box for not publishing the .war file if you have uploaded it for **Web** platform under the **Manage Client App Assets** tab.
4. Click other tabs to add servers.
5. Click **TEST CONNECTION** to validate the details.
6. Click **SAVE**.

Note: You can view all the applications published to a Runtime Environment. Refer to Applications Published to a Runtime Environment.

7.3 How to Delete an Environment

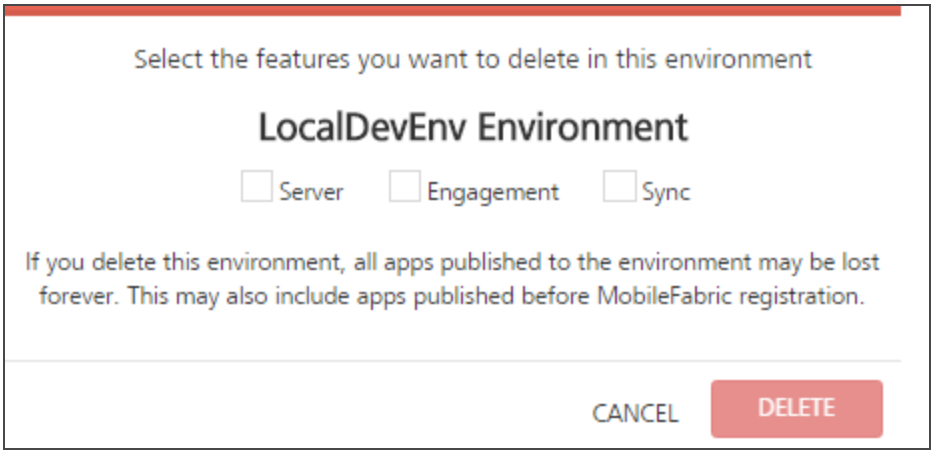
When an environment is deleted, the system deletes the environment and its data from the console.

1. In your Kony Fabric account, click **Environments**. The **Environments** page appears.



2. In the **Environment Name** column, select an environment, and then click **DELETE**.

   The **Delete** confirmation dialog appears, shown below:



> ***Note:*** The **DELETE** button dims when you have not selected any check boxes for an environment. When you select a check box for an environment, only then the **DELETE** button is available.

3. Select the check box for each of the listed environments and click **DELETE.**

   The system deletes the environment from the grid. If you delete this environment, all apps published to the environment may be lost forever. This may also include apps published before Kony Fabric registration.

> ***Note:*** You can view all the applications published to a Runtime Environment. Refer to Applications Published to a Runtime Environment.

8. Viewing Applications Published to a Runtime Environment and EAS

You can view the list of apps published to an environment and EAS and the runtime status of the apps through the **Environments** > **More Options** > **Published Apps** page.

From Kony Fabric V9 onwards, the list of apps published to your EAS is displayed in the **Published Apps** page, in addition to the apps published to your environment.

> ***Note:*** The **More Options** button in the **Environments** page is available only if you have the Admin access.

From the **Published Apps** page, you can perform the following actions:

- Navigate to the application's definition of the published app.
- Navigate to **Published Services**:
    - **Published Channel Assets** section displays the list of channels for which an app has been published. This is applicable only to the apps that are published to EAS.
    - **Unpublish** button is associated with the channel type of the app published to EAS. You can unpublish an app from EAS for a specific channel by clicking the **Unpublish** button associated to that channel.
    - The **Published Services** section displays the list of all the services associated to the current version of the published app.

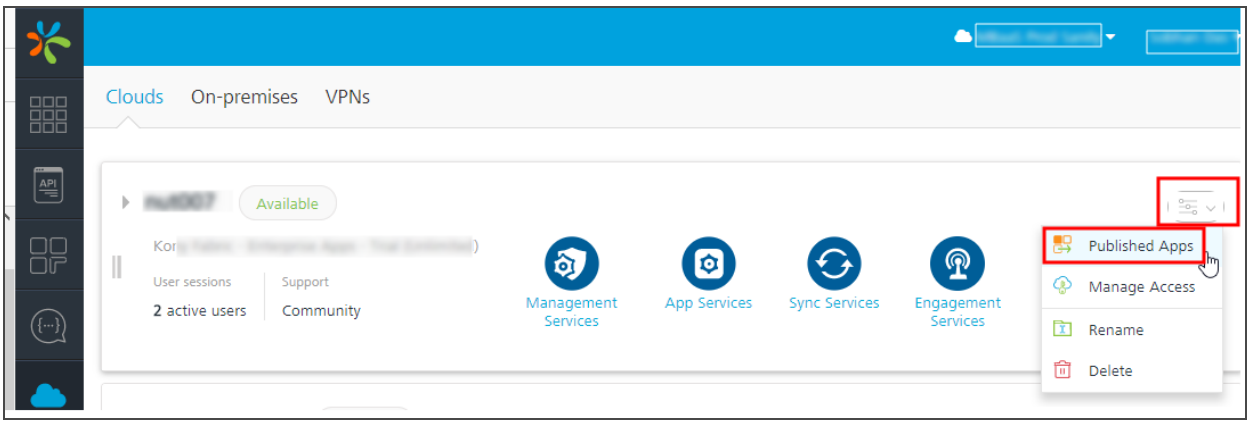
To view the list of all the apps published to an environment, follow these steps:

1. From the left pane in your Kony Fabric Console, click **Environments**. This displays the list of clouds or environments configured for your Kony Fabric account.

2. Click the **More Options** button of an environment.

> ***Note:*** The **More Options** button in the **Environments** page is available only if you have the Admin access.

3. From the context menu, select **Published Apps**.

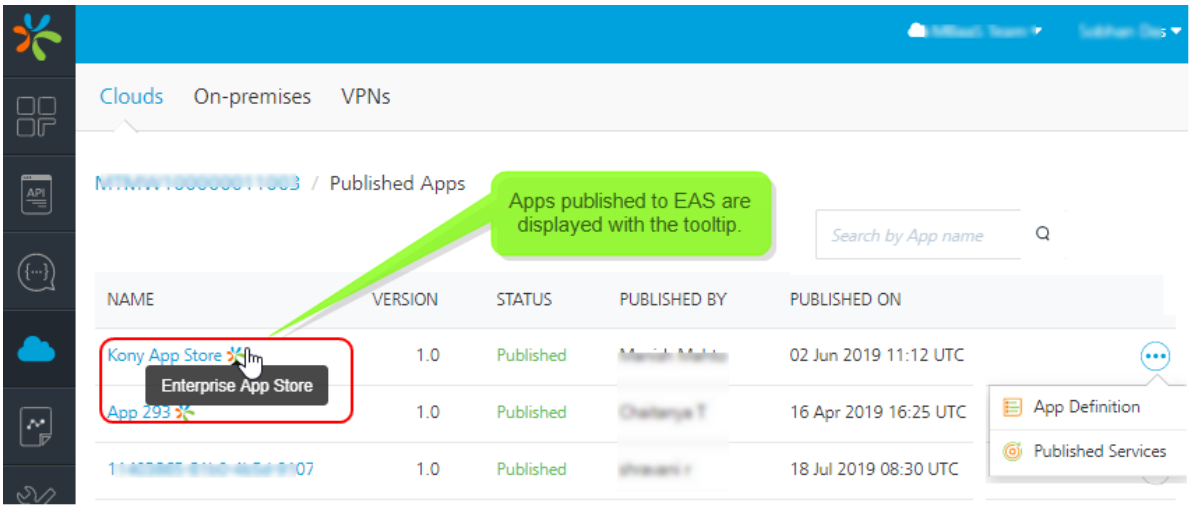


The following details for all the apps published to the environment are displayed in the **Published Apps** page.

- **NAME**: Displays the name of the published app.
  - The apps published to EAS are indicated with the EAS app icon and a tooltip.
- **VERSION**: Displays the version of the published app.
- **STATUS**: This section displays the status of the app published in the environment. Following are the different statuses available for an environment.
  - **Published**: App is published to this cloud or environment.
  - **In Progress**: App is in the process of being published to this cloud or environment.
  - **Error**: The app publishing process is canceled while publishing, or there was an error while publishing.

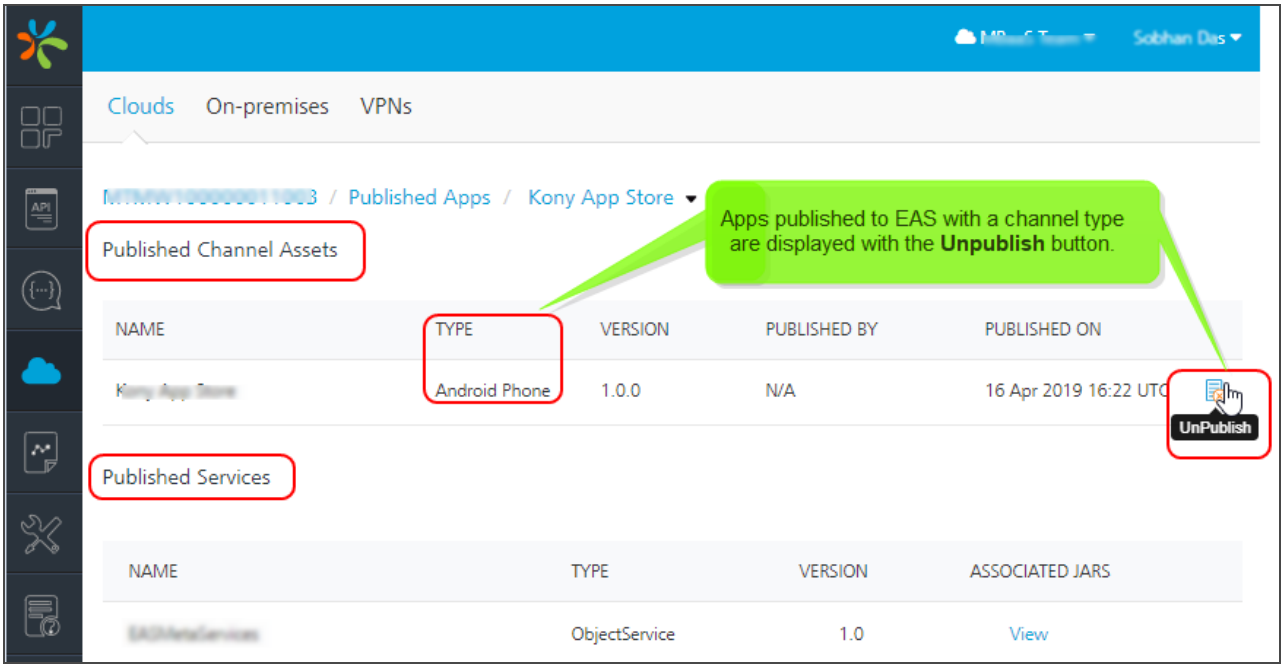
- **PUBLISHED BY**: Displays name of the user that published the app to this environment or cloud.
- **PUBLISHED ON**: Displays the date and time when the app was last published.

From the **Published Apps** page, you can navigate to the definition and published services of an app. For example, `<Environment-name> Published Apps > <app-name>` page.



4. To view the application's definition, click **More Options** > **App Definition**. The Identity Service Designer page for the app is displayed. You can edit the service definition and services, if required. If you edit any details, these changes will not be available at runtime environment until you publish the updated app/services again.
5. To view the associated services and the channel types with the published app, click the **More Options** > **Published Services**.

   The following app details are displayed for the published apps: *app publish status*, *app version*, *app publish time stamp*, and the *user name* of the user who published the app.



- The **Published Channel Assets** section is applicable only for the published apps to EAS. This section displays the following details:

Column	Description
NAME	Displays the name of the app binary published to EAS.
TYPE	Displays the type of the channel to which the app has been published.
VERSION	Displays the version of the app.
PUBLISHED BY	Displays the name of the user who published the app.
PUBLISHED ON	Displays the time-stamp when the app was published.
Unpublish button	You can click this button to unpublish the app.

- The **Published Services** section displays the following details for the associated services:

Column	Description
NAME	Displays the name of the service.
TYPE	Displays the type of the service.
VERSION	Displays the version of the service.
ASSOCIATED JARS	Displays the **View** hyperlink. When you click the **View** link, the list of jars associated with that service is displayed. This column is shown only for Application Servers with Kony Fabric V7.2 or later.

9. Features

Following are the features of Kony Fabric:

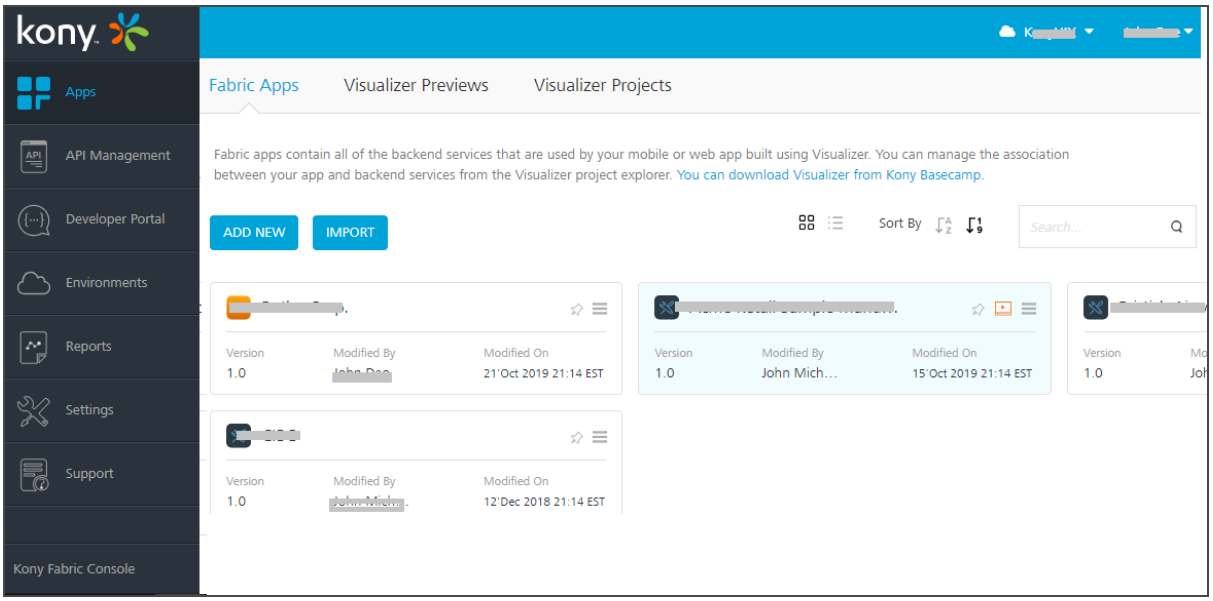
- Adding Applications: You can provide the basic details of an app, such as the name and icon. You can add the following services to the app:
    - Identity: This feature allows you to define the type of authentication used for granting access to your application.
    - Integration: This feature allows you to define various back-end services for your application.
    - Orchestration: Service orchestration is the coordination or integration of several services and exposing them as a single service. This feature allows you to create two types of orchestration services.
    - Objects: Allows you to create app models for LOB objects, storage objects, and Service-Driven Objects.
    - Logic: The logic services feature in Kony Fabric helps you import and integrate Node.js services (APIs) directly into Kony Fabric for developing server-side and networking applications.
    - Offline sync: This feature allows you to define the synchronization services for your application. Sync supports only Web Services, except SAP Sky.
    - Engagement: This feature allows you to send push notifications, email, SMS and passes to subscribed applications.
- API Management: Configure and manage (create, edit, and delete) app services (identity, integration, and orchestration) without linking or configuring them within an app.
- Export and Import Apps: Export apps from one workspace (Kony account) and import them to different workspaces of Kony Fabric Console.

- Manage Client App Assets: Manage client binaries through Kony Fabric Console such as creating mobile applications, publishing the apps to a Kony Management Environment and Server.
- Publish: After adding the required services, publish your app.

10. How to Add Applications

To add an app to your Kony Fabric, follow these steps:

1. From Kony Fabric Console's left pane, click **Apps**. In the right pane, the **Fabric Apps** page appears. By default, the **Fabric Apps** page appears listing existing apps.



   - The **Visualizer Previews** page lists the test live previews that you performed in a particular Cloud account. Kony Visualizer supports the Run Live Preview option that you can use to preview a prototype of your Visualizer application.
     For more information on How to user Live Preview in Kony Visualizer, refer Live Preview
   - The **Visualizer Projects** page lists the projects that you published to a particular Cloud. The Project tab in Kony Visualizer contains the Export > Cloud Project option.
     For more information on How to share a project on the Cloud, refer Publish your project to the cloud

2. In the the **Fabric Apps** page, click **ADD NEW**. By default, the **Configure Services** tab is selected.

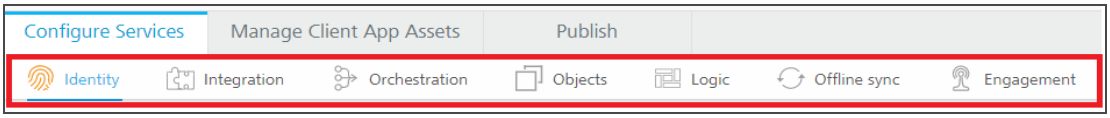
> ***Note:*** From Kony Fabric V8 SP4, you can pin your favorite apps for quick access on the Apps page by clicking the **Pin To Dashboard** button of an app. So, the next time you log on to Kony Fabric, you will notice that all the last pinned apps are available on the Apps page (Dashboard).

> ***Important:*** You can access your favorite pinned apps until you clear the cookies from your browser. You cannot access the apps you pinned in one browser by logging in from another browser or a private browser.

3. A new app is added, and you are directed to the **Identity** page of the new app. From here you can manage an app and add services to your app.
    - Manage your app as follows:
        - Click the **Image** button to select an image from your local machine.

          > ***Note:*** The image size should be less than 20 KB.

        - Click the **Sample Code** button to view the sample code of your app.
        - Click the **Edit App Name** button to provide a unique name for your app.
        - Click **Console Access Control** button to control the access to the applications and services of apps.
        - Click the **Delete** button to delete the app from your account.
        - Click the **Import** button to import an app to your account.
        - Click the **Export** button to export the current from your account.
    - Under the **Configure Services** tab, add and configure Kony Fabric services.



You can add and configure the following services under the Configure Services tab:

- Identity
- Integration
- Orchestration
- Objects
- Logic
- Offline sync
- Engagement

After you add services to your app, you can also do the following from an app page:

- Manage Client App Assets
- Publish

11. Console Access Control

Kony Fabric supports access controls for Kony Fabric applications and services. Kony Fabric users that have the permission to create Kony Fabric apps and services can control the access to the applications and services.

For example, an owner invites a new user to the Kony Fabric account. The new user can create a Kony Fabric application and services, and can then control the access by other users to the application. The role of a user determines the access the user has to applications and services, the access control the user can set, and the access control that other users can set for that user. For example, a user that has a member role can create an app and then give full access to the app to specific member users, while setting read-only access for all other member users. Users that have an owner or admin role always have full access to all Kony Fabric apps and services.

11.1 Use Case

The following describes a use case for access control of Kony Fabric applications and services. The scenarios in this use case help you understand how users can control access to applications and services. This use case also shows how the role of a user determines the level of access the user can control, and the user's level of access that other users can control.

Account Owner

1. A user creates a new Kony Fabric account. As the user that created the account, he is the first user and is assigned the Owner role. He is referred to as AccountOwner.

   An owner of a Kony Fabric account has full access rights to all Kony Fabric apps that are created on the account. The owner also has permissions to perform create, retrieve, update, and delete (CRUD) operations in all the services in all the apps.

2. The AccountOwner invites AppUser1 and AppUser2 to the Kony Fabric account as members.

   By default, AppUser1 and AppUser2 have the rights to create new apps and services and have full access to existing apps and services.

Create Applications

1. AppUser1 creates App A that has a Weather Service and a News Service
2. AppUser2 creates App B that has a GeoLocation Service and ATM Locator Service.

   By default, the applications and services that users create have global read-write access for all users of the account.

Control Access to Applications

1. AppUser2 decides that he needs to protect his App B.
2. From the apps page, AppUser2 selects *Console Access Control* from the App menu.
3. AppUser2 removes general access for all users and adds himself as a specific access user.

   At this point, AppUser3 logs in and can no longer see App A in the console. If AppUser2 had set read-only for All Users, then AppUser3 could see the application, but not modify its configuration (for example, add or remove services).

Control Access to Services

1. AppUser2 decides that he needs to control access to a service in his App B from other users.

   Identity, Integration and Orchestration services are shared components. AppUser3 can still go and modify any of those services which would affect the functionality of App A. If AppUser2 wants to completely lock down App A, he would also have to change the access control of all the services associated with the app.
2. AppUser2 opens App A and selects *Console Access Control* from the Settings menu for a service.
3. AppUser2 adds himself as a specific access user to the service just as he did for App B.
4. AppUser2 downgrades general access for all users to Read Only.

AppUser2 cannot completely remove access for all users to the service. He can only downgrade all users from Full Access to Read Only. This is because other users have access to an app, but they do not have access to all the associated services.

Full Access to App and Full Access to Service

In the case where a user has full access to the app and full access to the service:

- The user can add and create new services, unlink the services, switch between versions of a service, delete a version of a service, and save a service as a new version.
- The user can switch between the versions of a service, but he cannot save the service or unlink the service regardless of whether he has read-only or full access to the service.

Full Access to App and Read Only Access to Service

In the case where a user has full access to the app and read only access to the service:

- The user can use the Use Existing option to add a service. If the user has full access rights on the service he adds, he can modify the service.
- The user can clone a read-only service and gain full access to the new service.
- The user can configure a new service and unlink a read-only service.
- The user cannot edit a read-only service in the app, but he can save it (clone it) as a new service.
- The user can change the version of the read-only service within the app.
- The user can switch between the versions of the service within the app, as he is configuring the app and not the service. He can also unlink the same service. However he cannot create a new version of the service or delete a version because he does not have full access to the service.

Read Only Access to App and Full Access to Service

In the case where a user has read-only access to the app and full access to the service:

- The user cannot modify the configuration of the app, and the Use Existing and Clone services are disabled.
- The user cannot unlink a service or configure new service to the app.
- The user can modify the service, but cannot save it as a new service within the app, as he does not have full access rights to modify the app.
- The user cannot change the version of the service within the app.

Read Only Access to App and Read Only Access to Service

In the case where a user has read-only access to both the app and service:

- The user cannot modify the configuration of the app, and the Use Existing and Clone services are disabled.
- The user cannot unlink a service or configure new service to the app.
- The user cannot edit the service and save it as a new service from within the app.

11.2 How to Use Access Control

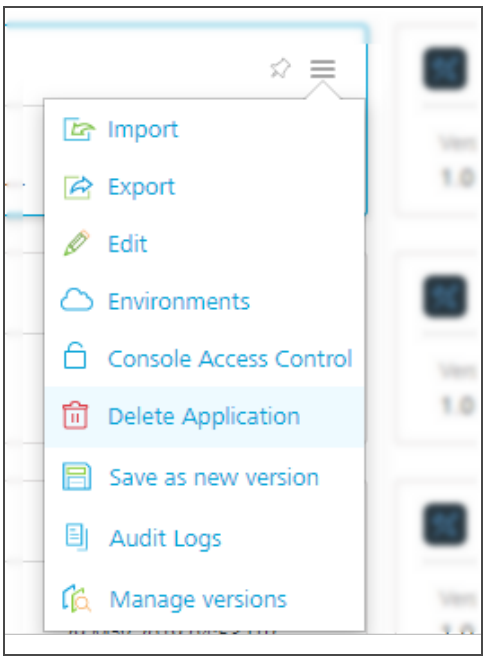
You can configure access control for Kony Fabric applications and services. By default, all users have full access rights to create and access apps and services. Use the Apps Console Access Control page to control access to an applications. Use the Services Console Access Control page to control access to a service.

11.2.1 How to Use Access Control for Applications

You can access the Apps Console Access Control page from either the Applications page or from within in an app.

To set access control from the Applications page, do the following:

1. From Kony Fabric Console, click **Apps** to display the Applications page.
2. In the **Applications** page, hover your cursor over the **App menu** button of an app.



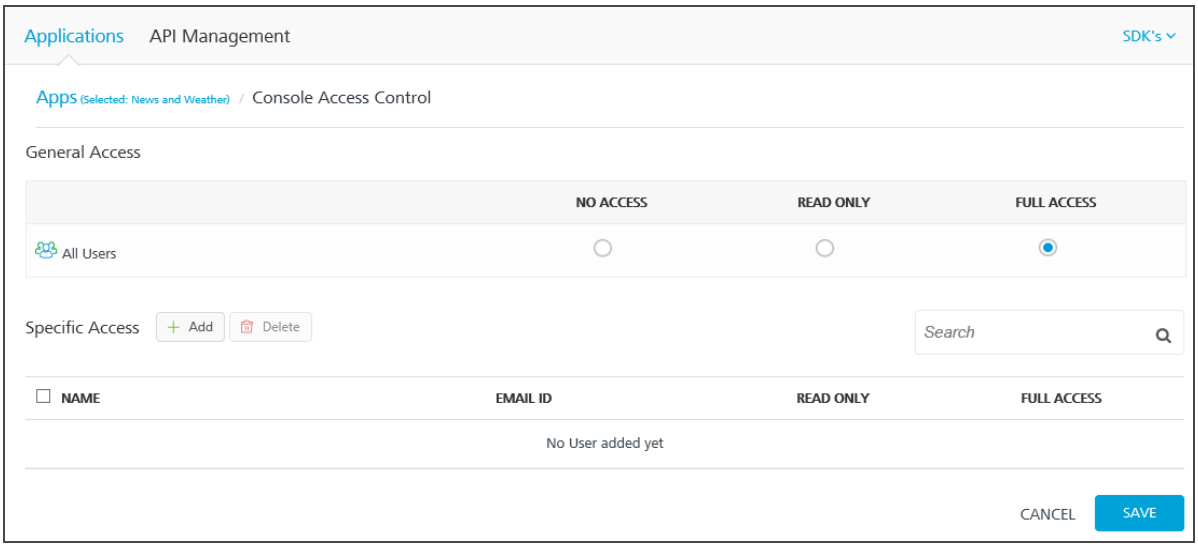
3. Click **Console Access Control**.

   The Apps Console Access Control page appears.

To set access control from an app, do the following:

1. From Kony Fabric Console, click **Apps** to display the Applications page.
2. In the **Applications** page, select an app.
3. Click the Console Access Control button  .

The Apps Console Access Control page appears.



To set general access to an app for all users, do the following:

- In the **General Access** area, for **All Users**, select an access control option.

  By default, all users have full access rights to create and access apps. If you want to block access to any of the other members on the account, select No Access. Users with no access permissions cannot access the app and the app does appear on the Applications page.

  Before you select No Access on the General Access level, add yourself or another user on the Specific Access level.

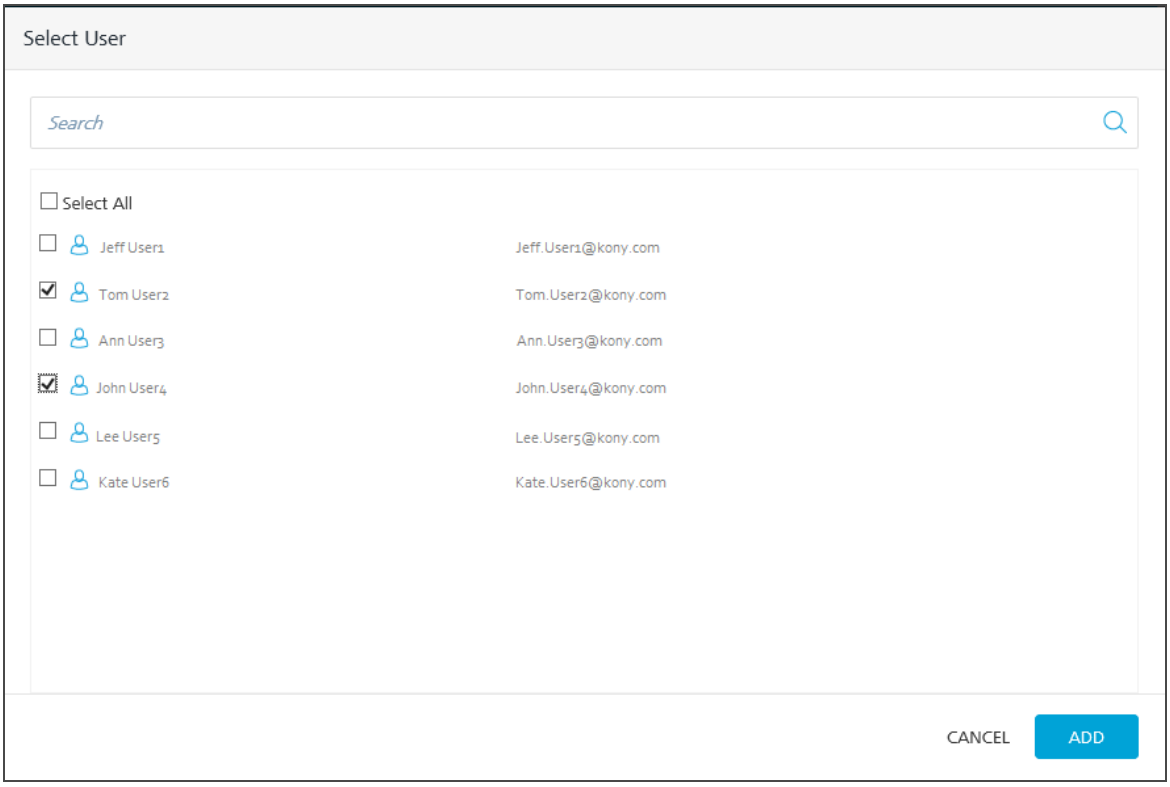
To set specific access to users for an app, do the following:

1. In the **Specific Access** area, click **Add**.

   The Select User windows appears. All the owners and admins might not be shown in the list of Specific access users.

2. Select the users that you want to add to the Specific Access list.

   The Select User window appears.



3. Click **Add**.

   The users that you selected are added to the Specific Access list.

4. Select the access control option.

   Note that the Read Only option for a user is disabled if General Access permission is set to Full Access. You cannot set access control for Specific Access at a setting lower than General Access Setting.

5. Click **Save**.

If a member user gives a second member user Full Access permission for an app, both member users have the same permissions for the app.

To remove specific access to users for an app, do the following:

1. In Specific Access, select the users that you want to remove.
2. Click the **Delete** button.

   You can hover your cursor over a user, and then click the Delete icon.

What Full Access Permission for an App Means

- A user with full access can link or unlink any services to the app. The user can publish the app if the user has permissions to publish for that environment.
- A user with full access has permission to configure and change the control access list.

What Read Only Permission for an App Means

- A user with read-only access cannot link or unlink any services to the app.
- A user with read-only access cannot configure or change the control access list for the app.

11.2.2 How to Use Access Control for Services

You access the Services Console Access Control page from within an application or from API Management.

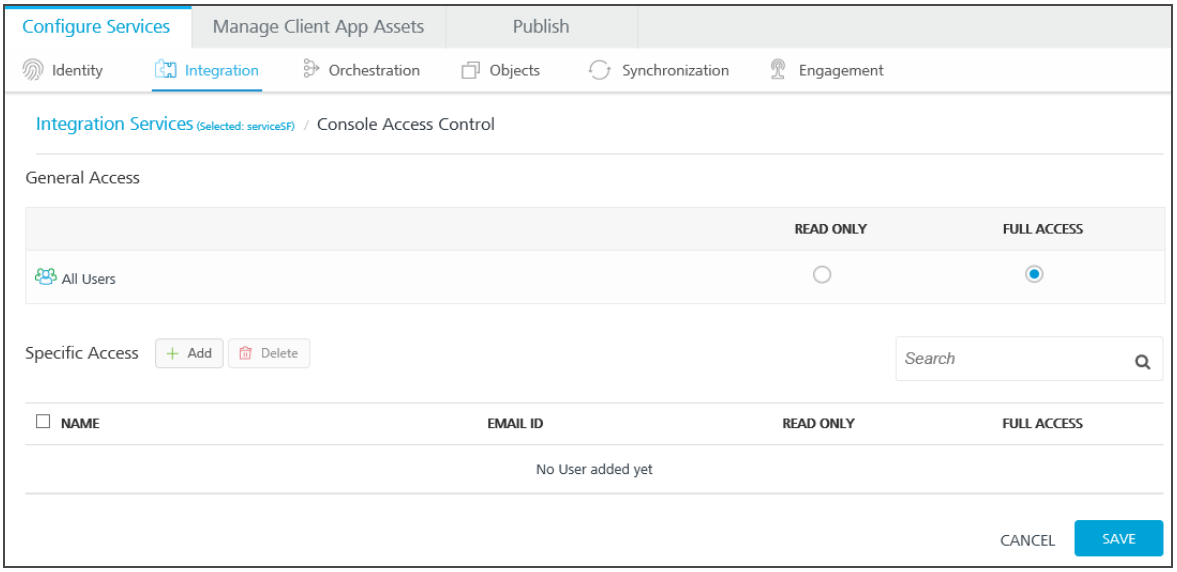
By default, all the services created have Full Access for all users. You can control access to an integration services, orchestration services, and object services by setting the access control for the service. For example, UserA decides to give read-only access to all users for an integration service (General Access), and give full access to the service to UserB (Specific Access).

To set access control for a service from an app, do the following:

1. From Kony Fabric Console, click **Apps** to display the Applications page.
2. In the **Applications** page, open an app.
3. In **Configure Services** tab, click the Integration, Orchestration, or Objects service tab.

4. Hover your cursor over the required service, click the **Settings** menu , and then click **Console Access Control**.

   The Services Console Access Control page appears.



To set access control for a service from API Management, do the following:

1. From Kony Fabric Console, click **Apps** to display the Applications page.
2. In the **Applications** page, click **API Management**.
3. Click the Integration or Orchestration service tab.
4. Hover your cursor over the required service, click the **Settings** menu , and then click **Console Access Control**.

   The Services Console Access Control page appears.

To set general access to a service for all users, do the following:

- In the **General Access** area, for **All Users**, select an access control option.

  By default, all users have full access rights to create and access services. Before setting the general access to read-only for all users, give specific access to at least one user. For example, UserA adds UserB to the Specific Access user list and gives him full access. Then UserA sets Read Only for General Access. UserB now has full access to this weather services but another member user, User3, has read-only access.

To give specific access to users for an app, do the following:

1. In the **Specific Access** area, click **Add**.

   The Select User windows appears. All the owners and admins might not be shown in the list of Specific access users.

2. Select the users that you want to add to the Specific Access list.

   The Select User window appears.

3. Click **Add**.

   The users that you selected are added to the Specific Access list.

4. Select the access control option.

   Note that the Read Only option for a user is disabled if General Access permission is set to Full Access. You cannot set access control for Specific Access at a setting lower than General Access Setting.

5. Click **Save**.

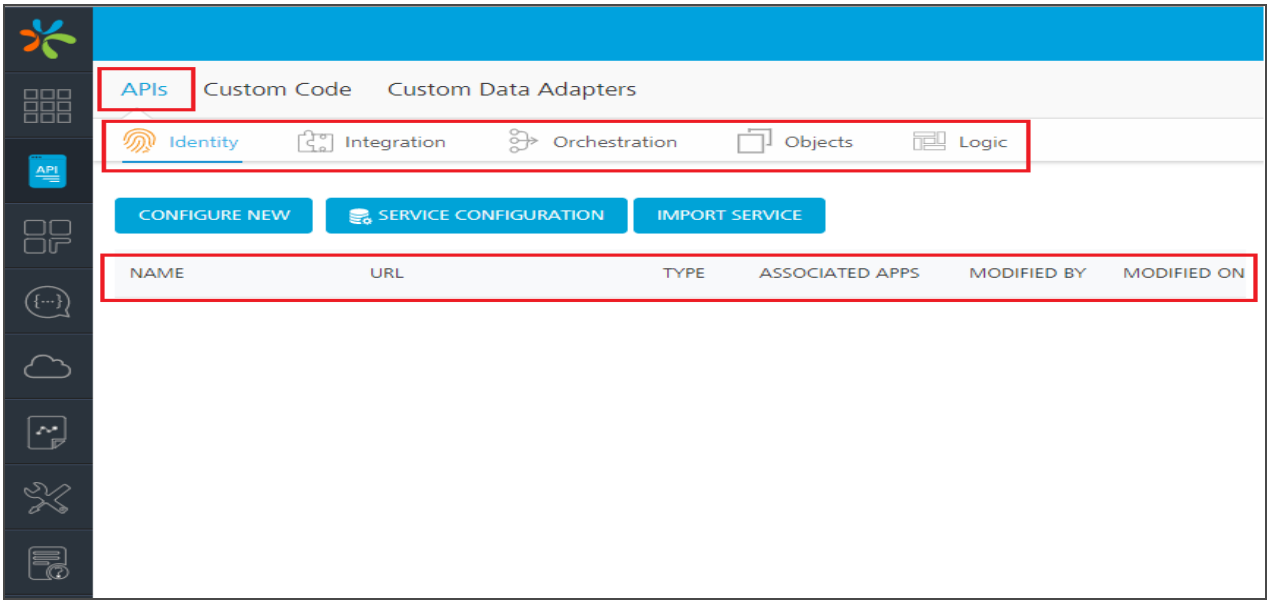
12. APIs - API Management

A Kony Fabric app comprises a group of services, shared and non-shared services. With API Management, you can manage (create, edit, and delete) shared services (identity, integration, orchestration, objects, and logic for Node.js services) without linking or configuring them within an app. After configuring the services in the **APIs** page, you can edit, clone, view a sample code, and delete a service. When you create the services in the **APIs** page, the services are not linked to apps automatically. You can link the services across any apps created for that account in Kony Fabric Console. The **APIs** page also allows you to view the list of apps or services that are using or referencing a given service. When you make changes to the services in the **APIs** page, the changes will be reflected in the services associated with other apps.

> ***Note:*** You can edit a service. When you make a change to the service, the changes will be reflected in the services associated with other apps.
>
> You cannot delete a service if it is associated with an app or a service.

To create services through the APIs page, follow these steps:

1. In the Kony Fabric Console, in the left-pane, click the **API Management** tab to display the services tabs such as **APIs**, **Custom Code**, and **Custom Data Adapters**. By default the **Identity** service tab is selected under the **APIs** tab.



Under the **APIs** page, the **Identity**, **Integration**, **Orchestration**, **Objects**, and **Logic** tabs appear and display list the existing services (if any). Under the **APIs**, the **Identity**, **Integration**, **Orchestration**, and **Objects** views display the following columns:

Column	Description
NAME	Displays the Name of the service.
URL	Displays the URL of the service. ***Note:*** The URL column is displayed only for identity service.
ENDPOINT TEYnPdEp / SERVICE TYPE	Displays the type of the service. The TYPE column is displayed only for identity service. ***Note:*** The SERVICE TYPE column is displayed only for integration service.

Column	Description
ASSOCIATED APPS	Displays the **View** hyperlink. When you click the **View** link, the system displays the **Associated Apps** page. The **Associated Apps** page displays the number of apps associated with a particular service. For more details, refer to Associated Apps
MODIFIED BY	Displays the name of the user.
MODIFIED ON	Displays the date and time of the modified service.

2. From the **APIs** page, follow these steps to create services:
   - Identity
   - Integration
   - Orchestration
   - Objects
   - Logic

12.1 Identity Service

Kony Fabric identity services help you secure your application by adding an authentication layer.

12.1.1 How to Create an Identity Service in APIs

1. In Kony Fabric Console, select **API Management** from the left navigation panel. The **Identity** page in the API tab appears by default. The **Identity** page appears and lists the existing identity services (if any). The fields for an identity service are displayed such as NAME, URL, TYPE, ASSOCIATED APPS, MODIFIED BY, and MODIFIED ON.

2. Click **CONFIGURE NEW**. A new identity service is added.
3. Configure the details for the identity service. For more details, refer to Identity.

Note:

- You can perform different actions on an existing service such as edit and delete. For more details, refer to Context Based Options.
- You can configure a default timeout for the apps globally in API Management. For more details, refer to Service Configuration > Identity Timeout Settings.
- Enabling cross-origin resource sharing (CORS) allows external web applications on domains to access the identity services in your Kony Fabric account. For more details, refer to Identity Service Security Settings.

12.2 Integration Service

An Integration Service is an application component that represents the application interaction with an external system or data source.

12.2.1 How to Create an Integration Service in APIs

1. In Kony Fabric Console, select **API Management** from the left navigation panel.
2. In the **APIs** page, click the **Integration** tab.
   The **Integration** page appears and lists the existing integration services (if any). The fields for an integration service are displayed such as NAME, SERVICE TYPE, ASSOCIATED APPS, VERSION, MODIFIED BY, and MODIFIED ON.
3. Click **CONFIGURE NEW**. A service definition tab is added.

4. Configure the details for the integration service. For more details, refer to Integration.

> ***Note:*** After creating an integration service in the **APIs**, you can perform different actions on an existing service such as edit, clone, view a sample code, delete all versions of a service, manage versions of a service, and export an integration service. For more details, refer to Context Based Options.
> Services created under the **APIs** page are not linked to apps. You can link or unlink services to an app only through the **Existing Services** dialog while you are adding apps.

12.3 Orchestration Service

An Orchestration service leverages the concept of combining multiple integration services into a single orchestration service to reduce the complexity and number of calls from the app to the backend.

12.3.1 How to Create an Orchestration Service in APIs

1. In Kony Fabric Console, select **API Management** from the left navigation panel.
2. In the **APIs** page, click the **Orchestration** tab.
   The **Orchestration** page appears and lists the existing orchestration services (if any). The fields for an orchestration service are displayed, such as NAME, ASSOCIATED APPS, VERSION, MODIFIED BY, and MODIFIED ON.
3. Click **CONFIGURE NEW**. A service definition tab is added.
4. Configure the details for the orchestration service. For more details, refer to Orchestration.

> ***Note:*** After creating an orchestration service in the **APIs**, you can perform different actions on an existing service such as edit, clone, view a sample code, delete all versions of a service, manage versions of a service, and export an orchestration service. For more

> details, refer to Context Based Options.
>
> Services created under the **APIs** page are not linked to apps. You can link or unlink services to an app only through the **Existing Services** dialog while you are adding apps.

12.4 Object Service

Kony Fabric Object Services enable model-driven application design and development by following a microservices architectural approach to create reusable components and link them to fit into your solution.

12.4.1 How to Create an Object Service in APIs

1. In the **APIs** page, click the **Objects** tab.
   The **Objects** page appears and lists the existing objects services (if any). The fields for an object service are displayed, such as NAME, ENDPOINT TYPE, VERSION, MODIFIED BY, and MODIFIED ON.
2. Click **CONFIGURE NEW**. A service definition tab is added.
3. Configure the details for the object service. For more details, refer to Objects.

> ***Note:*** After creating an object service in the **APIs**, you can perform different actions on an existing service such as edit, clone, clone app data model, sample code, unlink an object service or delete a specific version of a service. For more details, refer to Context Based Options.
>
> Services created under the **APIs** page are not linked to apps. You can link or unlink services to an app only through the **Existing Services** dialog while you are adding apps.

12.5 Logic Service

Kony Fabric logic services help you import and integrate Node.js services (APIs) directly into Kony Fabric for developing server-side and networking applications.

12.5.1 How to Integrate Node.js Services into Kony Fabric Apps using API Management

1. How to Publish Node.js Package into Node.js Runtime Server in API Management. This section details how to import a Node.js package into Kony Fabric Console and publish the package to the Node.js Runtime Server.
2. How to Integrate Node.js Services into Kony Fabric Apps. This section details how to integrate Node.js services to Kony Fabric apps.

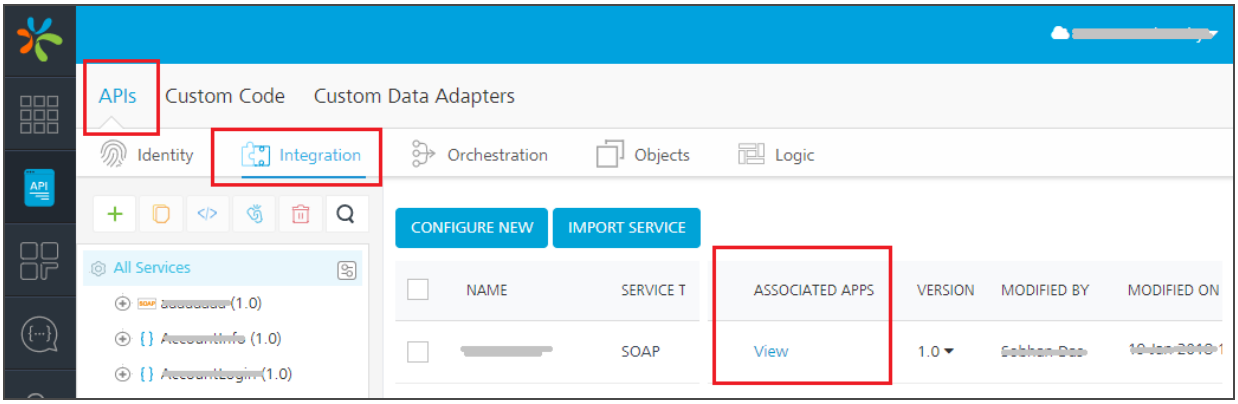
> ***Note:*** For more details on how to Publish Node.js Package into Node.js Runtime Server in API Management, prerequisites, use case, limitations, and Node.js package structure, refer Logic - API Management

12.6 How to View Associated Apps in APIs

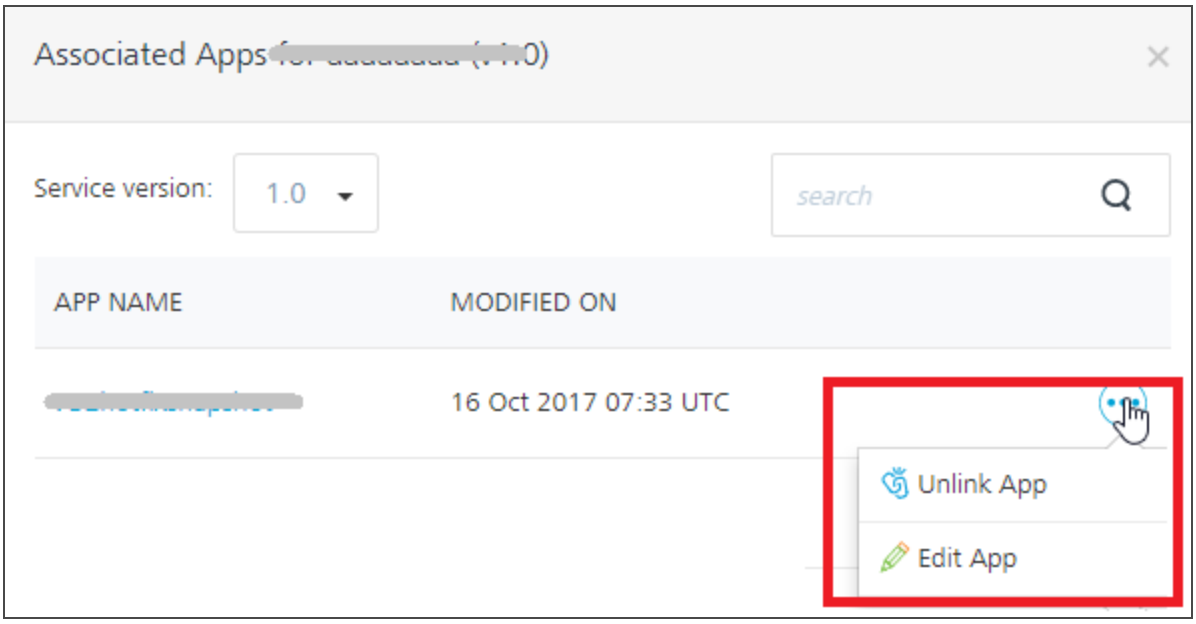
After you link services (identity, integration, orchestration, and objects) created in the **APIs** page, you can view the list of apps that are associated to the services through the **Associated Apps** page. From the **Associated Apps** page, you can edit the app and unlink the app from the service.

To view associated apps, follow these steps:

1. In the **APIs** page, click the service (identity, integration, orchestration, objects) tab to display the service details page.



2. Click the **View** link under the **ASSOCIATED APPS** column to view the apps associated to the service. The **Associated Apps** dialog appears with the list of apps linked to the service for the current version. You can change the version of the service if required.



From the **Associated Apps** page, you can perform operations such as edit the app and unlink the app from the service. For identity services, from the Associated Apps page you can also enable SSO for the application.

- To edit the app through the **Associated Apps** page, hover your cursor over the required service, click the **Settings** button, and then click **Edit**.

- To enable SSO for the app through the Associated Apps page, hover your cursor over the required service, click the **Settings** button, and then click **Enable SSO**. You must republish the app for this new setting to take effect. For more information about SSO for applications, refer to Application SSO.
- To unlink the app through the **Associated Apps** page, hover your cursor over the required service, click the **Settings** button, and then click **Unlink app**. When you click the **Unlink app** button, the app is disassociated from a particular service.

12.7 Context Based Options

You can perform various actions on an existing service from the **API Management** page.

In the API Management page, click the contextual menu of the required service. It contains the following options:

- **Edit**: Opens a service in the console and allows you to edit the service definition and operations. After you edit a service, you need to republish all the apps that are using the service to apply the changes.

  > ***Note:*** If a service is a part of a published app, you can rename that service only after the app is unpublished.

- **Edit Configuration**: For objects services, opens the service in the console and allows you to edit the objects service definition and operations.
- **Clone**: Duplicates an existing service. Clone a service to create a different version of the service. Changes made to a cloned service will not affect the original service. The name of a cloned service indicates that it is a copy of an existing service.
- **Clone App Data Model**: For objects services, duplicates an existing app data model of an objects service. Clone a service to create a different version of the service. Changes made to a cloned service will not affect the original service. The name of a cloned service indicates that it is a copy of an existing service.
- **Publish Service**: Republishes a service. Refer Publishing Individual Services

- **Sample Code**: Generates dynamic code for each SDK type based on the configuration of a service. You can use the code in your mobile app. For example, generate the sample code for an integration or orchestration service from Kony Fabric. Then use that code in the mobile app to invoke the orchestration service instance.
- **Delete all versions**: Deletes all versions of a service. You cannot delete a version of the service that is in use. A service in use is a service that is referenced by a Kony Fabric app or another service or a Sync scope.
    i. Click the **Delete all versions** button to display the **Delete Service** dialog.
    ii. Click **DELETE**. If the current of the service is not linked to any apps, the versions is deleted. Otherwise, the Error warns you that you are trying to delete the current version of the service that cannot be unlinked or deleted as it is being used by the following apps or services or Sync scopes: [App Name]

    > ***Note:*** If a service is a part of a published app, you can delete that service only after you unlink the service from all the published app.

- **Manage versions**: Delete one or more versions of a service.
    i. Click the **Manage versions** button to display the **Manage versions** dialog lists versions of the service.
    ii. Hover your cursor over the required version, and click the **Delete** button. When you click the **Delete** button, the **Delete Service** dialog appears.
    iii. To confirm the deletion of the version, click **DELETE**. If the current version of the service is not linked to any apps, the versions is deleted. Otherwise, the Error warns you that you are trying to delete the current version of the service that cannot be unlinked or deleted as it is being used by the following apps or services or Sync scopes: [App Name]

> ***Important:*** You cannot remove the current version of a service from a service level. For example, If you are in the **Service Definition** tab of a service from **APIs** and try to delete a version of a service, you can delete all the versions of a service except the current version.

- **Audit Logs** helps you to capture all the user activities performed in a service. **Object Name**, **Object Type** and **Modified On** fields are prepopulated with the **Service Name**, **Services**, and **Last 7 Days** respectively.

  For more information on Audit Logs, refer to Audit Logs documentation.

  > ***Note:*** This feature is applicable only for Integration, Orchestration, and Object Services.

- **Console Access Control**: Controls the access to the applications and services of apps.
- **Export as XML**: Exports an existing version of a service to an XML file.
- **Export**: Exports an existing version of a service to a zip file. You can import the service to an existing app in the Kony Fabric Console. For more information, refer Exporting and Importing an Application.
- **Validate**: Validates the service definition of an objects service.

12.8 Service Configuration

After an Identity service is configured, from Service Configuration section, you can configure session timeout (idle timeout and fixed timeout) for an app identity session which applies across to all the apps that use this service. You can also configure cross-domain security settings, security headers from here.

Click **API Management** > **APIs** > **Identity** > **Service Configuration** to navigate to the Service Configuration page.

> *Important:* At least one identity service must be configured before you can enable cross-domain security.

You can do the following from the Service Configuration page:

- Configure Identity service security settings.
- Set the identity timeout duration. In the **Identity Timeout Settings** section > **Session Duration** box, type required duration in HH:SS.
- Add custom response headers in a valid JSON array. These headers will be appended to the response of identity requests originating from applications. In the Custom Response Headers box, type the required response headers in the following format: `[{"name":"X-Content-Type-Options","value":"abc"}, {"name":"X-Frame-Options","value":"deny"}]`
- Click **Save**.

12.8.1 Identity Service Security Settings

The Identity service security settings allow external web applications on existing domains to access the Identity service in your Kony Fabric account. You can configure the following from this section:

- **XDomain proxy for Internet Explorer 8 and Internet Explorer 9** - To enable it, select the **Enable XDomain Proxy for IE 8 & 9 Support** check box.
- **Cross-origin resource sharing (CORS)** - It allows external web applications on existing domains to access the identity service in your Kony Fabric account. To enable CORS, select the **Enable Cross-Origin Resource Sharing (CORS)** check box, and do the following:
    - Under **Settings**, select the radio button that corresponds to how your services will set Access-Control-Allow-Origin headers:
        - None
        - All

- Echo
- If you select Echo, in the **List of Domains** box, type one or more approved (whitelisted) domains.

- **Custom Response Headers**: Add custom response headers in a valid JSON array. These headers will be appended to the response of identity requests originating from applications. In the Custom Response Headers box, type the required response headers in the following format:
`[{"name":"X-Content-Type-Options","value":"abc"}, {"name":"X-Frame-Options","value":"deny"}]`

> ***Important:*** The changes made to an Identity service here effects all the apps associated with it.

> ***Note:*** You must publish the app to reflect the changes.

> ***Note:*** More details to configure Identity Session Timeout and HTTP Message Body Integrity

12.9 Logic - API Management

The logic services feature in Kony Fabric helps you import and integrate Node.js services (APIs) directly into Kony Fabric for developing server-side and networking applications. You can now enhance the capability of your Kony Fabric applications by adding custom services that are created within Node.js packages/projects. Node.js is an open-source and cross-platform runtime environment. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on Mac OS, Microsoft Windows, and Linux. Node.js also provides a rich library of JavaScript modules that simplify the development of web applications using Node.js. Currently, Kony Fabric supports Node.js V6.2.2 or above.

12.9.1 Node.js Package Structure

A Node.js package contains the following folders and files.

Swagger.json:

- The Swagger.json file includes all paths for services / APIs of the Node.js package.
- The `host` key should be present and hostname should be **localhost** and port value must be within **9000 - 10000.**
- The `basePath` key should be present and value should with `/services/` and followed by user app base path.
- The `mfidentitylevel`, which is optional, is used to enable auth protection for the resources. Possible values are: `public`, `protected`, and `anonymous`.

The following is a sample code for swagger.json.

```
 {
  "info": {
    "title": "Kony Fabric Logic (Node.js) sample application for
managing Contacts",
    "version": "1.0.0",
    "description": "This sample application demonstrates use of Logic
(Node.js) app in Kony Fabric"
  },
  "host": "localhost:9000",
  "basePath": "/services/mailapp",
  "swagger": "2.0",
  "paths": {
    "/api/v1/contact": {
      "post": {
        "tags": [
          "Contacts"
        ],
        "description": "Contact Details object",
        "produces": [
          "application/json"
        ],
        "parameters": [
```

```
        {
          "name": "Details",
          "description": "Contact Details",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/Contact Details"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "Successfully created"
        }
      },
      "mfidentitylevel": "public"
    }
  }
},
"definitions": {
  "Contact Details": {
    "properties": {
      "firstName": {
        "type": "string"
      },
      "lastName": {
        "type": "string"
      },
      "rollNumber": {
        "type": "string"
      },
      "mobileNumber": {
        "type": "string"
```

```
        }
      }
    }
  },
  "responses": {},
  "parameters": {},
  "securityDefinitions": {},
  "tags": []
}
```

Package.json:

- The Package.json file contains Node.js module dependencies.
- The script `start` key must be present. Kony Fabric logic services uses the start key to start the user app. The following is a sample code for package.json.

```
{
 "name": "Sample_USERAPP",
 "version": "0.0.0",
 "private": true,
 "scripts": {
   "start": "node ./bin/www"
 },
 "dependencies": {
   "body-parser": "~1.15.1",
   "cookie-parser": "~1.4.3",
   "debug": "~2.2.0",
   "express": "~4.13.4",
   "hjs": "~0.0.6",
   "less-middleware": "1.0.x",
   "mfgateway": "^1.0.0",
   "morgan": "~1.7.0",
```

```
    "serve-favicon": "~2.3.0",
    "swagger-jsdoc": "^1.3.0"
  }
}
```

12.9.1.1 Sample Node.js Package Structure

User app: A user app is built on express JS. The following is a sample folder structure of a user app (Node.js package).

```
 Sample_USERAPP
├──bin
     |_www
|
├──public
├──routes
      |_ contact.js (Add your routes by adding new JS file here if you
are using Kony Fabric sample app)
|
├──views
├──app.js
├──package.json
├──swagger.json
```

12.9.2 How Node.js Works in Kony Fabric

1. Once you build a node.js package (in a .zip file), you need to import the package file into Kony Fabric Console.
2. Publish the Node.js package into the Node.js runtime server through the console.

   Node.js services published from the package are available in the **Kony Fabric Console > Apps > Configure Services > Logic** tab.

3. You can then integrate/associate services created in Node.js packages to Kony Fabric applications.
4. Finally, the app needs to be published by selecting a Node.js environment.

> *Important:* Ensure that your package details (files and folders) are zipped in the root of the .zip file.

12.9.3 Benefits of Node.js App Development

The logic services feature offers the following advantages for quickly mobilizing Node.js services in Kony Fabric for building apps:

- Logic services exposes services created in Node.js helps you use custom Node.js services for building mobile apps.
- Integrate custom services (APIs) created in Node.js language directly into Kony Fabric apps, and develop server-side and networking applications.
- Protect Node.js services (APIs) by setting security levels that control Kony Fabric Identity services for authentication of apps users.
- Build scalable network applications using Node.js technology in Kony Fabric apps.
- Node.js supports a non-blocking input/output model that makes it lightweight and efficient.

12.9.4 Use Cases

- For **creating real-time web apps**: Node.js technology improves the performance and development of web apps.
- For **creating Web sites**: Node.js technology helps you share code between the browser and the back end.

12.9.5 Limitations

Read the following limitations for Node.js services before associating the services with Kony Fabric apps.

- Currently, one Node.js package for one user is allowed.
- Logic services tab support is available only for Kony Cloud.

12.9.6 Prerequisites

- Always use the `npm install save` option to install any modules.

  For example, to add the package in dependencies, follow the commands:

  ```
  npm install --save my_dep
  ```

  or

  ```
  npm install -S my_dep
  ```

  The above commands ensure that all the dependencies are updated to `package.json` file.
- Ensure that your package details (files and folders) are zipped in the root of the .zip file.
- Ensure the port specified in swagger.json range starts from 9000 to 10000.
- Ensure that the port in swagger.json should match with the one specified in the application code.
- Avoid the `npm_modules` folder in the final .zip file, which you want to upload to Kony Fabric. The `npm_module` folder increases unnecessary package file size.

Important: The user app should not depend on PORT environment variable. By default, if you are using **express-generator** to generate an express basic project, the `bin/www` folder will have a PORT environment variable to refer to port value. In the `bin/www` folder, use port number directly.

For example,

```
var port = normalizePort(process.env.PORT || '3000');
```

will become

```
var port = normalizePort('9000');
```

12.9.7 Logging Service Support for User App

The following sample code is to enable logging in the user apps. Using the mflogger, the user app logs can also be captured.

```
 let mflogger;

// Kony logger to capture application logs to display using logging
service
try {
// The following require statement imports the admin app's logging
service into the user app through which user can capture logs.
mflogger = require(process.env.KONY_LOGIC_LOGGER);
} catch(e) {
// If the above statement fails, all the user app logs are captured in
the default console output.
mflogger = console;
}
```

Use the `mflogger` instance for logging in the user app. Supported logging methods: `debug`, `info`, `warn`, and `error`.

```
mflogger.info('Message to log ...');
```

12.9.8 Troubleshooting

Unauthorized: If you see the following message indicates that your endpoint resource is protected by Kony Fabric identity services configured for your application. So you need to generate Kony Fabric login token and provide it as part of header with name as `X-Kony-Authorization`

```
{
    "message": "Authentication failed"
}
```

Error: If you see the following error, unpublish the user package from the **API Management > Logic** tab, and re-publish it.

```
{
    "error": "Connection error."
}
```

For technical questions, suggestions, and comments, or to report problems on Kony's product line, contact support@kony.com.

12.9.9 Node.js Services Integration in Kony Fabric

Node.js services integration in Kony Fabric apps involves the following two steps:

1. How to Publish Node.js Package into Node.js Runtime Server in API Management. This section details how to import a Node.js package into Kony Fabric Console and publish the package to the Node.js Runtime Server.
2. How to Integrate Node.js Services into Kony Fabric Apps. This section details how to integrate Node.js services to Kony Fabric apps.

12.9.9.1 How to Publish a Node.js Package into Node.js Runtime Server in API Management

After creating services in Node.js, you must zip (package) the project including services, and then publish the package to Node.js Runtime Server by using Kony Fabric Console.

To publish Node.js package, follow these steps:

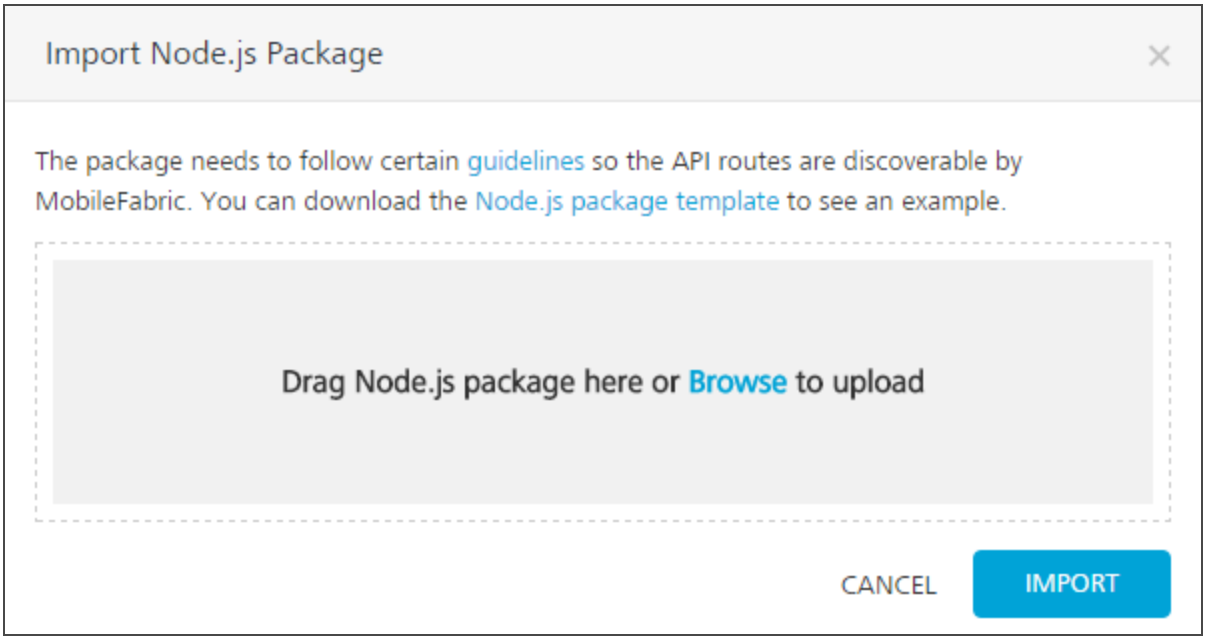
1. In the API Management page, click the **Logic** tab.

   The **Logic** page appears and lists services imported from a Node.js package (if any).

2. If you do not have a package.zip, click **Node.js package template** link to download sample package. The https://github.com/kony/Kony-Logic-Nodejs-Contact-Sample page appears.

3. In the Github page, click the `userpackage.zip` file and click **Download**.

4. In the Kony Fabric API Management > Logic, click the **IMPORT PACKAGE** button. The **Import Node.js Package** dialog appears.

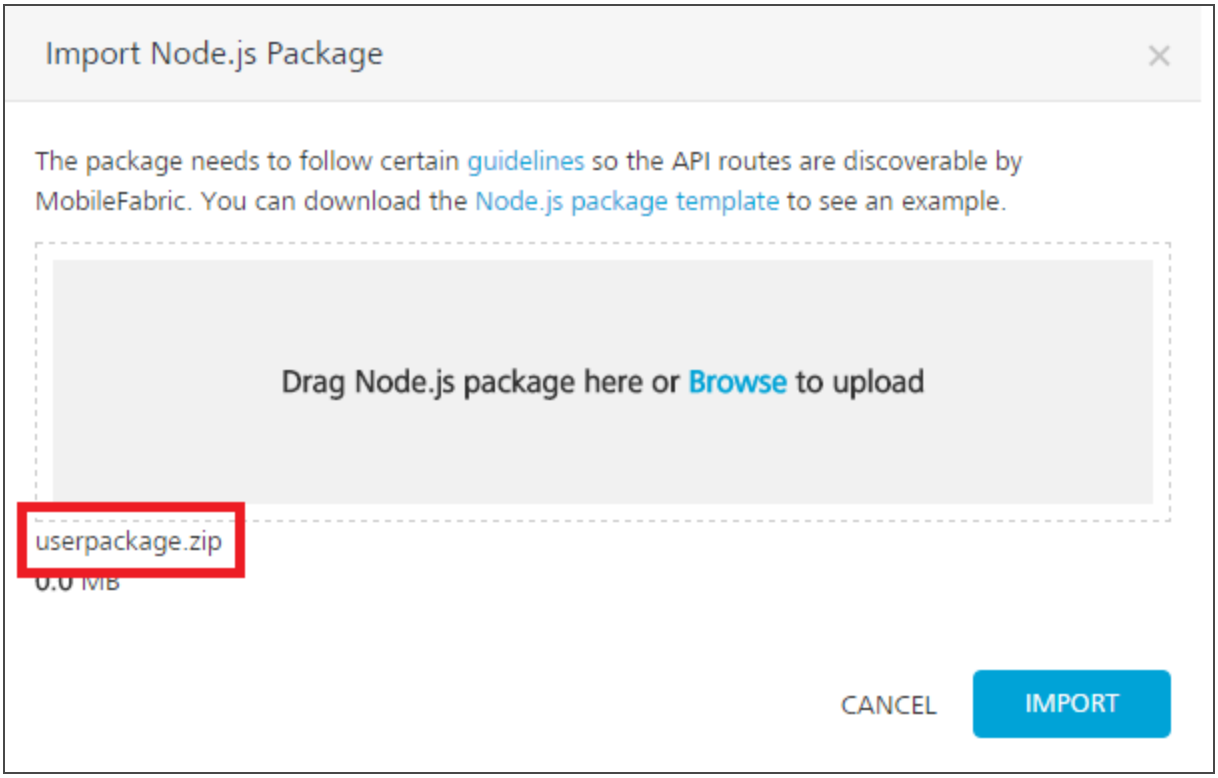


5. In the **Import Node.js Package** dialog, drag the `Node.js` package from your local system and drop it to the dialog. You can also click **Browse** to select the package through the Microsoft Windows Open dialog. In the **Open** dialog, locate your exported package (for example, USERAPP.zip file), and select it. Click **Open**. In the Import Node.js Package dialog, the selected files from the package are added to the dialog.

   While importing a package, if the existing package in Kony Fabric Console has the same name as the importing package, the system throws an error, shown below: The system overwrites the existing package with the data in the .zip file.

   > ***Important:*** While replacing a package, if the package names are same, the new data will override the existing data.

   The selected package.zip is loaded into the **Import Node.js Package** dialog.



6. Click **IMPORT**. The services from the package are imported to the **Logic** tab.

After the Node.js package is imported into Kony Fabric Console, the following fields are displayed:

Number	Field	Description
1	**NODE.JS PACKAGES**	Displays the imported package name.
2	**RELATIVE PATH FOR SERVICES**	Displays the relative paths of the imported services.
3	**METHOD**	Displays the method type of the service.

Number	Field	Description
4	**SECURITY LEVEL**	Displays the security level set in the swagger.json file for the service. You can set the security level for required relative path using `mfidentitylevel`. For example: for the service. `"mfidentitylevel":"<securitylevel>"` • The possible values for security level <public/protected/anonymous> The following security levels are supported. • **protected** - indicates that the operation is secured. To use the operation, an app user must be authenticated by an associated identity service. • **anonymous** - indicates that a user must have the app key and app secret to access the operation. • **public** - indicates that the operation requires no special security. **To change the security level, follow these steps**: 1. Click the **Edit** button under the **SECURITY LEVEL** column, and select the type of security level from the drop-down list. 2. Click **SAVE**. To cancel the changes, click **CANCEL**.

Number	Field	Description
5	**Export Package**	Allows you to export a package in a zip file.
6	**Delete Package**	Allows you to delete a package. ***Important:*** You cannot delete a package if it is associated with any of the apps. You cannot delete a package if it published to Node.js Runtime Server.
7	**PACKAGE LAST UPDATED**	Displays the date and time of Node.js package when last updated.
8	**ASSOCIATED APPS**	Displays the **View** hyperlink. When you click the **View** link, the system displays the **Associated Apps** page. The **Associated Apps** page displays the number of apps associated with a particular service. For more details, refer to Associated Apps.
9	**PUBLISHED STATUS**	Displays the **View** hyperlink. When you click the **View** link, the system displays the **Environments** dialog. The **Environments** dialog displays the names of the environments, publish status of the package, and runtime consoles associated with a particular Environment.

After the Node.js package is imported into Kony Fabric Console, you can perform the following actions along with publishing the package.

- To replace the imported Node.js package, click **REPLACE PACKAGE** . Replacing the package will erase the existing package and will remove the links to the associated

application.

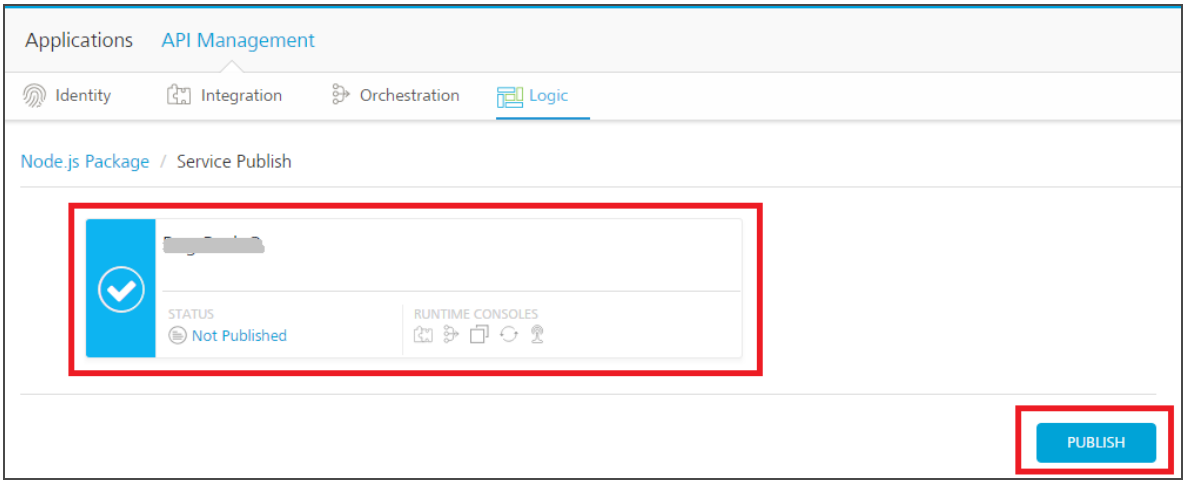
- To export the updated Node.js package, click **Export Package**.
- To delete the imported Node.js package, click **Delete Package**. Replacing the package will erase the existing package and will remove the links to the associated application.

> ***Note:*** If a service is published or part of a published app, you can delete that service only after you unlink the service from all the published app.

7. Click the **PUBLISH PACKAGE** button to publish the package to Node.js Runtime server.

   The **Node.js Package > Service Publish** section appear with the list of Node.js Runtime servers configured for the Kony Fabric account. The list also displays the following Node.js package status for that Node.js Runtime Server environment.

   - **Published**: A Node.js package is published to a Node.js Runtime Server environment. You can unpublish the Node.js package, if required.
   - **Not Published**: A Node.js package is not published to a Node.js Runtime Server environment. You can publish the Node.js package, if required.
   - **Failed**: A Node.js package is canceled while publishing or unpublishing. You can publish or unpublish the Node.js package, if required.



8. Select the environment.
9. Click the **PUBLISH** button. The process of uploading the Node.js package to Node.js Runtime server begins.

   > ***Note:*** The **PUBLISH** button dims when you have not selected any Node.js environment. When an environment is selected, only then the **PUBLISH** button is available.

   After the Node.js package is published to a Node.js Runtime Server, you can link the Node.js services to apps through the **Apps > Logic** tab. For details, refer to How to Integrate Node.js Services into Kony Fabric Apps.

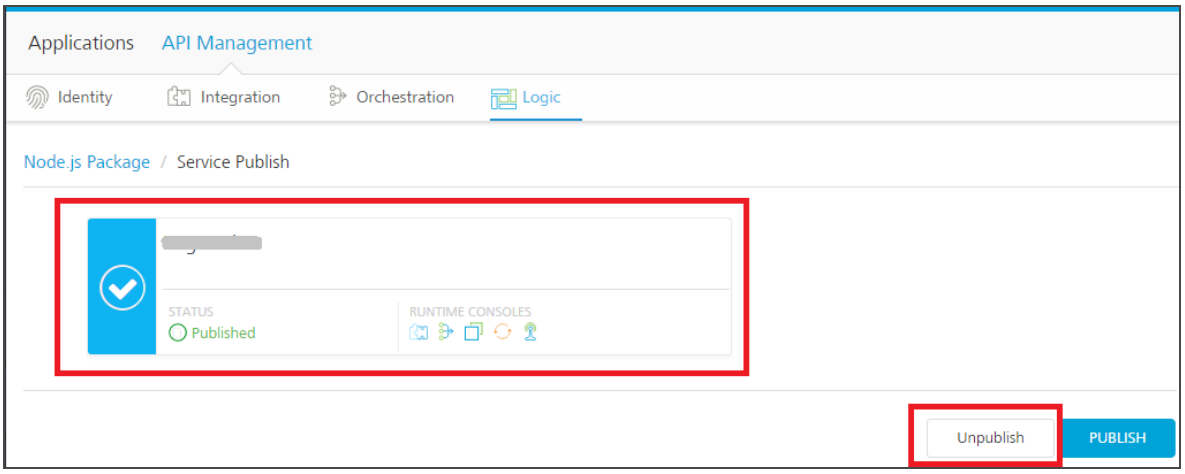
12.9.9.2 How to Unpublish a Package from Node.js Runtime Server

After a Node.js package is published to Node.js Runtime Server, you can unpublish the package if required.

To unpublish a package from m Node.js Runtime Server, follow these steps:

1. In the **API Management** page, click the **Logic** tab.
2. Click **PUBLISH PACKAGE**. The **Node.js Package > Service Publish** section appears with the list of Node.js Runtime servers configured for the Kony Fabric account . The list also displays the following Node.js package status for that Node.js Runtime Server environment.
   - **Published**: A Node.js package is published to a Node.js Runtime Server environment. You can unpublish the Node.js package, if required.
   - **Not Published**: A Node.js package is not published to a Node.js Runtime Server environment. You can publish the Node.js package, if required.
   - **Failed**: A Node.js package is canceled while publishing or unpublishing. You can publish

or unpublish the Node.js package, if required.



3. Under the **Node.js Package > Service Publish** section, select an environment.

> ***Note:*** The **UNPUBLISH** button dims when you have not selected any Node.js environment. When a published environment is selected, only then the **UNPUBLISH** button is available.

4. Click the **UNPUBLISH** button. The process of unpublishing the Node.js package from the Node.js Runtime server begins.

> ***Note:*** If a service is published or part of a published app, you can unpublish that package only after you unlink the service from all the published app.

12.9.9.3 How to Integrate Node.js Services into Kony Fabric Apps

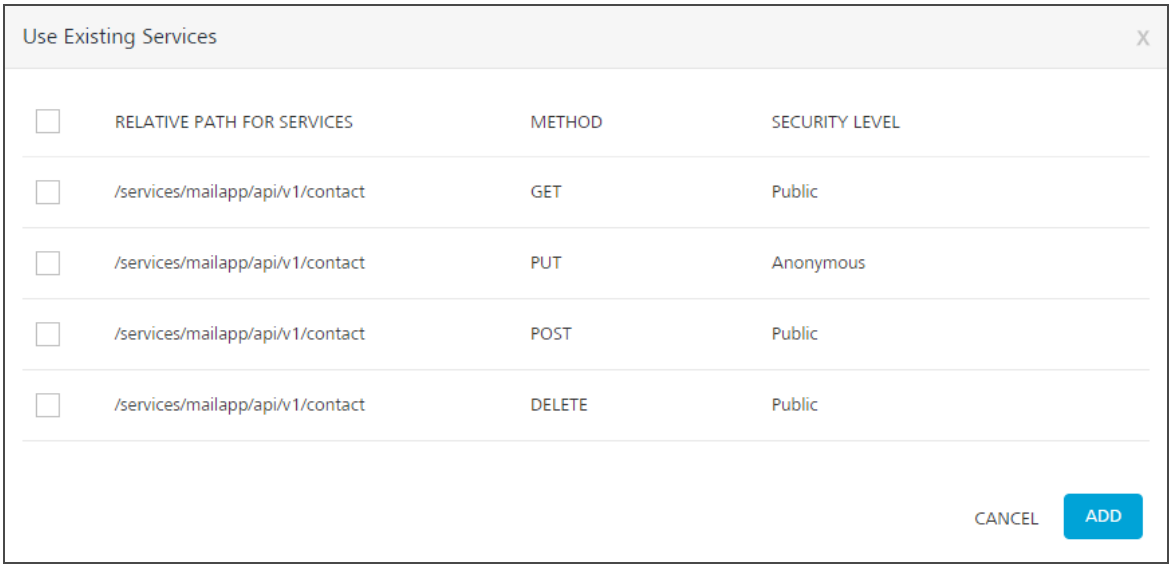
After you import a Node.js package into Kony Fabric and publish the package to the Node.js Runtime Server in the API Management > Logic tab, you can now integrate the Node.js services to Kony Fabric apps.

To integrate Node.js services into Kony Fabric app, follow these steps:

1. After you create an application, in the **Configure Services** tab, click the **Logic** services tab.
2. Click **Use Existing**.

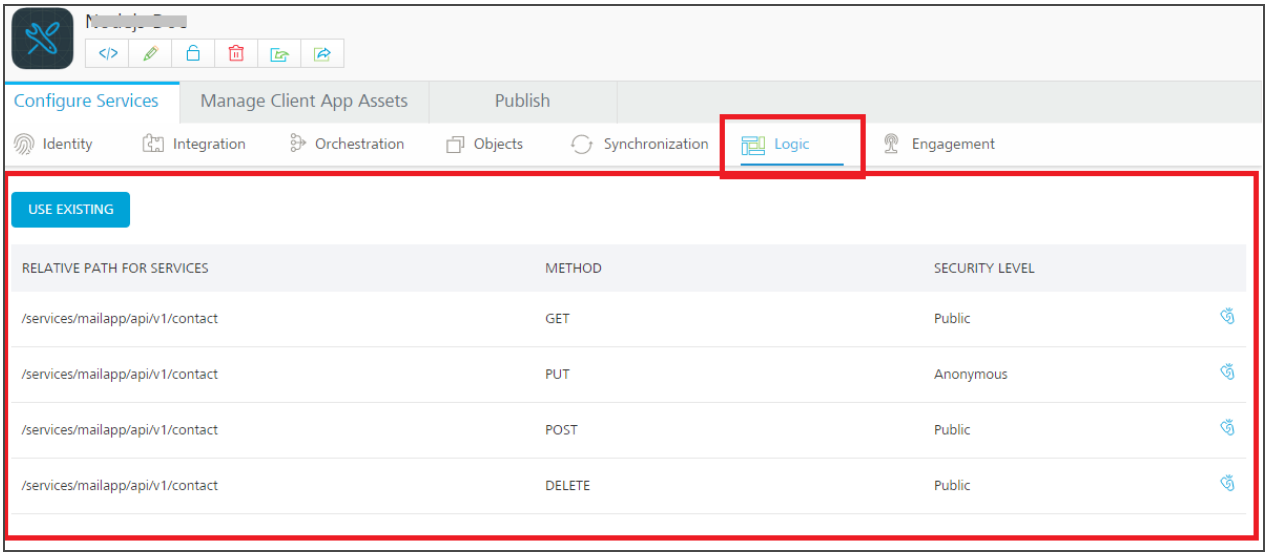
   The **Use Existing Services** screen appears with the list of services that you imported from the Node.js package.
   The following fields are displayed for the imported services.

   - **RELATIVE PATH FOR SERVICES**:
   - **METHOD:**
   - **SECURITY LEVEL**: Displays the security level set for this service. You can change the security level in **API Management** > **Logic** tab, if required.



3. Select check box for required services and then click **ADD**. After the services are added into Kony Fabric app successfully, the services are listed in the **Logic** page of your app.

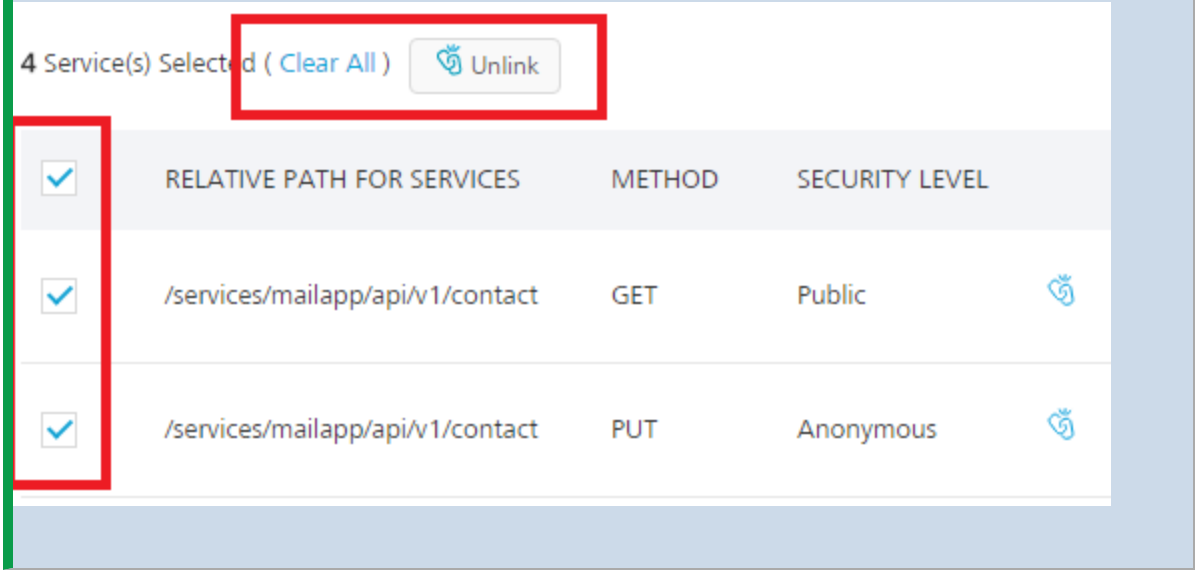
> ***Note:*** If a service is part of a published app, you can delete that service only after you unlink the service from all the published apps.



Unlink: Allows you remove the service from the **Logic** list page of an app. When a service is unlinked, it is disassociated from a particular app.

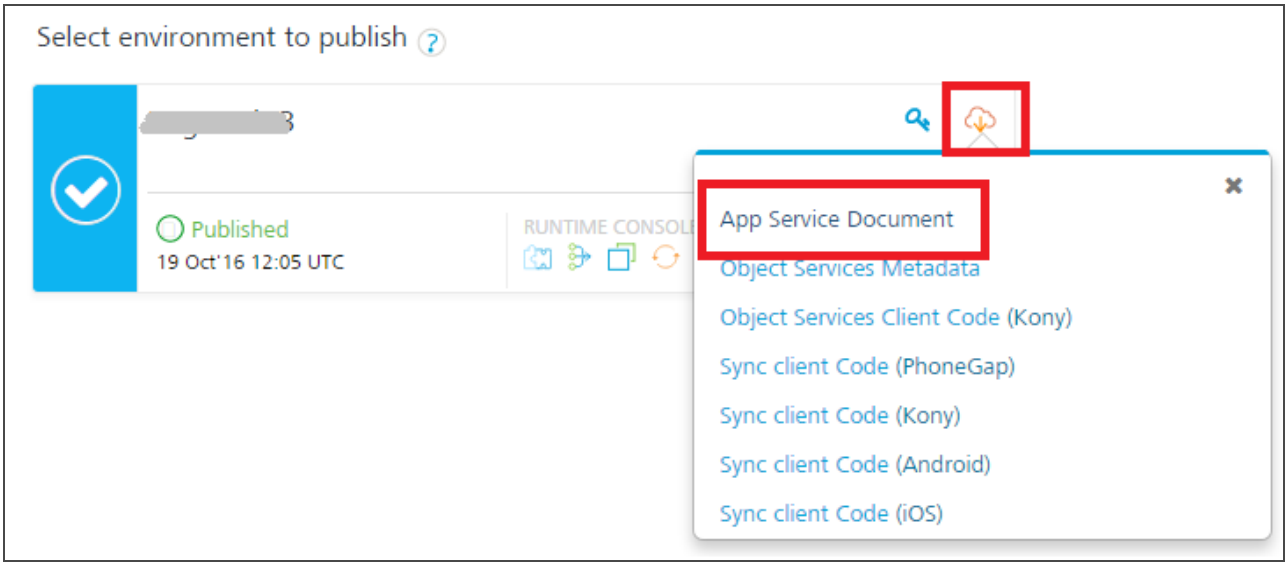
> ***Note:*** You can also unlink multiple services from the **Logic** list page. To unlink multiple services, select the required check boxes. The quick access bar for the selected services appears with actions (such as Clear All and Unlink). Click **Unlink**.
>
> - **Clear All**: Allows you to clear the one or more check boxes for the services.

Unlink: Allows you remove the service from the **Logic** list of an app. When a service is unlinked, it is disassociated from a particular app.



4. Publish the app. For information about publishing a Kony Fabric app, refer to Publish.

   Once the app is published, you can view the app publish details by clicking the **Download** button > **App Service Document** link.



The **App Service Document** dialog appears with logic services.

12.10 Publishing Individual Services

Kony Fabric supports editing and republishing service APIs for integration and objects services for published apps. With the **Publish Service** functionality in Kony Fabric, you (Kony Fabric Users/APIs Admins) can modify a service API of an integration/objects service for **published apps**, and then publish the service again. When you republish a modified service, Kony Fabric publishes only the modified service API to the apps run-time environment. Then published apps with latest services are available to users when they log in to apps/launch apps on their devices. You can use the Publish Service functionality from the API Management page.

12.10.1 Benefits of Publishing Individual Services

- Publishes service APIs quickly to published apps on run-time environment.
- Saves your time by publishing only modified service APIs of services, without requiring you to republish the entire app.

12.10.2 Limitations of Publishing Individual Services

The following service types are not supported for publish individual services:

- **Orchestration and Service-Driven Object** services are not supported for publish services because of dependencies.
- **Storage services are not supported**.
- **Identity handling**: The service will be published with the relevant identity linkage, but the onus of publishing the identity service itself is on App. And will not be handled through service publish.

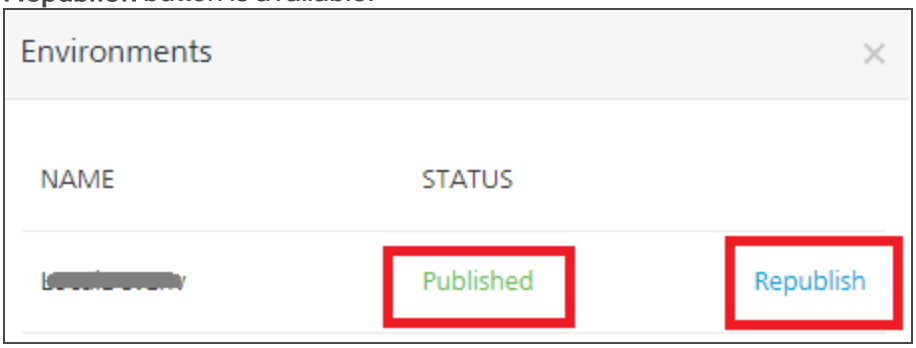
To publish an individual service to published apps on run-time environment, follow these steps:

1. In the **API Management** page, open the Integration Services or Objects Services listing page.
2. In the services pane, click the cogwheel button next to the service that you modified and want to publish, and then select **Publish Service**.

   The **Environments** dialog appears and displays the published environments names, and the service publish status.

   - If the service is not part of a published app, the status shows **Not Published**.

- If the service is part of a published app or apps, the status shows **Published**. And, the **Republish** button is available.



> ***Note:*** You can publish a service again only if it is published.
> While publishing a service, the **Republish** button is available only if the service is part of a published app or apps to environments.

3. In the **Environments** dialog, click **Republish**.

   After the services is published to the environment, the system displays the status.

   The following statuses are possible:

   - **Published**. The service is successfully published to the environment.
   - **Failed**: The publish is failed due to a network/internal error. You can click **Show details** to view the error details. You can retry publish the service by clicking **Retry**.
   - If an integration service has reference of an identity service, and when you republish the integration service, Kony Fabric displays the message for your confirmation. Click **OK** to authenticate and proceed.

13. Kony Fabric API Versioning

Kony Fabric supports versioning for integration services, orchestration services, and object services. Versioning is the process of assigning unique version numbers to unique states of software. Within a given version number category (major, minor), these numbers are generally assigned in increasing order and correspond to new developments in the software. API Versioning is useful for handling changes to an API. For example, additions and changes to an API that cannot cause an error in the existing API can be handled within one version. If there are additions and changes to an API that can result in errors or compatibility issues, the changes should be introduced in a new version of the API.

In runtime, you can run different versions of the same service. A Kony Fabric app published with two different versions of the same service is treated as two separate Kony Fabric apps. A new App key and App secret are generated when you publish the Kony Fabric app associated with a different version of the same service. The mobile app developer uses the key/secret to invoke the related services.

The version of a service is not invoked when the service is called through the Kony Fabric SDK . From the SDK, the name of the service is called, not the version of the service. Kony Fabric invokes the latest version of the service.

13.1 API Versioning Use Cases

The following use case describes the conditions and scenario for using API Versioning in Kony Fabric.

Scenario: Versioning of Integration Service

A Kony Fabric app named WeatherAppA uses an integration service named Weather. There is one version of the Weather integration service, version 1.0. The Kony Fabric app that uses the Weather integration service is deployed to production.

A bug is discovered in the Weather.getForecast operation of the Weather integration service. The operation uses Fahrenheit values for Temperature instead of Celsius, a requirement of WeatherAppA. There are currently no users of WeatherAppA. The Kony Fabric developer fixes the defect in version 1.0 of the Weather service, saves it, and redeploys WeatherAppA. Versioning may not be needed in this case.

In a different scenario, the Kony Fabric apps WeatherAppA and WeatherAppB use the Weather integration service. There is one version of the Weather service, version 1.0. The Weather service uses Fahrenheit values for Temperature.

The WeatherAppA developer requires Celsius and has requested a change to the Weather service. WeatherAppB is working as expected and shows the Fahrenheit value. The back-end developer configures WeatherAppA and edits the Weather service. The back-end developer saves a new version of Weather service, version 1.1, makes the change to select Celsius, saves the service, and then republishes WeatherAppA.

At this point, all existing instances of WeatherAppA invocations to Weather.getForecast start getting responses from version 1.1. WeatherAppB continues to get responses from version 1.0 of the Weather service.

13.2 API Versioning in Kony Fabric Services

You can save an integration service, orchestration service, object service as a new version of the service. When you save a service as a new version, Kony Fabric unlinks the latest version of the service from the Kony Fabric application, and links the new version. When you create a new service, Kony Fabric creates the new service as version 1.0.

You can use *major.minor* numbering for versions. Kony Fabric supports major versions from 1 through 999, and minor versions from 0 through 99. Kony Fabric supports version numbers from 1.0 to 999.99. Note that the dot '.' is a separator for version numbers and does not function as a decimal.

A Kony Fabric app can be associated with only one version of a service. When you associate an existing version of a service with a Kony Fabric application, Kony Fabric unlinks the latest version of the service from the Kony Fabric application, and links the version that you selected.

For more details, refer to How to Save or Use a Version of an Integration Service.

An orchestration service can reference other integration services or orchestration services. Therefore an Orchestration service can reference only one version of a specific integration service or orchestration service.

For more details, refer to How to Save or Use a Version of an Orchestration Service.

An Object service that is service-driven can reference only one version of a specific integration or orchestration service. A service-driven object is defined from existing APIs that use XML, Salesforce, REST, SOAP, JSON, and Java web services.

For more details, refer to How to Save or Use a Version of an Object Service.

13.3 API Versioning in API Management

In Kony Fabric API Management, Kony Fabric supports versioning for Integration services, orchestration services, and object services. You can select an existing version of a service, or you can save a new version of the service.

If multiple versions of a service are available in API Management, you can select the version of the service that you want to use. You can then link the version of the service to the Kony Fabric application through the Existing Services dialog in integration services, orchestration services, or object services.

In API Management, you can delete all versions of an integration service, orchestration service, or object service, or you can delete an individual version of a service. For more details, refer to Context based Settings.

13.4 Exporting a Versioned Service

The folder structure of an exported app (a .zip file) has folders, files, and certificates configured for that app. The logical flow of an exported app folder structure has four levels of folders. The primary, or root, level is the Apps folder, which contains all sublevel folders including files and metadata. The second level contains sections for integration services, orchestration services, and object services.

The integration section of an exported app has the referenced (shared) integration services configured for the app. Each version of an integration service configured for the app has a folder under the integration service folder. Each version of the integration service is self-contained in its own folder.

The orchestration section of an exported app has the referenced (shared) integration services configured for the app. Each version of an orchestration service configured for the app has a folder under the orchestration service folder. Each version of an orchestration service is self-contained in its own folder.

The object section of an exported app has the referenced (shared) object services configured for the app. Each version of an object service configured for the app has a folder under the object service folder. Each version of an object service is self-contained in its own folder.

13.5 API Versioning Compatibility

API Versioning is backward compatible with previous versions of Kony Fabric.

- If you import a Kony Fabric app that was built using MobileFabric 6.5 or an earlier version, Kony Fabric creates the default version 1.0 for the integration services, orchestration services, and object services in the imported app.
- If you upgrade to MobileFabric 7.0 from MobileFabric 6.5 or an earlier version, Kony Fabric creates the default version 1.0 for the integration services and orchestration services in the upgraded Kony Fabric apps.

14. Kony Fabric Application Versioning

Problem statement:

Consider a scenario where an Enterprise has a mobile application (MA) and a Kony Fabric counterpart (KF/MF) that is live and in use. The next updated version of the Kony Fabric app needs to be deployed and tested in Production and then released publicly. However, the Enterprise must meet the following requirements during this process:

- The publicly available mobile application should have no downtime and should continue to work without any interference or updates related to the new version.
- Once the testing is completed on the new Kony Fabric version, you should be able to switch to the new app version without requiring additional publish and/or modifications to the mobile application.

Solution:

App Versioning is a feature of Kony Fabric that allows you to create multiple versions of a Kony Fabric application, which can be developed and deployed at the same time. And provides a mechanism to configure the default version that you can bind with the Mobile app.

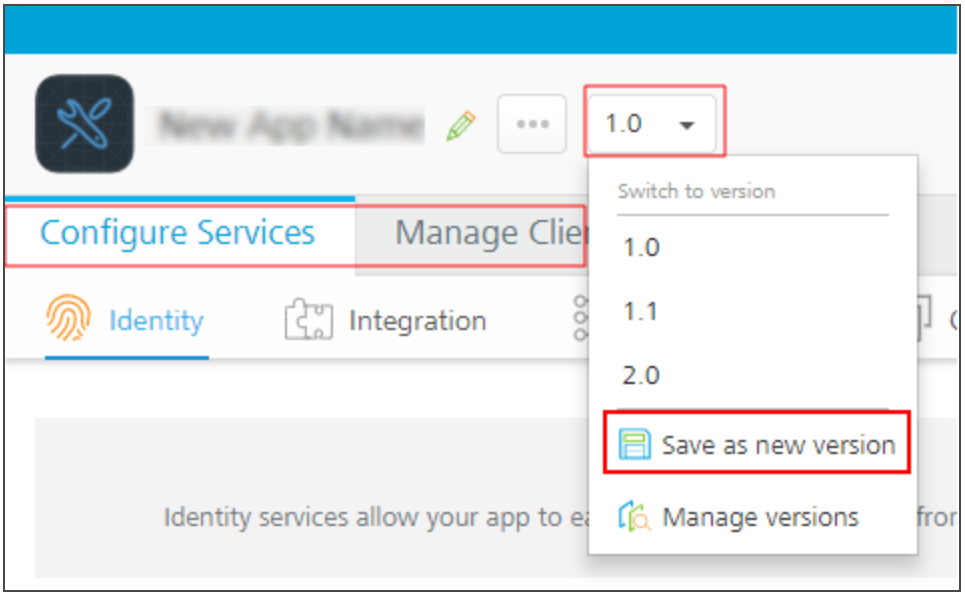
> ***Important:*** App Versioning is not supported for Kony Fabric apps using Engagement or Sync services.

The app service option is available across all the stages of the app creation in Kony Fabric Console such as **Configure Services**, **Manage Client App Assets**, and **Publish** tabs. For the first time created apps, version number 1.0 is added to the app by default. For the existing apps, you can change the version by saving the app with another version.

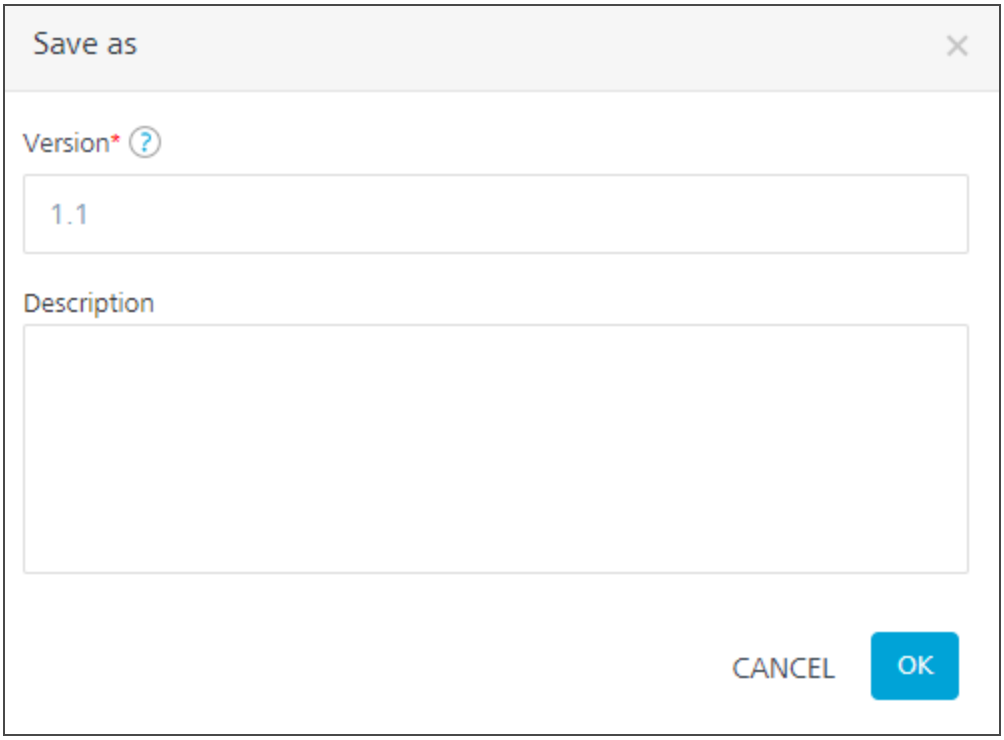
14.0.1 How to change the existing app version

You can change the current app version to a new version by saving the app with another version number.

1. Click an app. Your current app version is selected in the version list.
2. In the app edit mode (Configure Services, Manage Client App Assets, and Publish), next to the app name, click the version list and select **Save as new version**.



The **Save as** dialog box appears.



3. In the **Save as** dialog box, do the following:
    a. Under **Version**, choose the version number.
    b. Provide the appropriate description for the app.
    c. Click **OK**.

    The new app version is added to the versions list and the latest version is selected (active) in your app edit mode.

14.0.2 How to Manage App Versions

You can edit the description of apps and delete one or more versions of an application.

1. In the app edit mode (Configure Services, Manage Client App Assets, and Publish), next to the app name, click **Manage Versions**. The **Manage versions** dialog lists versions of the app.

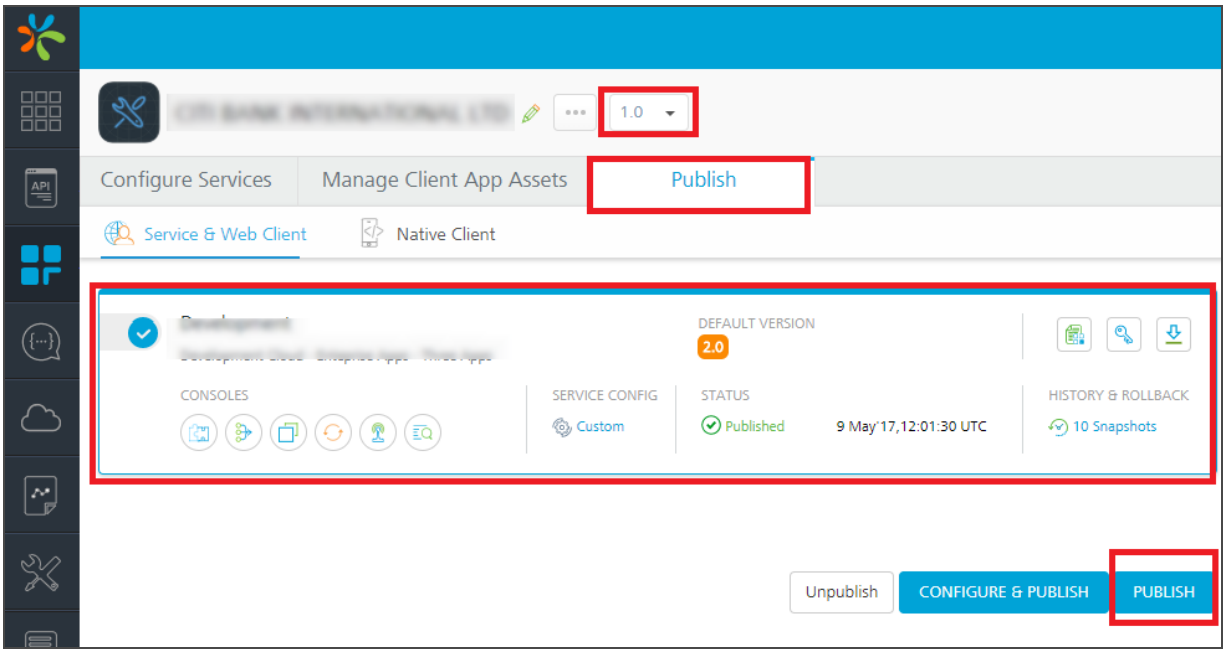
2. Hover your cursor over the required version, and click the **Edit** button. You can edit the description of the app.
3. Hover your cursor over the required version, and click the **Delete** button. When you click the **Delete** button, the **Delete App** dialog appears.
4. To confirm the deletion of the version, click **DELETE**. If the current version of the app is not published, the versions is deleted. Otherwise, the Error warns you that you are trying to delete the current version of the app that cannot be deleted as it is published.

14.0.3 How to Change the Default App Version Published at Runtime Server

By default, the latest version of an app is set for publishing. You can change the default version to another existing app version before publishing.

1. From the Applications page, select the desired app.
2. Click the **Publish** tab. By default, the **Service & Web Client** tab is selected and lists clouds or environments configured for the Kony Fabric account. The list also displays the app status for that cloud or environment. The default version of the app published to different environments is displayed.
3. Click an environment.
4. Select the desired version of the app from the version list.
5. Click **PUBLISH** to start the publishing. The process of publishing the app begins. Now, the version the you selected becomes the default app version for publishing. The selected app

version is published to run-time server.



14.1 When a default version of the app is unpublished, the following cases may arise

- If there is only one instance of the application in the runtime environment, the app is unpublished successfully.
- If there are two versions of the application in runtime, and the default instance is unpublished, the remaining one version should be made default.
- If there are multiple versions of the application in runtime, you can set one of the app versions as default by only viewing the list of published versions. Based on the list, you can choose the new default version at the time of un-publishing an app.

15. Custom Data Adapters on Kony Fabric

Kony Fabric Console supports the creation of custom data adapters. The custom data adapters uploaded into the Kony Fabric Console in API Management are supported in Integration and Object Services. An interface to upload, download, update, and delete data adapters is provided in the Kony Fabric Console. The data adapters are uploaded in a ZIP file format and stored in a workspace. Once the data adapters are uploaded successfully, you can create an integration or object service using the adapters .

Custom Data Adapters act as reusable services with a defined set of operations. These services work similar to any other service in the run-time environment.

15.1 When to use Custom Data Adapters

Consider the following scenarios in which you can use Custom Data Adapters.

Scenario: Adding an existing service definition.custom

Consider a scenario where you want to have a reusable service definition for your Kony Fabric applications, and you already have this definition available in a format compatible with Kony Fabric.

Instead of creating an API on Kony Fabric from a scratch, you can directly import your existing service definitions onto Kony Fabric as Custom Data Adapters. You can then use these Custom Data Adapters to create Integration or Object services.

Scenario: Creating a service by using a code written in RAML or Swagger

Consider a scenario where you have your API definitions in either RAML or Swagger, and this code exposes all the APIs that you want to use for your service.

Instead of creating an API with the service type as RAML or Swagger, you can directly upload the file to Kony Fabric and it will create a Custom Data Adapter for you. You can also add metadata for your service such as the Name, Version, and Logo.

15.2 Why use Custom Data Adapters?

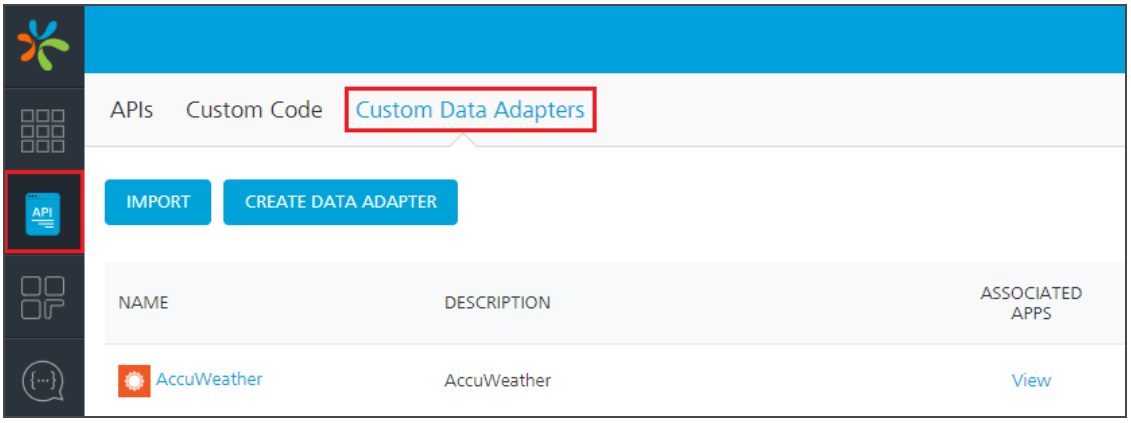
Custom Data Adapters are readily available as connectors. Moreover, you can add any number of services while creating a Custom Data Adapter.

You can also create a Custom Data Adapter directly from your backend API definitions if they are compatible with Kony Fabric. The compatible definitions can be in an XML file that you can structure according to the Custom Data Adapter Zip Structure.

You can also **upload** your Custom Data Adapters to the Kony Marketplace, so that other users will be able to use it.

15.3 Getting Started

You can configure Custom Data Adapters from the API Management section, under the **Custom Data Adapters** tab.



There are two types of Custom Data Adapters you can add to the Kony Fabric console.

- **RAML or Swagger Based** - You can use this type of Custom Data Adapter if your backend can expose its API definitions to Kony Fabric. These API definitions should be in RAML or in Swagger. You can add these Custom Data Adapters by clicking on the **CREATE CUSTOM DATA ADAPTER** button.

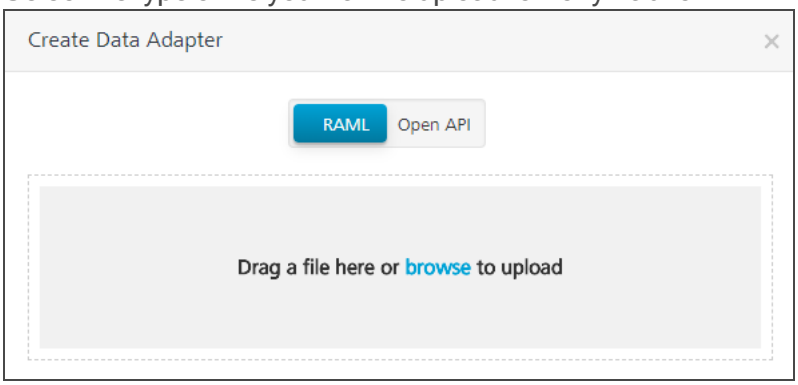
- **Kony Fabric App Based** - You can use this type of Custom Data Adapter if you have already configured any Kony Fabric services connected to your backend. These Custom Data Adapters should be in a structured zip archive. You can add these Custom Data Adapters by clicking on the **IMPORT** button.

15.4 Creating Custom Data Adapters

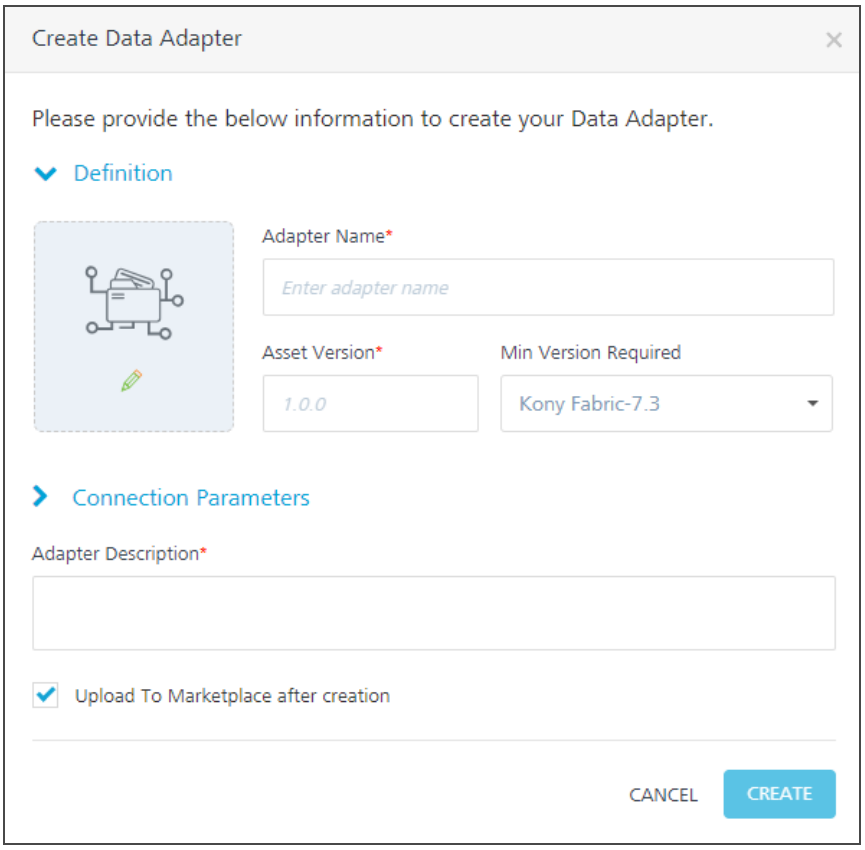
You can create Custom Data Adapters on Kony Fabric by uploading a RAML file or Swagger file. The file should contain all the API definitions that you want to expose to Kony Fabric.

To create a Custom Data Adapter, follow these steps:

1. Navigate to the **Custom Data Adapters** tab in your **API Management** section.
2. Click on the **CREATE CUSTOM DATA ADAPTER** button. The Create Data Adapter dialog appears.
3. Select the type of file you want to upload to Kony Fabric.

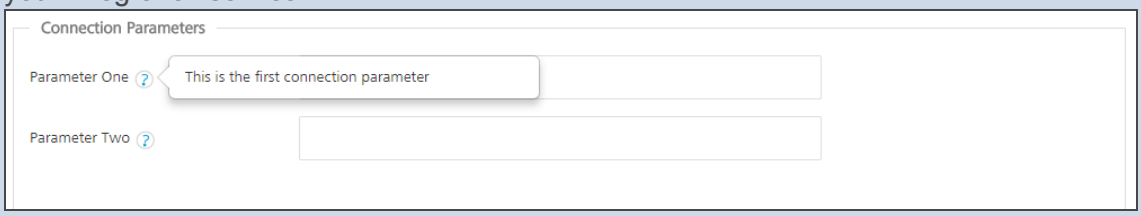


   - Select **RAML** if you want to upload a RAML file.
   - Select **Open API** if you want to upload a Swagger file.
4. **Drag and Drop** a file into the upload window.
   Alternatively, you can **browse** for a file on your system.
5. Enter the details for the Custom Data Adapter such as Adapter Name, Asset Version, and Description.



- **Adapter Name***[Mandatory]* - Name that Kony Fabric displays on the console.
- **Asset Version***[Mandatory]* - Version number that Kony Fabric displays on the console. The version number follows the **X.X.XX** format.
- **Min Version Required** - Minimum version of Kony Fabric that can run the Custom Data Adapter.
- **Image** - Image that Kony Fabric displays for the Custom Data Adapter's icon.
- **Adapter Description***[Mandatory]* - Description that Kony Fabric displays on the console.
- **Upload to Marketplace after creation** - Select this check box if you want to upload your Custom Data Adapter to the Kony Marketplace.

Note: **Connection Parameters** - You can configure dynamic connection parameters that you want to pass to the Custom Data Adapter.
These parameters show up when you select the Custom Data Adapter as a service type for your Integration service.



6. Click **CREATE**.

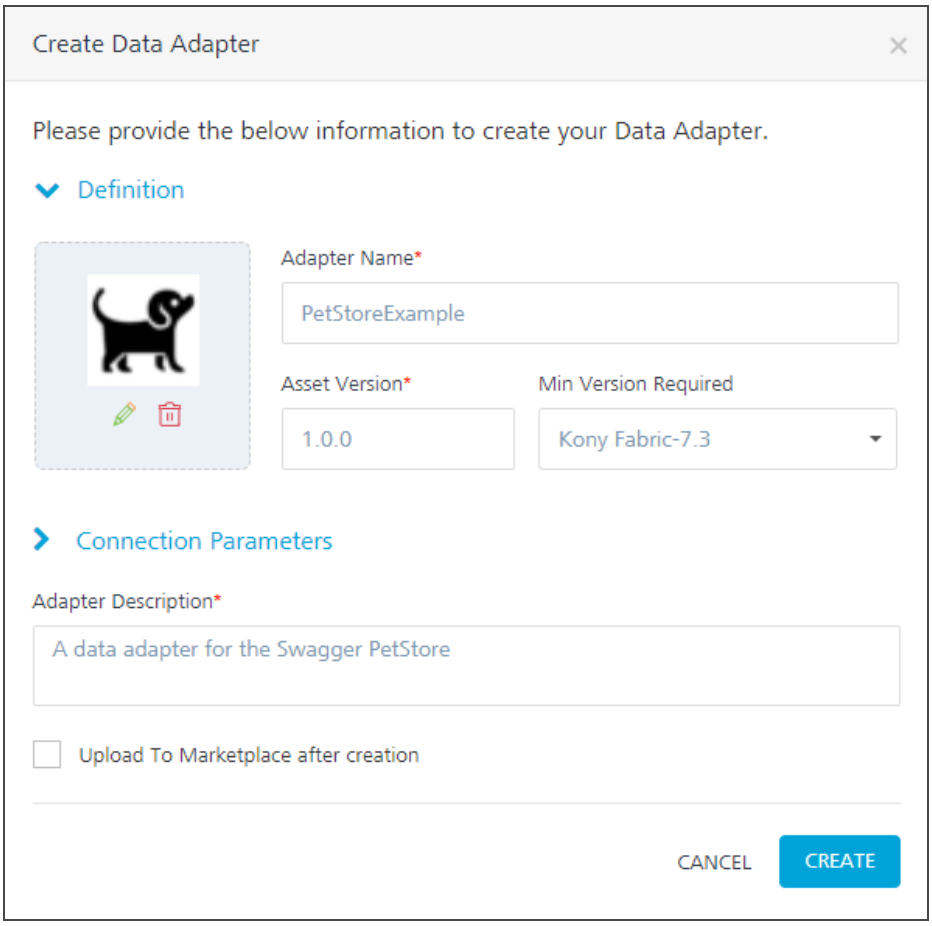
15.4.1 API Based Custom Data Adapter Example

Let us take a Swagger based example for a Custom Data Adapter that connects to the Swagger PetStore from Kony Fabric.

You can use a sample Swagger JSON for this example.

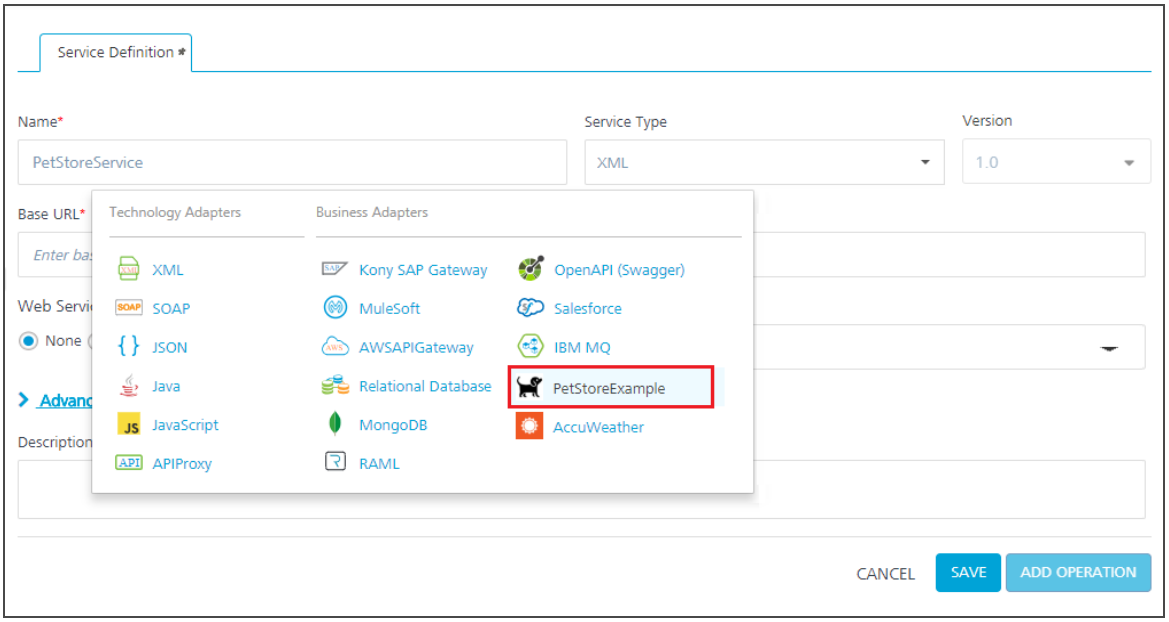
Follow the given steps to add the Swagger PetStore Custom Data Adapter.

1. Configure your Custom Data Adapter based on the steps given in the Creating a Custom Data Adapter section. Let us name the adapter **PetStoreExample**.

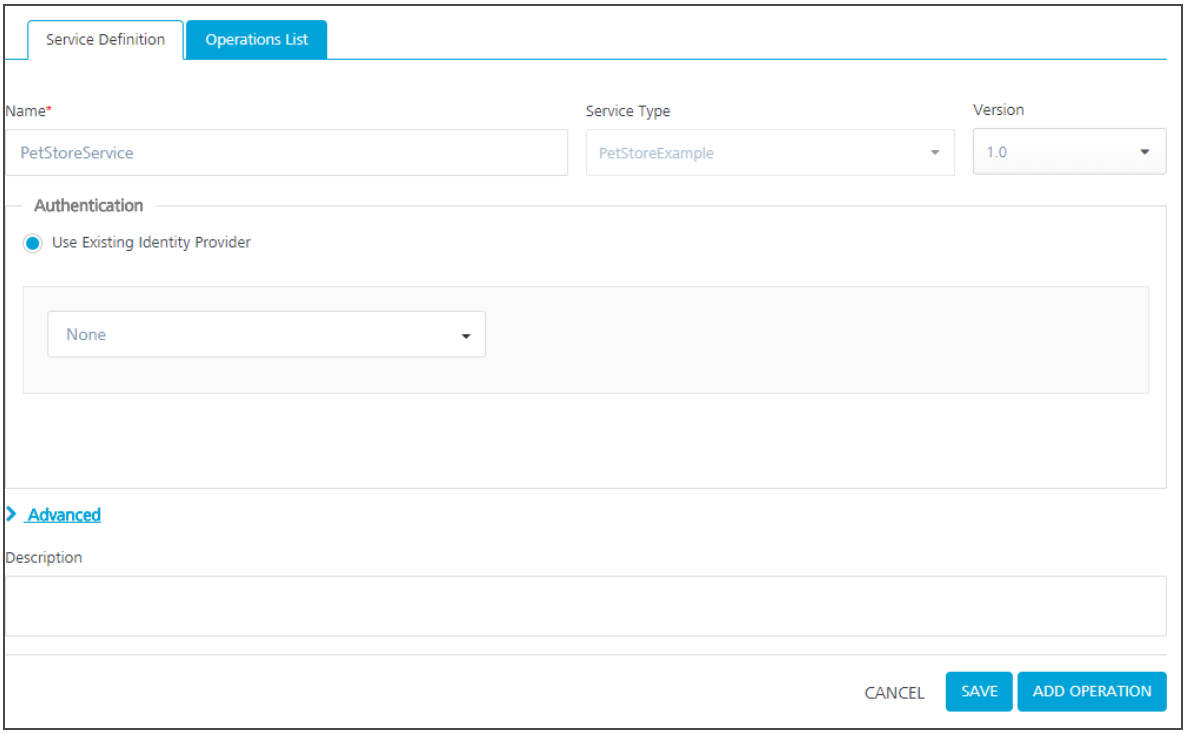


2. Create an Integration Service in your Kony Fabric application. Select your Custom Data Adapter in the **Service Type** list and click **Save**.
   As we named our adapter **PetStoreExample**, click on PetStoreExample to use it in your

service.

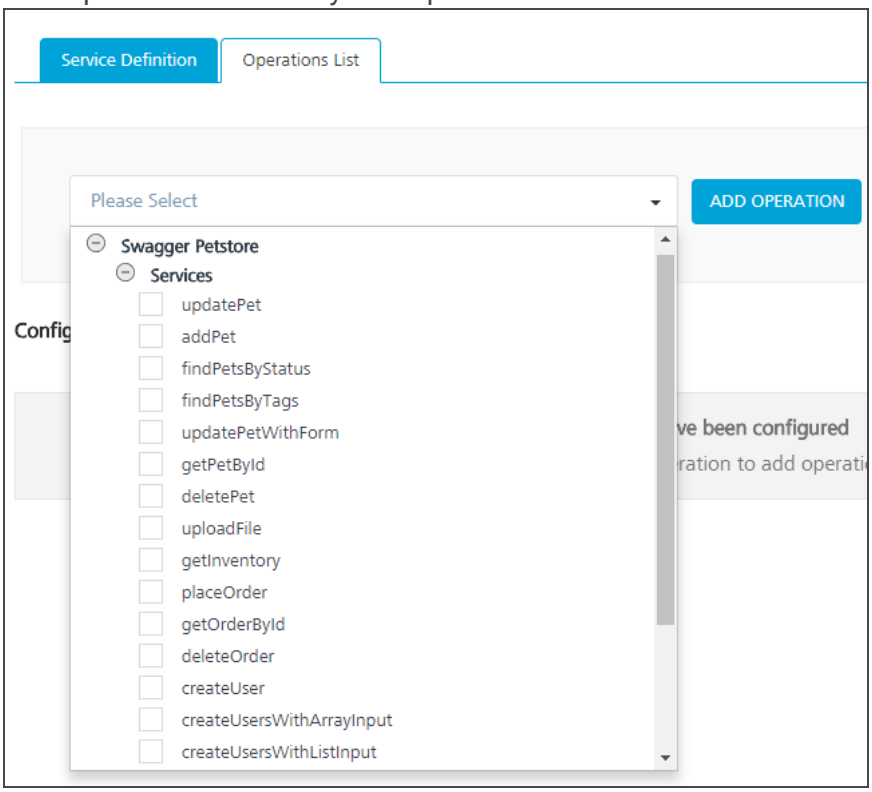


3. Click on **ADD OPERATION**.

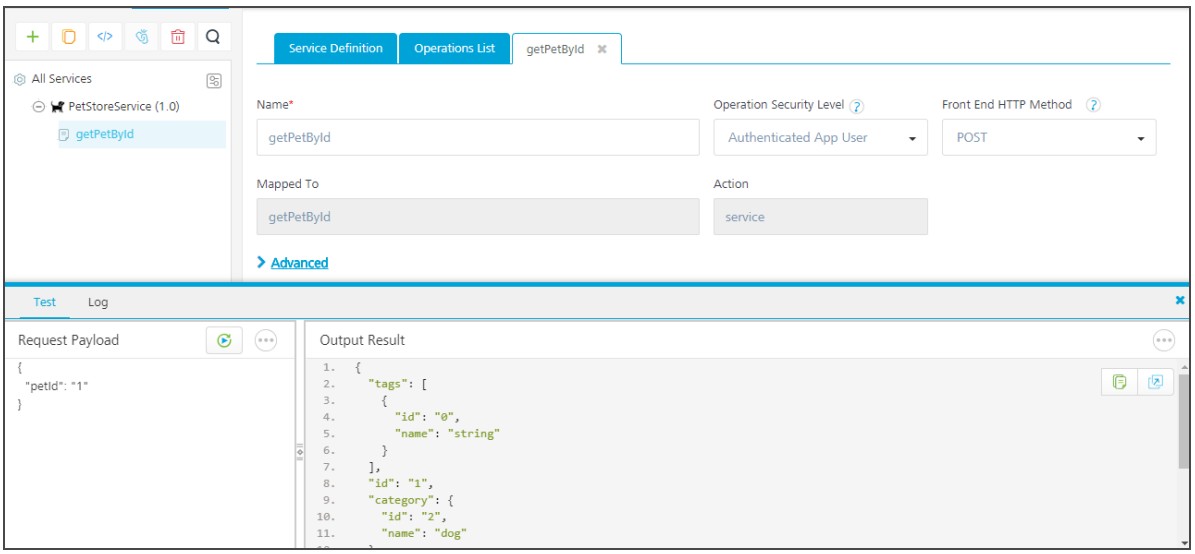


> ***Note:*** You can configure the **Authentication** details if you want to enable Enhanced Identity Filters for your Service.
> You can also configure the features in the **Advanced** tab if you want to add a custom code or enable throttling for your Service. For more information, you can refer to the Integration section of the Kony Fabric User Guide.

4. Add Operations based on your requirements.



You can then make service calls to the Swagger PetStore from Kony Fabric.



15.5 Importing A Custom Data Adapter

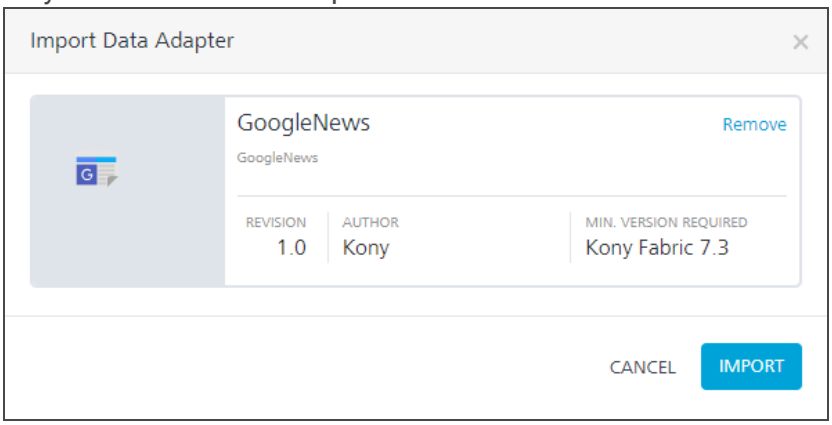
You can manually create a Custom Data Adapter with all the respective metadata on your system and import it into Kony Fabric. This Custom Data Adapter should be a **zip** file with all the service definitions in **XML** files and all the properties in **JSON** files. You can find more information in the Custom Data Adapter Zip Structure section.

Follow the given steps to import an existing Custom Data Adapter to Kony Fabric.

1. Navigate to the **Custom Data Adapters** tab in your **API Management** section.
2. Click on the **IMPORT** button.
3. **Drag and Drop** a zip file into the upload window.
   Alternatively, you can **browse** for a file on your system.

> ***Note:*** You can also import Custom Data Adapters from the Kony Marketplace by clicking the **IMPORT FROM KONY MARKETPLACE** button.

4. Click **IMPORT** on the next window. This window also shows the metadata that you configured for your Custom Data Adapter.



You can then use this Custom Data Adapter to create services and operations in your Kony Fabric Apps.

15.6 Custom Data Adapter Structure

The structure of your Custom Data Adapter depends on whether it is **RAML/Swagger** based, or **Fabric App** based.

15.6.1 RAML or Swagger Based Structure

If your Custom Data Adapter is based on RAML or Swagger, your zip file should contain the respective file at the root location. Let us take the same example of the Swagger PetStore Data Adapter.

You can use a sample PetStore.zip file for this example.

Ensure that the root location of your zip contains the following files.

- **properties.json** - This JSON file should contain the metadata about the Custom Data Adapter. For Example:

```
/* This file contains metadata about the Swagger PetStore Data Adapter */
```

```
{
       "createdBy" : "Kony",
       "propertiesVersion": "1.0",
       "itemName": "PetStore",
       "displayName": "PetStore",
       "description": "PetStore",
       "icon": "petstore.png",
       "assetVersion": "1.0",
       "createdMFVersion": "7.3",
       "adapterType":"swagger",

/* You can also configure dynamic connection properties for your data
adapter */
       "connectionManager":
       {
       "connectionProperties":
       [
       {
       "displayName": "Parameter One",
       "description": "This is the first connection parameter",
       "name": "ParamOne",
       "order": 0
       },
       {
       "displayName": "Parameter Two",
       "description": "This is the second connection parameter",
       "name": "ParamTwo",
       "order": 0
       }
       ]
}
}
```

Important:
The value for the **adapterType** key depends on the type of file you use.

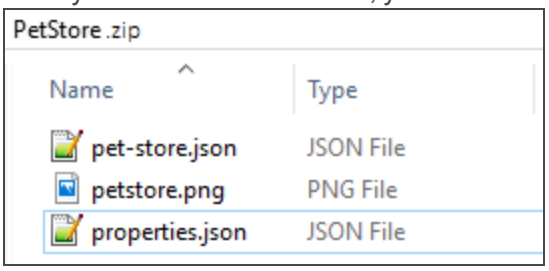
- **raml** - If the adapter is based on RAML.
- **swagger** - If the adapter is based on Swagger.

- **Image** *[Optional]* - You can include an image to use as the icon for the Custom Data Adapter.

> ***Note:*** You need to insert the name of the image under the **icon** key of the **properties.json** file to ensure that Kony Fabric uses the image as the logo.

- **RAML** or **Swagger** File - This file should contain all your API definitions.

After you create all these files, your folder structure should look similar to the given image.



You can then compress these files into a zip file and import it into Kony Fabric.

15.6.2 Kony Fabric App Based Structure

If your Custom Data Adapter is based on Service Definitions from Kony Fabric, you need to create folders for each service that you want to include. Let us take a sample **Google News Custom Data Adapter**.

You can use a sample Googlenews.zip file for this example.

The root location of the zip files should contain the following files.

- **properties.json** - This JSON file should contain the metadata about the Custom Data Adapter. For Example:

```
/* This file contains metadata about the Google News Data Adapter */
{
       "createdBy" : "Kony",
       "propertiesVersion": "1.0",
       "itemName": "GoogleNews",
       "displayName": "GoogleNews",
       "description": "GoogleNews",
       "icon": "icon.png",
       "assetVersion": "1.0",
       "createdMFVersion": "7.3",
       "adapterType":"mf_app",

/* You can also configure dynamic connection properties for your data
adapter */
       "connectionManager":
       {
       "connectionProperties":
       [
       {
               "displayName": "Parameter One",
               "description": "This is the first connection parameter",
               "name": "ParamOne",
               "order": 0
       },
       {
               "displayName": "Parameter Two",
               "description": "This is the second connection parameter",
               "name": "ParamTwo",
               "order": 0
       }
       ]
}
                                }
```

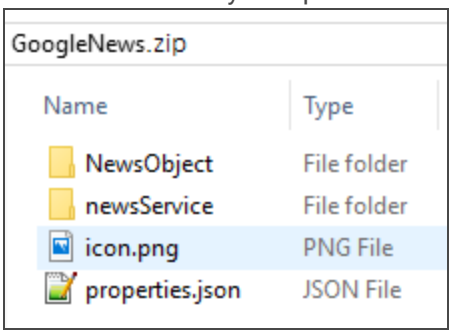
> ***Important:***
> The value for the **adapterType** key should be **mf_app**.

- **Image** *[Optional]* - You can include an image to use as the icon for the Custom Data Adapter.

  > ***Note:*** You need to insert the name of the image under the **icon** key of the **properties.json** file to ensure that Kony Fabric uses the image as the logo.

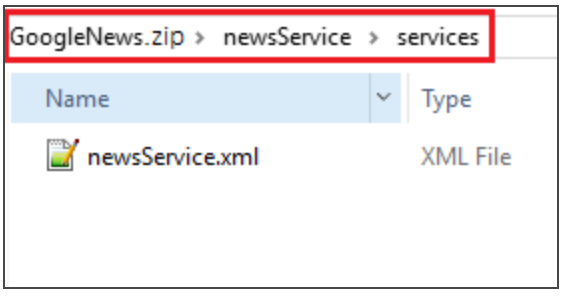
- **Folders** - You need to configure your services into sub-folders in your zip file's root location.

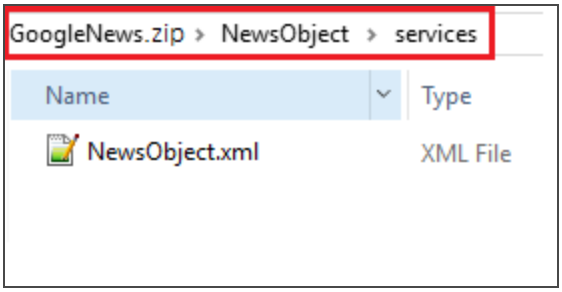
  The root location of your zip file should be similar to the given image.



  - Ensure that the name of the first level folder matches with your service name.
    In the given example, **NewsObject** is the name of the Object Service, and **newsService** is the name of the Integration Service.

  - The second level folder's name should be **services**. The folder should contain your service definition file.



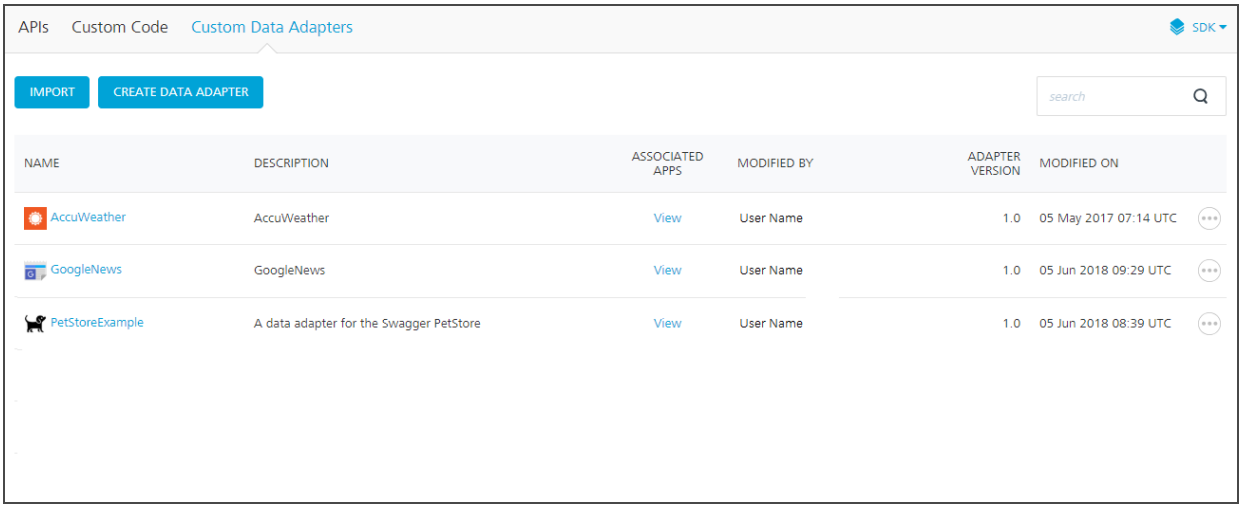


Once you've configured all the files and folders, you can compress it into a zip file and import it into Kony Fabric.

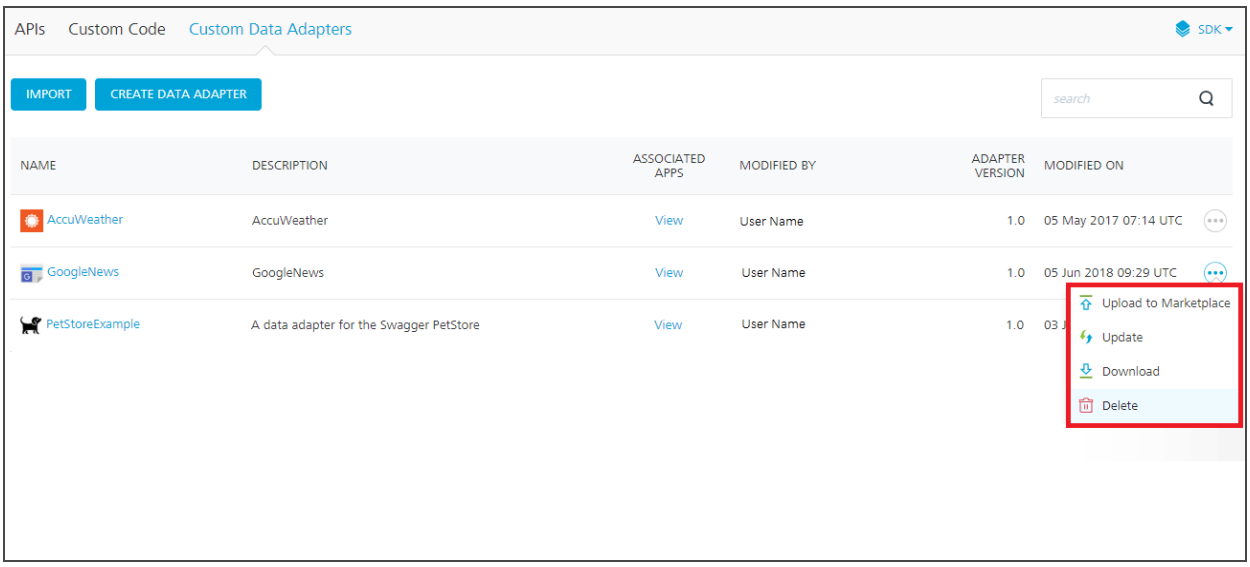
15.7 Managing Custom Data Adapters

You can manage your Custom Data Adapters from the **API Management** section under the **Custom Data Adapters** tab.

The Custom Data Adapters page on the console has a list of all the existing Custom Data Adapters and their metadata. You can view all the apps associated with any of the Custom Data Adapters. You can also view the **Modified On** and **Modified By** details on this page.



You have some additional options available under the **ellipsis** menu, that is the **three dots** menu on the right side, for every Custom Data Adapter.



- **Upload to Marketplace** - Publish the Custom Data Adapter to the Kony Marketplace.
- **Update** - Update the existing Custom Data Adapter. Clicking on this option will open a window to import a Custom Data Adapter.
- **Download** - Download the Custom Data Adapter as a **zip** file. The downloaded archive follows the structure mentioned in the Custom Data Adapter Zip Structure section of this document.
- **Delete** - Delete the Custom Data Adapter.

16. Managing JAR Files

This section describes procedures for managing JAR files in Kony Fabric. The following procedures explain how to import, update, and delete JAR files.

16.1 Importing a New JAR File

To import a new JAR file, do the following:

1. In **API Management**, click **Custom Code**.
2. Click the **Add New** button.
3. In the **Import JAR File** window, do either of the following:
   - Drag and drop the JAR file directly into the window.
   - Click on **Browse** to add the JAR from the file directory.

   > ***Important:*** Make sure that you upload a JAR file that is built using the same JDK version that you use for installing Kony Fabric.
4. Click to close the JAR Imported Successfully notification.

16.2 Updating a JAR File

To update a new version of an existing JAR file, do the following:

1. In **API Management**, under **Custom Code**, click on the JAR file that you want to update.
2. Click **Update**.

3. In the **Update JAR File** window, do either of the following:
    - Drag and drop the JAR file directly into the window.
    - Click on **Browse** , select the JAR from your file directory, and click **Open**.

    > ***Important:*** Make sure that you upload a JAR file that is built using the same JDK version that you use for installing Kony Fabric.

16.3 Deleting a JAR File

To delete a JAR file that is not needed, do the following:

1. In **API Management**, under **Custom Code**, click on the JAR file that you want to delete.
2. In **Associated Services**, if there are associated services listed, click on the service and then click **Unlink**.
3. After all associated services are removed, click **Delete** to remove the JAR file.

17. Analytics for Third-party Client App Binaries

With Kony Fabric Analytics, you can add instrumentation to your existing mobile app binaries without having to modify the source code and recompiling them. The Kony Fabric analytics library is added to your existing app binary and resigned with your certificate. You can then distribute the analytics enabled app binary to users knowing the functionality of the app is unchanged, but now the app will send usage data to the Kony Fabric backend that can be analyzed using the reporting capabilities within the Kony Fabric Console.

The MFCLI utility is enhanced to enable analytics for *third-party client app binaries*. The user can enable analytics to their apps by using the MFCLI analytics commands such as the wrap, wrap-fetch, and wrap-delete.

The `wrap` command adds the analytics capability into the third-party client app binaries by adding the Kony Fabric SDK. It also downloads a copy of the analytics enabled app binary to the local system and makes a copy of the analytics enabled app binary available in the **Kony Fabric Console > Apps** page. The Kony Fabric Admin can distribute the analytics enabled app binary to the targeted users, and the targeted users can install the binary and start using it. The supported platforms for analytics enabled binary are `iOS` and `Android`. The supported platforms' channels are `ios_phone`, `ios_tablet`, `android_phone`, and `android_tablet`.

17.1 Benefits of using MFCLI for Analytics

- Allows customers to get usage data on applications, for which they don't even have the source code.
- Saves the customer valuable time by injecting analytics into existing apps, without requiring a developer to write code to get the reporting functionality.
- It also allows users to distribute the analytics enabled app binary to the end-users either manually or using the console.
- Reports: The user can view the following standard reports from the *analytics enabled app*

binary:

- **Application Activity Reports**: This activity shows the application use based on application sessions per day, week, or month of a given date range.
- **User Activity Reports**: This report provides a count of new users for an application in a given date range and helps explore that data through charts.
- **Location Reports**: This activity shows user sessions based on geolocation per country on a map of the world.

  For more details on statistics, refer to Standard Reports.

17.2 Usage

17.2.1 Prerequisites for Kony Analytics Enabled App Binary

The following binaries and certificates are required to wrap third-party client binaries for Analytics.

- **client-binary**. The client binary to wrap an Android Package Kit (APK) file for an Android application. A .ipa (iOS application archive) file for an iOS application.

  > ***Note:*** By default Kony Fabric apps are enabled with analytics capability.

- **keystore**. The keystore file holds a certificate chain of which one will be used to sign the binary for distribution. For Android, this corresponds to the `.keystore` extension and for iOS, `.p12` files are expected. Keystore files may or may not use aliases to access certificates contained in the chain. Specifically, Android `.keystore` file requires aliases, while iOS `.p12` files do not.
- **keystore-passphrase.** The passphrase for opening/reading the keystore file as described above. This is applicable to both Android and iOS platforms.
- **provisioning-file.** The provisioning profile is specific to iOS platforms and is a `.mobileprovision` file containing a list of provisioned rights for the binary along with information about the distribution authority and the scope of targets, if applicable.

- **certificate-alias**. The alias by which the signing certificate is identified within the keystore chain, as described above. The certificate-alias applicable for Android only.
- **certificate-passphrase**. The passphrase that will be used to open/read the signing certificate specified by the alias above (if applicable) from the keystore file. This certificate contains signing information for the binary. This is necessary for the only Android platform since for iOS, the keystore passphrase suffices to open/read the certificate.
- *** --environment**. This environment is a cloud environment name. On this environment, app services and Management services should be running.

17.2.2 Download Links

Kony Fabric Command Line Utility can be downloaded from Kony download center.

17.2.3 MFCLI Commands for Analytics of Third-Party Apps (App Binaries)

The following are the analytics commands to wrap third-party apps' binaries.

17.2.3.1 Wrap Command

The `wrap` command helps the user to enable analytics into third-party client app binaries. When the user runs the wrap command through MFCLI, it uses the client app binary and app's platform specific certificates from the local system for wrapping. During the wrap process, the wrap command uses the client app certificates to inject analytics attributes into the client app binary.

For example, analytics attributes such as Kony Fabric-SDK, App Key and App Secret, and App Service URL. At the end of the successful wrapping, the app client binary gets bundled with the analytics attributes and downloaded to the MFCLI admin's directory. Now the client app binary is enabled with the analytics capability.

> ***Important:*** The analytics wrapping feature is supported with versions older than `SDK-GA-8.0.0`. If you use `SDK-GA-8.0.0` or higher versions of Kony Fabric SDK for analytics wrapping, the app will not work as expected after wrapping.
>
> Kony recommends that you use any of the Kony Fabric SDK versions less than or equal to `SDK-`

> `GA-7.3.0.17` for analytics wrapping available from Kony Download portal.
> While executing the **`wrap`** command from MFCLI, admin must pass the optional field value **`--sdk-version "x.x[.x][.x]"`**. Otherwise MFCLI consume the latest version of SDK by default.
>
> For example, if you want to specify the SDK version SDK-GA-7.3.0.17 for wrap command, pass the value as in this format: **`--sdk-version "7.3.0.17"`**

The following is a sample command to wrap an app binary to manage.kony.com

```
 java -jar mfcli.jar wrap -u <user> -p <password> -t <account id> -a
<app name> -e <environment name> -cb <client binary file> -plat
<client binary platform> -ks <keystore file> -kpp <keystore
passphrase> -cpp <certificate passphrase> -ca <certificate alias>

java -jar mfcli.jar wrap -u abc@kony.com -p password -t 100054321 -a
MyApp -e MyEnv -cb C:\temp\apk1.apk -plat android_phone -ks
C:\temp\MykeystoreFile.keystore -kpp MyKeystorePassphrase -cpp
MyCertificatePassphrase -ca  MyCertificateAlias
```

17.2.3.2 Wrap-fetch Command

The wrap-fetch command helps the user to download an analytics enabled app binary.

The following is a sample command to wrap-fetch an analytics enabled app binary from manage.kony.com

```
 java -jar mfcli.jar wrap-fetch -u <user> -p <password> -t <account
id> -a <app name> -e <environment name> -pl <platform Name> --binary-
version <app Binary Version>

java -jar mfcli.jar wrap-fetch -u abc@kony.com -p password -t
100054321 -a MyAppName -e My Env -plat android_phone --binary-version
1.0.0
```

17.2.3.3 Wrap-delete Command

The wrap-delete command helps the admin to delete the analytics enabled app binary. This command is applicable for Kony Cloud only.

An admin cannot delete an analytics enabled app binary from Kony Fabric Console if the binary is published on a **Web client** environment or **Native client** environment. Because the Console does not have an option to up-publish analytics enabled apps from the console.

The following is a sample command to delete an analytics enabled app binary from Kony Cloud

```
  java -jar mfcli.jar wrap-delete -u <user> -p <password> -t <account
id> -a <app name>

 java -jar mfcli.jar wrap-delete -u abc@kony.com -p password -t
100054321 -a MyAppName
```

> ***Note:*** When analytics enabled app binaries are published to Web client environments (app server), the app's published details are not displayed in the **App Publish** page in Console.

Therefore in MFCLI, a new command (wrap-delete) is added. The wrap-delete first unpublishes the analytics enabled app from all Web published environments and then deletes it from the console. If the analytics enabled app is published to any native client environments, when you try to delete the app, the following error appears:

```
C:\Users\Desktop\mfcli73GA\java -jar mfcli.jar wrap-delete -u
abc@kony.com -p pwd -t 100000008 -a SampleAnalytics --cloud-type
"sit2-"
Deleting analytics app [SampleApp] from Kony Fabric...
   Verifying app details...
   Verifying native client binary published info...
   App binary is published as native client on  :
https://gwlib.emm.sit2-konycloud.com
ERROR: The app binary with analytics is published on the above Kony
```

```
Management environments. Please unpublish and delete the app from all
Kony Management environments before wrap-delete.
```

> ***Note:*** To delete the analytics enabled app that is published on a native client environment, the admin must need to delete the app from that particular Management environment/native client environment. Once the app is deleted from all dependent native-client environments, the admin can re-run the wrap-delete command from MFCLI for deleting the app.

If the analytics app is not targeted to any of the Kony Management Environment, the wrap-delete command un-publishes and deletes the app from Kony Fabric Console without any error, shown below:

```
C:\Users\Desktop\mfcli73GA\java -jar mfcli.jar wrap-delete -u
abc@kony.com -p pwd -t 100000008 -a SampleAnalytics --cloud-type
"sit2-"
Deleting analytics app [SampleApp] from Kony Fabric...
  Verifying app details...
  Verifying native client binary published info...
  Verifying web client published info...
  Deleting analytics app [SampleApp] from Kony Cloud...
Successfully completed!
```

17.2.4 Attributes for Wrap, Wrap-fetch, and Wrap-delete Commands in MFCLI

The following are the attributes and commands for the wrap, wrap-fetch, and wrap-delete from the MFCLI.

> ***Note:*** Fields marked with an asterisk sign are mandatory.

Arguments	Platform
-t, --account	Nine-digit ID of the Kony Cloud account (visible in top right corner in Console) - for example, 100054321. Not relevant for an On-premise installation.
***-a, --app**	Name of the app to be published as part of the wrapping.
-k, --appkey	App Key to be used for the app. If the app key does not exist, the CLI/system generates it. This value makes sense only on a fresh publish and is ignored on a republish. Default: <empty string>
-s, --appsecret	App Secret to be used for the app. If the app secret does not exist, the CLI/system generates it. This value makes sense only on a fresh publish and is ignored on a republish. Default: <empty string>
--binary-version	The version of the client binary is in the format of x.x.x (for example 1.0.0). Default: 1.0.0
-ca, --certificate-alias	The certificate alias for verifying the Android certificate.
-cpp, --certificate-passphrase	The certificate passphrase for reading the Android certificate.
***-cb, --client-binary (.apk)**	Name of the client binary to wrap.
--description	Description of client binary. Default: Client binary for analytics wrapping.
***-e, --environment**	Name of the environment to publish to as part of wrapping.
***-ks, --keystore**	The keystore file for a certificate to use for wrapping.
***-kpp, --keystore-passphrase**	The passphrase for reading the keystore file.

Arguments	Platform
*** -p, --password**	Password for the Kony user. This could be plain text or, encrypted using 'encrypt' command
*** -plat, --platform**	The platform for the client binary. Possible Values: [android_phone, android_tablet]
-pf, --provisioning-file	iOS mobile provisioning file (for example, mobileprovision file)
--sdk-version	Kony Fabric SDK version to use for wrapping. Default: latest
*** -u, --user**	Kony user required for authentication - for example, abc@kony.com

17.2.5 Use Cases of Analytics through MFCLI

17.2.5.1 Use Cases for the Wrap Command

This section helps the user better understand the utility for Analytics.

For example, company MoBoCo has a native app binary for Android- for example, MoBoAnalytics.apk. The source code for the app is not available, but MoBoCo wants to enable analytics into the app binary and to reduce the effort to recode analytics logic. Finally, publishes the analytics enabled app binary to an enterprise distribution/app management environment, and end-users can install this app.

In this example, a user needs to perform the following two steps:

1. Wrap the client binary with analytics attributes using respective platform certificates.

   To wrap the app binary, run the following command from MFCLI.

```
 java -jar mfcli.jar wrap -u <user> -p <password> -t <account id>
-a <app name> -e <environment name> -cb <client binary file> -
```

```
plat <client binary platform> -ks <keystore file> -kpp <keystore
passphrase> -cpp <certificate passphrase> -ca <certificate alias>
```

The Kony Fabric app is created with the given name by the user. The analytics enabled app binary is stored in the Kony Fabric workspace. Kony Fabric Admin can access the analytics enabled app binary in **Kony Fabric Console > Apps** page - for example, `SampleAnalyticsWrapApp`. Now, the analytics enabled app binary can be distributed to users.

2. Make the analytics enabled app binary available to users in one of the ways mentioned below:
    - To distribute the analytics enabled app binary to users enrolled with Management Server, Kony Fabric Admin can use the **Kony Fabric Console > Publish > Native Client** feature to publish the analytics app to Kony Management environment as `Sign Only`. After successful publish of the analytics enabled app binary, the app gets published to the environment. Now, the targeted users can download and install the analytics enabled app on their devices using the Enterprise Store.
    - To distribute the analytics enabled app binary to users manually, a Kony Fabric Admin can share the analytics enabled app binary to the user. After that, the user installs the analytics enabled app binary on devices. For example, an Android user can use an enterprise specific distribution system to install the app on devices.
3. View the reports based on the usage of the apps. The Kony Fabric admin can view the statistics through the **Kony Fabric Console > Reports** tab. Currently, Standard Reports are supported for the analytics enabled app binary.

17.2.5.2 Use cases for the Wrap-fetch Command

- While the wrapping process was in progress, and if the user had skipped the wrap process wait time. The user can get the analytics enabled app binary. For example, while the wrap command is in progress and is taking longer time than expected, the user intentionally exits from the MFCLI. Later the user wants to get the wrapped client binary for the submitted app version and platform.
- To get the analytics enabled app binary from the workspace - for example, if the user deleted or

lost the analytics enabled app binary from the local system.

- To get the analytics enabled app binary while the wrap was in progress and if the Kony Fabric Admin gets disconnected from wrap process due to an unexpected error.

17.2.5.3 Use cases for the Wrap-delete Command

- When you no longer require to collect statistics of the analytics enabled app binary from devices, you can delete the app from Kony Cloud. When the app is deleted from Kony Cloud, the previous statistics are no longer available in the console. And no further statistics are captured even if the app is running on devices.

17.3 Limitations of MFCLI for Analytics

- Apps with the same bundle ID cannot be created in Kony Management Server. For example, the user has published a native app binary in Kony Management Server. When the user wraps the same app through the **MFCLI > wrap** command, and tries to publish the same binary (with bundle ID) through the **Kony Fabric Console > Publish > Native Client** feature to the same Management Server, the system throws publish error.
- The analytics enabled app exists in Kony Fabric in the **Published** state with Kony Fabric SDK wrapped into it. The binary is installed on devices. Now, if a Kony Fabric Admin unpublished the analytics enabled app from the environment, the statistics of the app usage cannot be reported.

18. Exporting and Importing an Application

18.1 Introduction

You can export apps from one workspace (Kony account) and import them to different workspaces of the Kony Fabric Console. An exported or imported app has services configured into it.

A Kony Fabric app comprises a group of services. They are:

- Non-shared services that cannot be shared with other apps, such as Kony Fabric Sync and Kony Fabric Messaging.
    - Kony Fabric Sync enables developers to add synchronization capabilities to mobile applications. Fundamental to Sync Framework is the ability to support offline and collaborative data between devices and the back-end systems.
    - Kony Fabric Messaging allows developers to upload push certificates for iOS, Android, BlackBerry, and Windows 8 RT platforms.
- Shared services that can be shared with other apps, such as custom code JAR files, integration services, and orchestration services.
    - The integration service of an application represents the application interaction with the external data source.
    - Service orchestration coordinates or integrates several services and exposes them as a single service.

> ***Important:*** Support for importing and exporting apps is available for identity services, such as Kony SAP, Kony Custom Identity, Salesforce, and Facebook.

18.2 Use Cases

You use exporting and importing apps based on the following scenarios:

- To move an app from one workspace (Kony account) to another workspace of the Kony Fabric installation. For example, a user completes the development of an app in a developer environment and later wants to move the app to a system integration testing (SIT) or user acceptance testing (UAT) workspace. A user exports an app from a developer environment and then imports the app into another workspace of the Kony Fabric installation. The user then moves the app to a production workspace.
- To merge changes made to an app in the repository (also known as check-in or commit) with the changes you have on your machine, such as Git source control management system.
  For example, a user exports an app from the Kony Fabric portal and merges the services of the app to the Git repository.

  > ***Important:*** To merge configuration changes made to an existing app to a source control system (for example, Git), you must export an updated app with the same details as the earlier version of the app in the source control system.

  > ***Important:*** You must republish the app for the new settings to take effect.

- To back up and restore (rollback) Kony Fabric apps. You can do backup of a Kony Fabric app package, and then you can restore the app to a previously defined state. You can restore a complete app or the services part of the app. For example, you can restore a complete app package to recover from an error.

18.3 How to Export an App

When an app is exported from a workspace, the exported app is saved with the same name of the app - for example, `ExportApp.zip`. An exported .zip file has an app's configured services information, such as icon files, certificates, .XML files, and meta files.

> ***Note:*** You cannot import an exported app after you modify the structure in the exported app. Support for importing an edited zip (exported app) file is not available. If you try to import an edited ZIP file, the system may fail to import the app successfully.

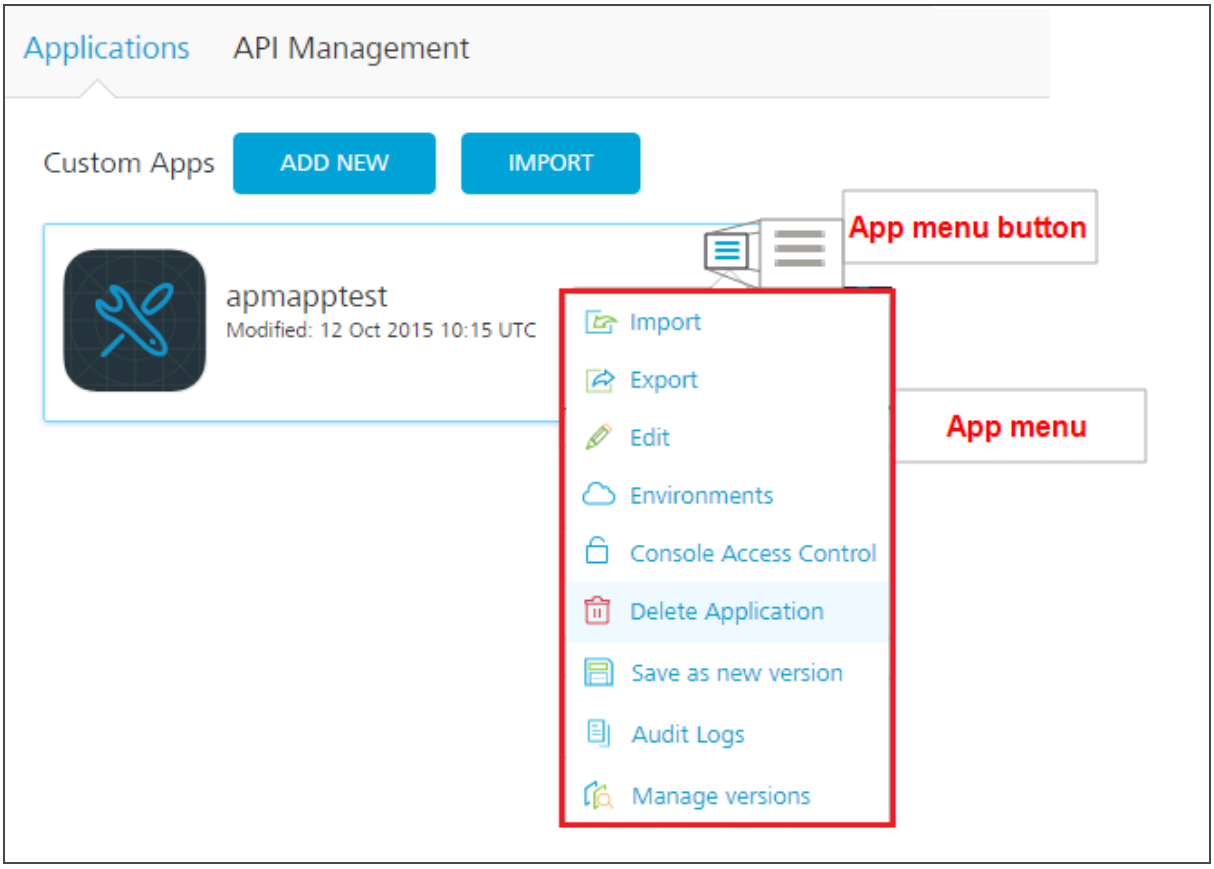
An exported zip file should have the correct folder structure. An exported zip file should have correct references in meta files. For more details about the folder structure of an exported app, refer to the Folder Structure of an Exported App section.

Important: Before exporting an app, do not unlink identity services that are referenced in the integration services of the app.
If you unlink a referenced identity service in the **Identity** tab and try to export an app, the system fails to export that app.

Important: Before exporting an app, do not unlink integration services that are referenced in the orchestration services of the app. If you unlink a referenced integration service and try to export an app, the system fails to export that app.

To export an app from a workspace (Kony account), follow these steps:

1. From the Kony Fabric Console, click **Apps** to display the Applications page.
2. In the **Applications** page, hover your cursor over the **App menu** button of one of the apps in the list. Click **Export**.



The system saves the app as `<AppName>.zip` in your browser's default download location.

> ***Important:*** You must republish the app for the new settings to take effect.

> ***Note:*** You can also export an App via API. For more details, refer to Continuous Integration - Export an app via API

18.4 How to Import an App as a New App

With importing an app as a new app, you can create new apps quickly by reusing configurations from existing apps. You save time because this method reduces the number of steps needed to re-create an app. After you import an app as a new app, you can modify configurations in the app as required.

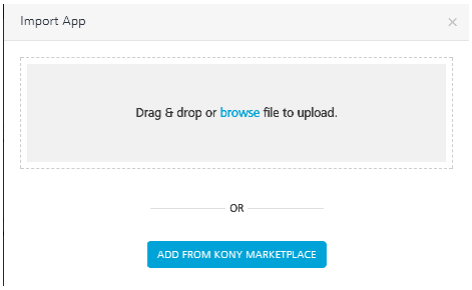
After an app is exported, you can import it as a new app or overwrite an existing app across various Kony Fabric Consoles. When you import an app as a new app, the system imports the app into the console. The imported app includes all data from the original app and the name of the app. The imported app is listed in the **Applications** page.

To import an app as a new app, follow these steps:

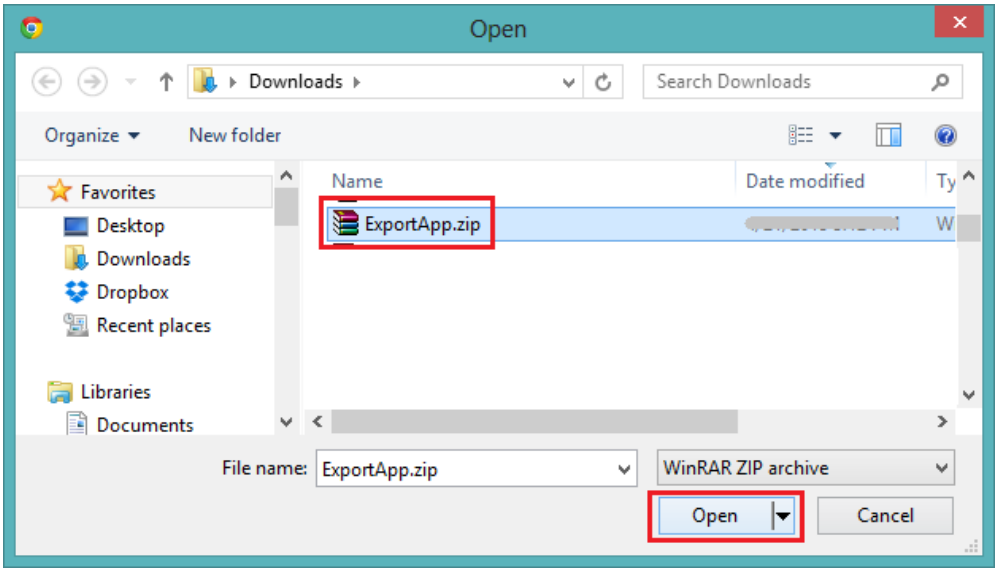
1. From the Kony Fabric Console, click **Apps**. The **Applications** page appears.
2. In the **Applications** page, click the **Import** button.

   The **Import App** dialog appears.

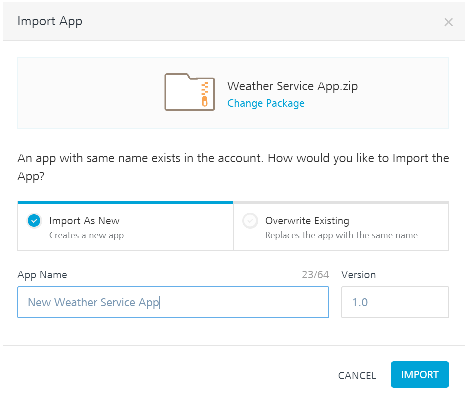
   

3. In the **Import App** dialog, do one of the following:
   - Drag and drop an app zip file into the dialog box.
   - Click **Browse** to select the app through the Microsoft Windows **Open** dialog box, click on the zip file of the app you want to import, and click **Open**.



- Click **Add from Kony Marketplace**. Select the app you want to import, and click **Import**.

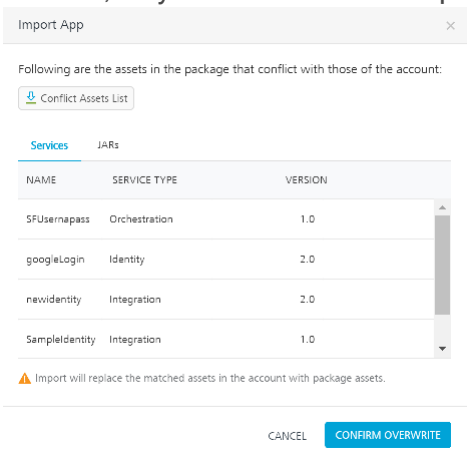
4. In the **Import App** dialog, the selected file is added under **Import Options > App Name**. To change the selected file, click **Change Binary**, select the app file you want, and click **Open**.



5. Under the **Import Options**, choose one of the following options.
   - Click **Import as New** to import the app as new. Click **Import**. The app is imported as a new app.

> ***Important:*** You must republish the app for the new settings to take effect.

- Click **Overwrite Existing** to overwrite an existing app, and then do the following:
    1. Click the **Select App** list, and select one of the existing apps from the list, and then click **Import**.
    2. If there are any associated JAR files or services in the app that will conflict with the account, they will be listed in the Import App dialog box.



        - If you want to download a file containing a list of the asset conflicts, click the **Conflict Assets List** button.
        - If you want to cancel the app import, click **Cancel**.
        - If you want to finish the import, click **Confirm Overwrite**.

> ***Important:*** While overwriting an app, if the app names are same, the new data will override the existing data.

Based on various services configured in an existing app, the system overwrites the existing data from a zip file. For example:

- While overwriting an app, if a provider in that Kony account exists with the importing identity provider name, the system fails to import the zip file.

- While overwriting an app, if the existing app has identity, integration, and orchestration services, these services will be unlinked from the existing app.
- While overwriting an app, if the names of the existing app's integration and orchestration services are the same as those in the zip file, these services will be updated.
- While overwriting an app, all non-shared services (synchronization and engagement) are overwritten into the existing app. The existing app will only contain new data. You cannot retrieve old data in the existing app.

> ***Note:*** You can also import an app via API. For more details, refer to Continuous Integration - Import an app via API

18.5 How to Import an App to an Existing App

You can update an existing app's configurations with the latest configurations made in another app in a different workspace. You can reuse the updated configurations from other apps to save time and development cost.

After an app is exported, you can import the app to an existing app in the Kony Fabric Console.

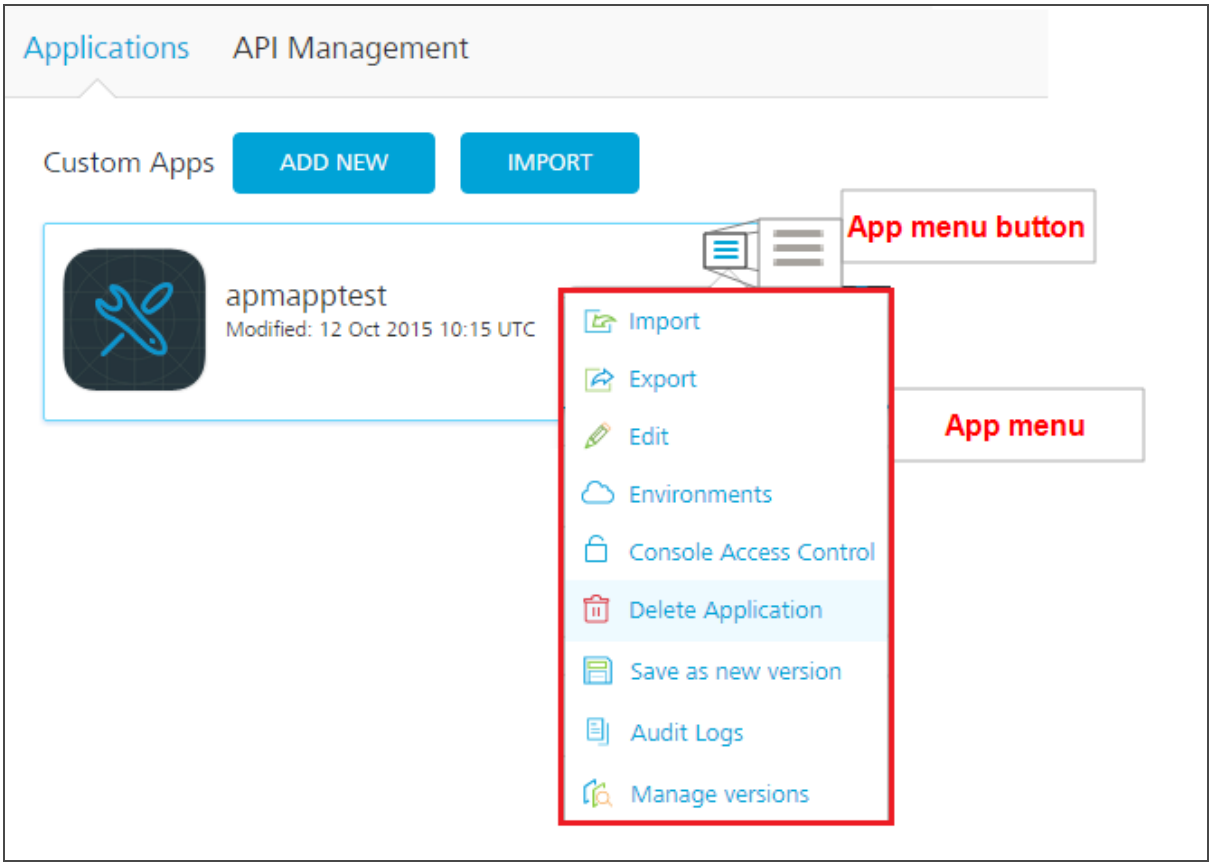
While importing an app to an existing app, if the app names are same, the system overrides the existing data with new data in the imported zip file. The app name will not be changed.

If the app names are different and you import an app, the existing app and data will be overwritten with the new app name and information in the zip file.

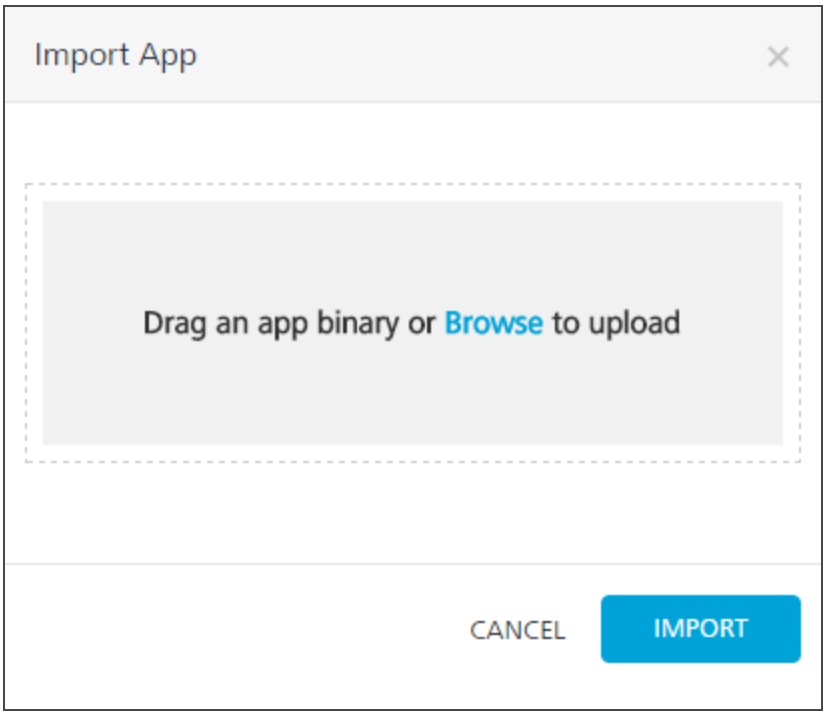
To import an app to an existing app, follow these steps:

1. From the Kony Fabric Console, click **Apps**. The **Applications** page appears.
2. In the **Applications** page, hover your cursor over the App menu button of one of the apps from

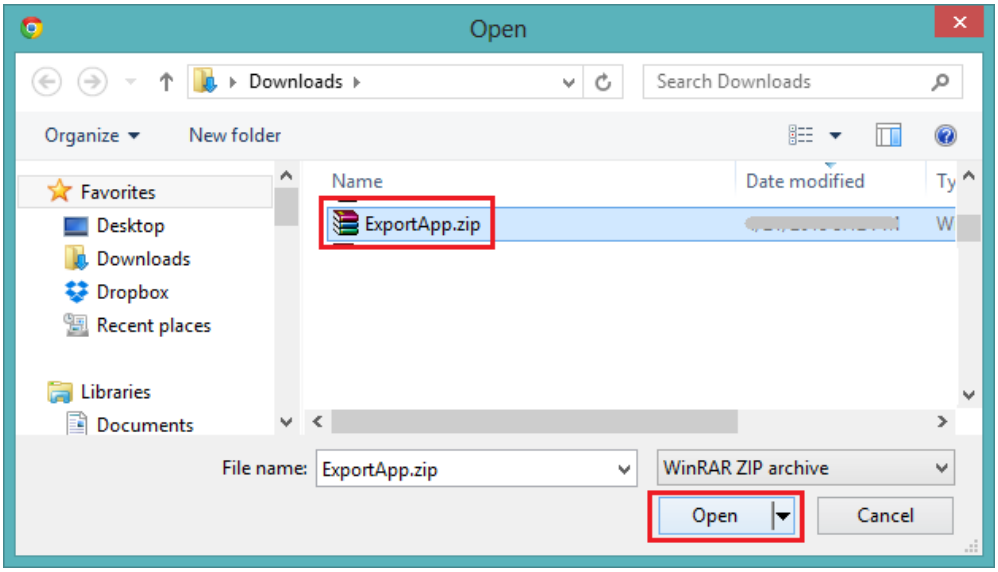
the list. The **App menu** appears.



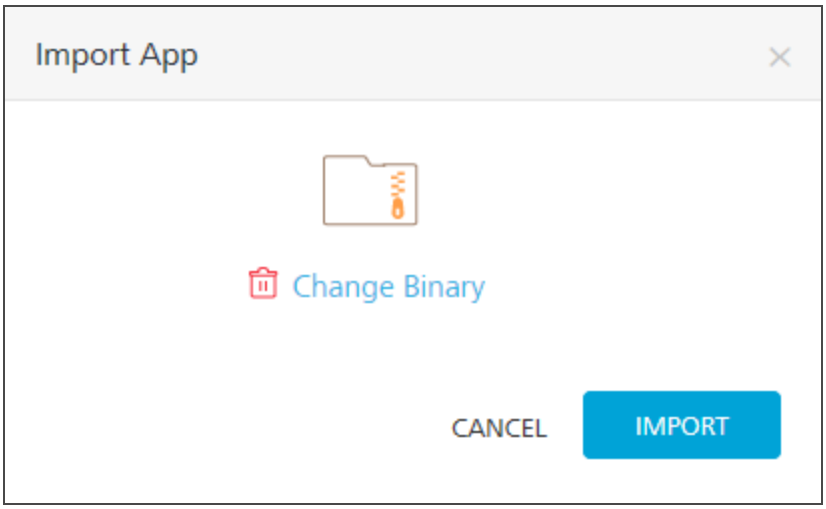
3. Click **Import**. The **Import App** dialog appears.



4. In the **Import App** dialog, click **Browse** to display Microsoft Windows **Open** dialog.



5. In the **Open** dialog, locate your exported app (for example, `ExportApp.zip` file), and select it. Click **Open**.
   In the **Import App** dialog, the selected file is added. To remove the selected file, click the **Remove** button.



6. Click **Import** to import the app. The existing app is updated with the data in the imported zip file.

> ***Note:*** You can also import an app via API. For more details, refer to Continuous Integration - Import an app via API

18.6 Folder Structure of an Exported App

The folder structure of an exported an app (a zip file) has folders, files, and certificates configured for that app. Do not make any changes to the folder structure outside Kony Fabric Console. If you make changes to the folder structure of an app, the system may throw an error while importing that app. The following section explains the hierarchical directory tree of an exported app:

```
 //Folder structure of an exported app

/Apps
   /App1
        Meta.json
```

```
        Icon file
      /_Messaging
          Meta.json
          AppleCert1.p12
          AppleCert2.p12
          AppleCert3.p12
          AppleCert4.p12
      /_Sync
          Meta.json
          /SyncScope1
              Meta.json
              Syncobject1.xml
              Syncobject2.xml
              …
   /App2
        …
/_Identity
   /Identity1
       Meta.json
       Metadata1.xml
       …
/_Integration
   /Service1
       /Endpoints
           Endpoint1.xml
       /Operations
           Operation1.xml
           Operation2.xml
           …
       WSDLFile
/_Orchestration
   /Orch1
       Operation1.xml
```

```
        Operation2.xml
        …
/_JARs
   Jar1.jar
   Jar1.meta
   …
```

The logical flow of an exported app folder structure has four levels of folders. The primary, or root, level is the Apps folder, which contains all sublevel folders including files and metadata. The following table explains hierarchical levels of an exported app folder structure:

Root	Second Level	Third Level	Fourth Level
Apps			
	/App1 • Meta.json • Icon file		
		/_Messaging • Meta.json • AppleCert1.p12	
		/_Sync • Meta.json	
			/SyncScope1 • Meta.json • Syncobject1.xml

Root	Second Level	Third Level	Fourth Level
	/_Identity		
		/Identity1 • Meta.json • Metadata1.xml	
	/_Integration		
		/Service1	
			/Endpoints • Endpoint1.xml
			/Operations • Operation1.xml
		WSDLFile	
	/_Orchestration		
		/Orch1 • Operation1.xml	
	/_JARs • Jar1.jar • Jar1.meta		

18.6.1 Apps Section

The root level (for example, App1) section has details of the apps meta file, icon file, messaging (meta file and certificates), and sync (meta file and objects). While exporting an app, an <appname>.zip file is saved with the root app name. You can rename an exported zip file, if required.

```
 //Sample data in apps (root) section of an exported app folder
structure

/App1
    Meta.json
    Icon file
    /_Messaging
       Meta.json
       AppleCert1.p12
       AppleCert2.p12
   /_Sync
       Meta.json
       /SyncScope1
           Meta.json
           Syncobject1.xml
           Syncobject2.xml
           …
```

18.6.1.1 App Meta File

The apps meta (meta.jason) file has configuration (shared and non-shared) details of an app, such as icon file, identity services, integration services, and orchestration services, shown below:

```
 //Sample data in the app meta file of an exported app folder
structure

{
  "Icon": "Iconfile",
  "description": "description",
```

```
  "Identity": [--> referencing identity providers
    "Identity1","Identity2"
  ],

  "Integration": [
    "Service1","Service2", referencing integration services
  ],
  "Orchestration": [
    "Orch1","Orch2", referencing orchestration services
  ],
}
```

18.6.1.2 App Icon File

The icon file is an image file for an app.

18.6.1.3 Messaging Section

The messaging section has referenced (non-shared) messaging services configured for an app, such as meta file and certificates configured for messaging services.

```
 //Sample data in the messaging section of an exported app folder
structure

/_Messaging
   Meta.json
   AppleCert1.p12
   AppleCert2.p12
   AppleCert3.p12
   AppleCert4.p12
```

Messaging Meta file

The messaging meta file contains information about configurations, such as ID, password, certificates, and push URL for messaging services for different platforms (Android, iPad, iPhone, BlackBerry, Windows 7, and Windows 8).

> ***Important:*** The configuration details, ID, password and push URL are not encrypted in the meta file.

```
 //Sample data in the messaging meta file of an exported app folder
structure

{
  "appleProdmode" : true/false,
  "iphonecertprod" : {
    "certName" : "AppleCert1.p12",
    "passwd" : "<password>",
  },
  "iphonecertdev" : {
    "certName" : "AppleCert2.p12",
    "passwd" : "<password>",
  },
  "ipadcertprod" : {
    "certName" : "AppleCert3.p12",
    "passwd" : "<password>",
  },
  "ipadcertdev" : {
    "certName" : "AppleCert4.p12",
    "passwd" : "<password>",
  },
  "Android": {
    "Key": "<GCM Key>",
  },
  "Blackberry": {
    "id": "",
    "passwd": "",
    "pushurl": "",
  },
  "Windows": {
```

```
    "id": "",
    "passwd": "",
    "windows7": true/false,
    "windows8": true/false,
  },
}
```

18.6.1.4 Synchronization Section

The synchronization section has the referenced (non-shared) SyncScopes configured for an app. A `syncobject.xml` file includes Sync objects of an app, such as attributes, target and source relationships, client-side filters, and life-cycle methods.

The following is the folder structure of a synchronization service:

```
//Sample data in the synchronization section of an exported app folder
structure

/_Sync
   Meta.json
   /SyncScope1 --> SyncScope1 is the name of the SyncScope
       Meta.json
       Syncobject1.xml
       Syncobject2.xml
       …
```

SyncConfig Meta file (/_Sync/Meta.json)

The SyncConfig meta file has information about database types.

> ***Note:*** MobileFabric 6.0.2 supports only MySQL database.

```
 //Sample data in the SyncConfig meta file of an exported app folder
structure
```

```
{
  "PersistentDBType": "MYSQL/Oracle/MYSQL Server",
}
```

SyncScope Meta File (/_Sync/<sync scope name>/Meta.json)

The SyncScope meta file has information about SyncScope configuration parameters specific to Sync (such as ChangeTrackingPolicy, ConflictPolicy, namespace, and strategy). The SyncScope meta file refers to an integration service and Sync interceptor jar.

The following is the meta file structure of a SyncScope service:

```
 //Sample data in the SyncScope meta file of an exported app folder
structure

[
     "SyncScope1": { --> Sync scope name
       "Strategy": "",
       "NameSpace": "",
       "ChangeTrackingPolicyType": "",
       "SoftDeleteFlag": "",
       "LastUpdateTimeStamp": "",
       "ConflictPolicyType": "",
       "DataSource": "Service1", --> Referencing integration service
       "SyncJar": "Jar1", --> referencing Sync interceptor jar
       "className": "sample", --> Class name used in case of custom
Sync
    },
    …
]
```

18.6.2 Identity Section

The identity section has the referenced (shared) identity services configured for an app.

The following is the folder structure of an identity service:

```
 //Sample data in the identity section of an exported app folder
structure

/Identity
   /Identity1 --> Identity1 is the name of the identity service
       Meta.json
       Metadata1.xml --> This metadata is required for identity
providers that have metadata, such as, SAML.
.
```

18.6.2.1 Identity meta file

The identity `meta.json` file has the configuration, type and metadata file information of the identity service. The identity metadata is required only for SAML identity services.

The following is the meta file structure of an identity service:

```
 //Sample data in the identity meta file of an exported app folder
structure

{
  "name": <name of the identity provider>,
  "displayName": <display name of the identity provider>,
  "version": <version>,
  "loginText": <login text of the identity provider>,
  "metaPreference": <meta preference >,
  "type": <type of identity provider>,
  "config": {},  --> configuration details of the identity provider
}
```

18.6.3 Integration Section

The integration section has the referenced (shared) integration services configured for an app, such as endpoints details of a particular service type, operations details of a particular service type, and additional attributes/elements for design time data.

The following is the folder structure of an integration service:

```
//Sample data in the integration section of an exported app folder
structure

/_Integration
   /Service1 --> Service1 is the name of the integration service
       /Endpoints --> only one endpoint per service is allowed
  Endpoint1.xml
       /Operations
           Operation1.xml
           Operation2.xml
           …
       WSDLFile
```

This section contains the Web Services Description Language (WSDL) file used by the soap integration service.

18.6.3.1 Endpoints file

The endpoints file has configured endpoints including the integration type, address, and credentials.

The following is the endpoint file structure of an integration service:

```
//Sample data in the endpoints file of an exported app folder
structure

<?xml version="1.0" encoding="UTF-8"?>
```

```
<endpoint name="default" type="[type]" encryptSecureInfo="false">
   <config>
      <entry>
         <key>config1</key>
         <value>value1</value>
      </entry>
      …
   </config>
</endpoint>
```

18.6.3.2 Operation file

This file contains XMLs of operations configured for an integration service.

18.6.3.3 WSDL File

This section contains the WSDL file used by the SOAP integration service.

18.6.4 Orchestration Section

This section contains only one `operation.xml` file. The orchestration section has the referenced (shared) orchestration services configured for an app.

The following is the folder structure of an orchestration service:

```
 //Sample data in the orchestration section of an exported app folder
structure

/_Orchestration
  /Orch1 --> Orch1 is the name of the orchestration service
      Operation1.xml --> looping or concurrent operation
```

18.6.4.1 Operation file

An operation file of an orchestration service has looping or composite operation configured for an orchestration service.

18.6.5 Custom Code JARs Section

This section has the referenced (shared) custom code JAR files configured for an app.

The following is the folder structure of custom code JARs:

```
 //Sample data in the custom code JARs section of an exported app
folder structure

/_JARs
         Jar1.jar       --> The JAR file
         Jar1.meta.json --> Meta for the JAR file contains information
about dependent jars.
         Jar2.jar
         Jar2.meta.json
         …
```

18.6.6 JAR Meta File

This file contains metadata of the JAR file.

The following is the structure of a JAR meta file:

```
 //Sample data in the JAR meta file of an exported app folder
structure

{
  "dependent_jars": [  --> JARs files that depend on other JAR files.
   "jar1.jar","jar2.jar"
  ]
}
```

19. Identity

Kony Fabric identity services help you secure your application by adding an authentication layer.

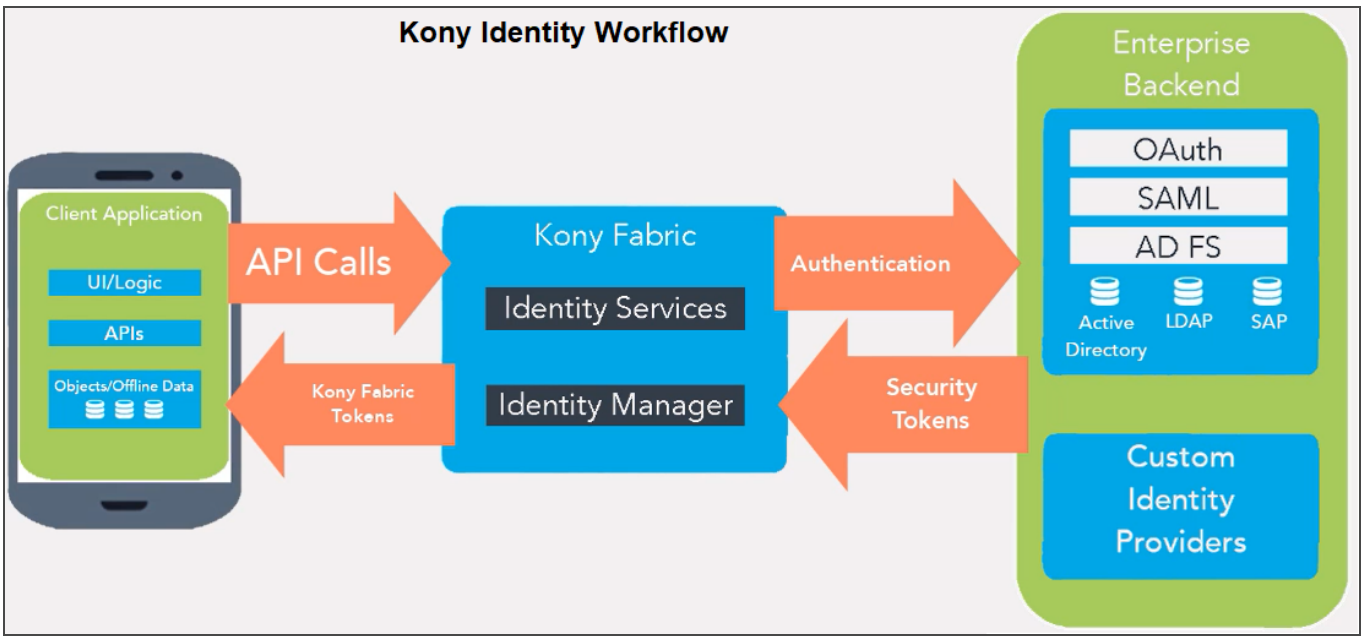
You can set up an identity service based on the type of the users who are allowed to access your application. To restrict access to your company's internal audience, use Microsoft Active Directory authentication. To allow access of your application to a larger audience, you can use enterprise identity providers (Microsoft Active Directory, Kony SAP Gateway, Open LDAP, OAuth 2.0, Salesforce, Custom Identity Service, SAML 2.0, Siteminder or Kony User Repository authentication) and social identity providers (Google, Linkedin, Instagram, Amazon, Microsoft, Yahoo, BOX, Facebook.)

19.1 Benefits of using Kony Identity

- Kony Fabric's Identity Service User Interface eliminates coding backend authentication mechanisms into your app.
- All the auth token handling is done at Kony Fabric layer to make applications more secure.

19.2 Workflow of Kony Identity Services

The following workflow describes the various stages of Identity services:



19.3 Supported Identity Providers

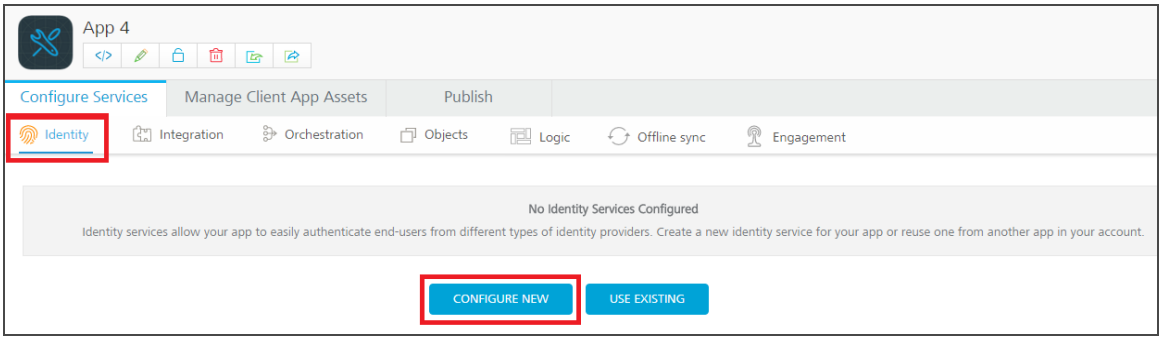
Kony Fabric Identity Services support connectivity to the following different identity providers.

Enterprise Identity Providers	Social Identity Providers (OAuth 2.0)

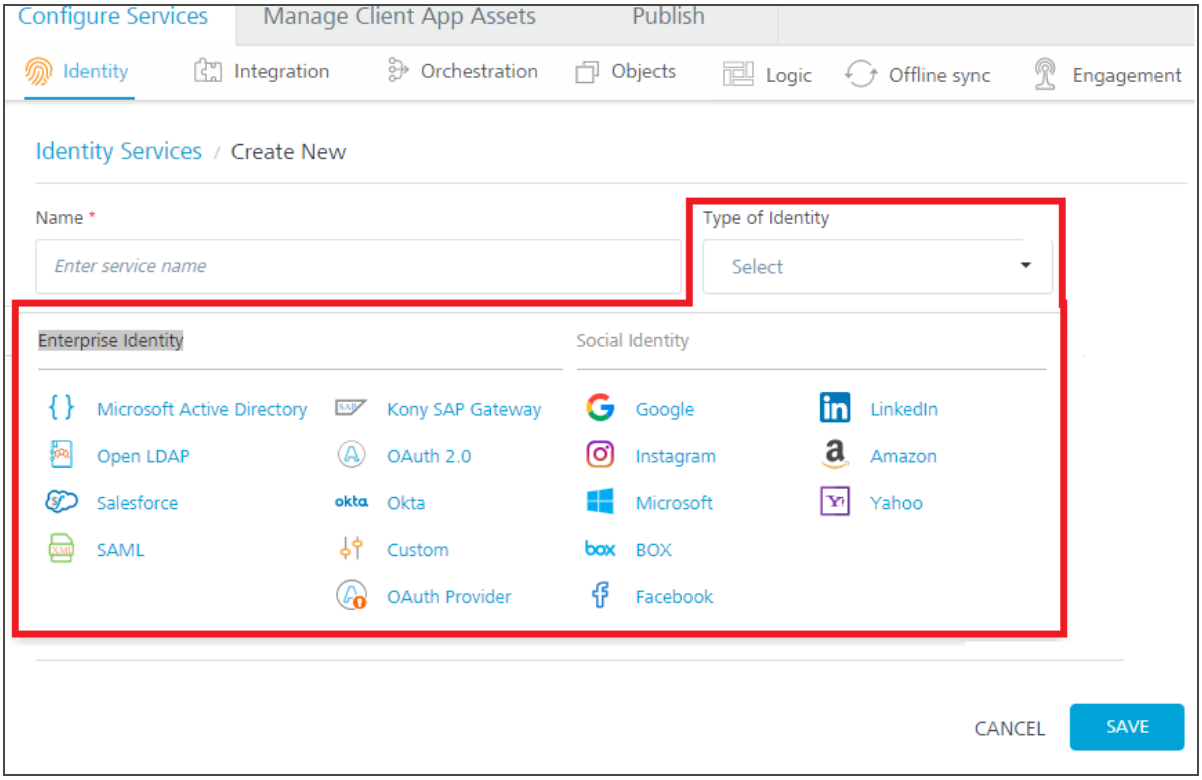
• Microsoft Active Directory
• Salesforce
• Open LDAP
• SAML 2.0
• Siteminder
• Kony SAP Gateway
• Kony User Repository (User Store) (Support is deprecated from V8 SP4 AprilFP onwards) (The Kony User Repository identity service is available in every Kony Fabric account by default. You are allocated only a single instance of the Kony User Repository for a single Kony Fabric account. Kony User Repository stores meta-data of all users in the account, such as Email, First Name, Last Name, and Password, and Groups.)
• User Repository Identity Service (You are allocated to create multiple instances in a one Kony Fabric Account. User Repository stores meta-data of users in the account, such as Email, First Name, Last Name, and Password, and Groups.)
• Kony Custom Identity Service
• Facebook
• Kony Fabric OAuth 2.0
• OAuth Provider
• Application SSO
	• Social identity providers, including:
	  • Google
	  • Linkedin
	  • Instagram
	  • Amazon
	  • Microsoft
	  • Yahoo
	  • BOX
	  • Facebook

19.4 Configure Identity Service

1. After you create an application, in the **Configure Services** tab, click the **Identity** service tab, if not selected.
2. In the **Identity** page, click **Configure New** to create an identity service.



The identity service designer appears.



3. Select the Identity provider from the Type of Identity list, and configure the service.

   For more information, refer supported Identity providers.

19.4.1 Microsoft Active Directory Identity Service

You can enable Microsoft Active Directory authentication for your application so that only those users listed in Active Directory can access your application.

> ***Note:*** NTLM authentication is not supported by Microsoft Active Directory identity service.

19.4.1.1 Configuring a New Active Directory Service

The process of configuring your Active Directory service depends on the authentication mode. Kony Fabric supports the following authentication modes:

- Security Assertion Markup Language (SAML) - An XML based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.
- Lightweight Directory Access Protocol (LDAP/LDAPS) - An open source application protocol that is commonly used for single sign-on (SSO) where one user's password is shared among various apps. The following LDAP protocols are supported:
    - LDAP without SSL - Your credentials are not encrypted before sending them for authentication.
    - LDAPS (with SSL) - Your credentials are encrypted before sending them for authentication.
- Identity Service Integration with Azure Active Directory

SAML

To create an Active Directory service using SAML authentication mode, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Microsoft Active Directory**.
3. From the **Auth Mode** list, select **SAML**.
4. Download metadata from your identity provider from the following link:

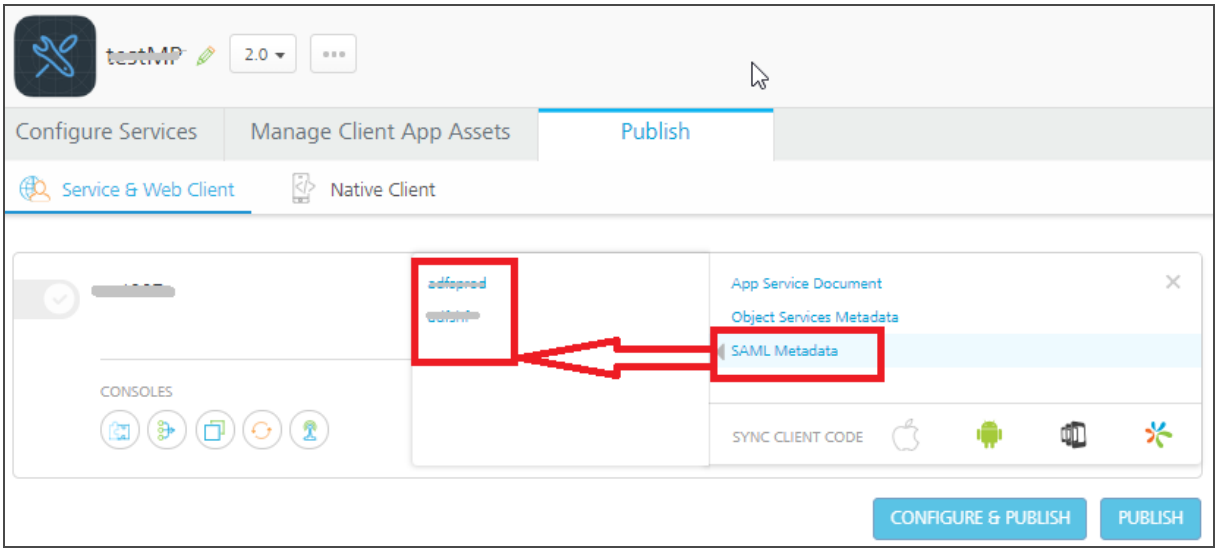
   How to download metadata from Active Directory Federation Service (ADFS)

5. From the **Metadata Mode**, select an option to upload metadata.
    - If you click **Metadata File**, the system displays **Metadata File** option. Click **Browse** to navigate to your identity provider metadata file that you downloaded, and then click **Open**. The system uploads your metadata file - for example, `idpmetadata.xml`.

- If you click **Metadata URL**, the system displays **Metadata URL** text box. Enter the URL for the metadata.

6. Under the **Choose Assertion Consumer Service Binding**, by default, this field is set to the Artifact Binding. Choose one of the following options:
   - **Artifact Binding** – to transmit SAML request and response messages in a single protocol using two different bindings.
   - **Post Binding** - to transmit SAML protocol messages within the encoded content of an HTML form control.
7. In the **Mapping of IDP SAML attributes (Optional)**, provide the information if required. This information is used for fetching profile or other information and to retrieve user information from an identity provider while logging in through SAML protocol.
   - For example, In the Mapping of LDAP attributes to outgoing claim types, you must map at least one attribute to the Name ID as SAML validates the Name ID attribute. If the Name ID is not mapped, the system throws an exception. The Name ID should not be empty - for example, User-Principal-Name to Name ID.

     Other mappings are optional - for example, Given-Name, Surname.
8. Click **Save** to create your identity provider.
9. Publish the app to an environment. The system generates the service provider's metadata for your identity provider.
10. To view the service provider's metadata, click the **Download app documentation** button in the **Published Environment** box.
11. Click **SAML Metadata** and then select the desired metadata from the list.
    The system downloads the metadata file generated by your authentication service (service provider) into your local system. For example, spmetadata.xml.



12. Upload service provider's metadata to your identity provider (ADFS). For more details, refer to How to Upload Service Provider's Metadata to ADFS.
13. In the **Publish** tab, navigate to your published app, and use the app key and app secret of your app to build the app.
14. Build your app by using Kony Fabric SDKs, and deploy the app to a device.
15. From the device, log in to your app by using the SAML identity provider that you configured.

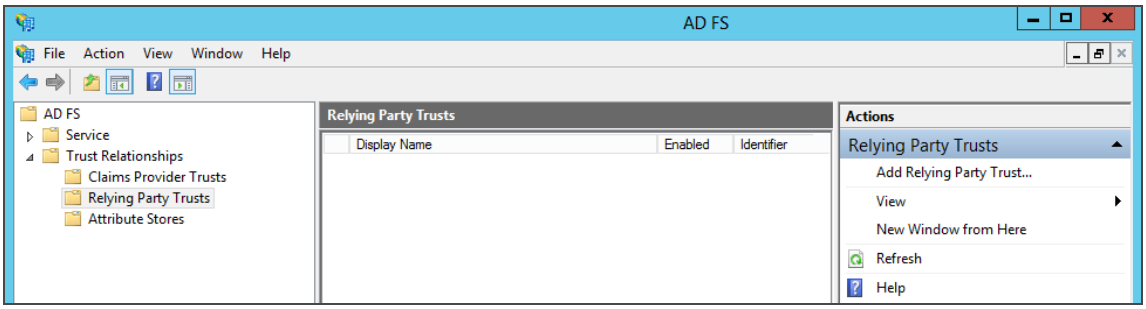
    After you are authenticated successfully, the system retrieves the profile information from the identity provider. The profile information depends on mapped attributes. If no attributes are mapped, Kony service provider shows an empty profile.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

How to Upload Service Provider Metadata to Active Directory Federation Service (ADFS)

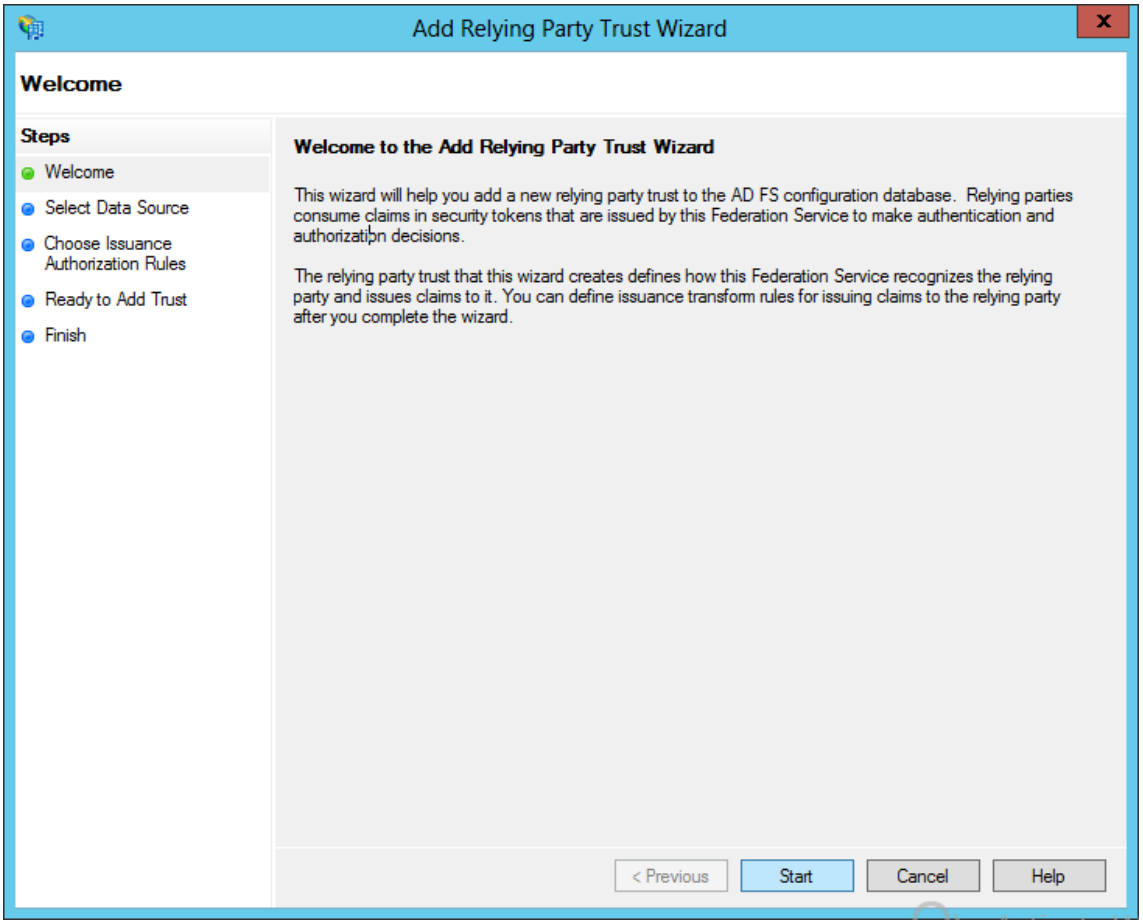
To upload your service provider's metadata to ADFS, follow these steps:

1. Log in to your IDP Active Directory Federation Services 2.0 (AD FS).



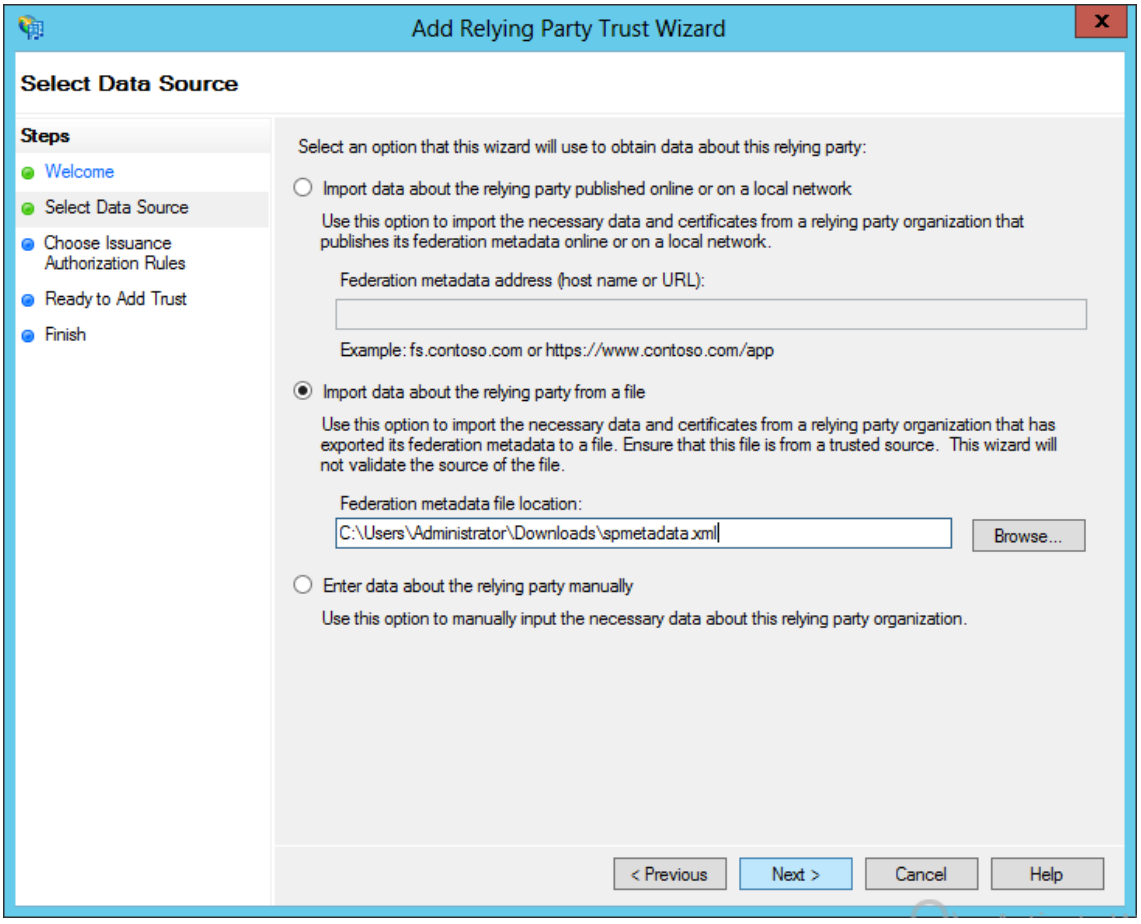
> ***Note:*** If your AD FS is configured on a system different from the current system, you need to copy the metadata file to your AD FS system.

2. In the left pane, navigate to **AD FS > Trust Relationships > Relying Party Trusts**.
3. From the **Actions** pane, click the **Add Relying Party Trust**. The **Add Relying Party Trust Wizard** appears.

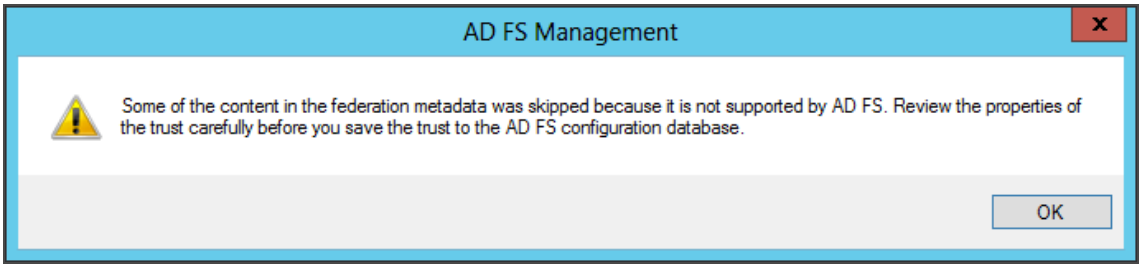


4. Click **Start**.

5. In the **Select Data Source**, select the **Import data about the relying party from a file** option. Click **Browse** to locate the metadata file that you just downloaded - for example, `spmetadata.xml`.

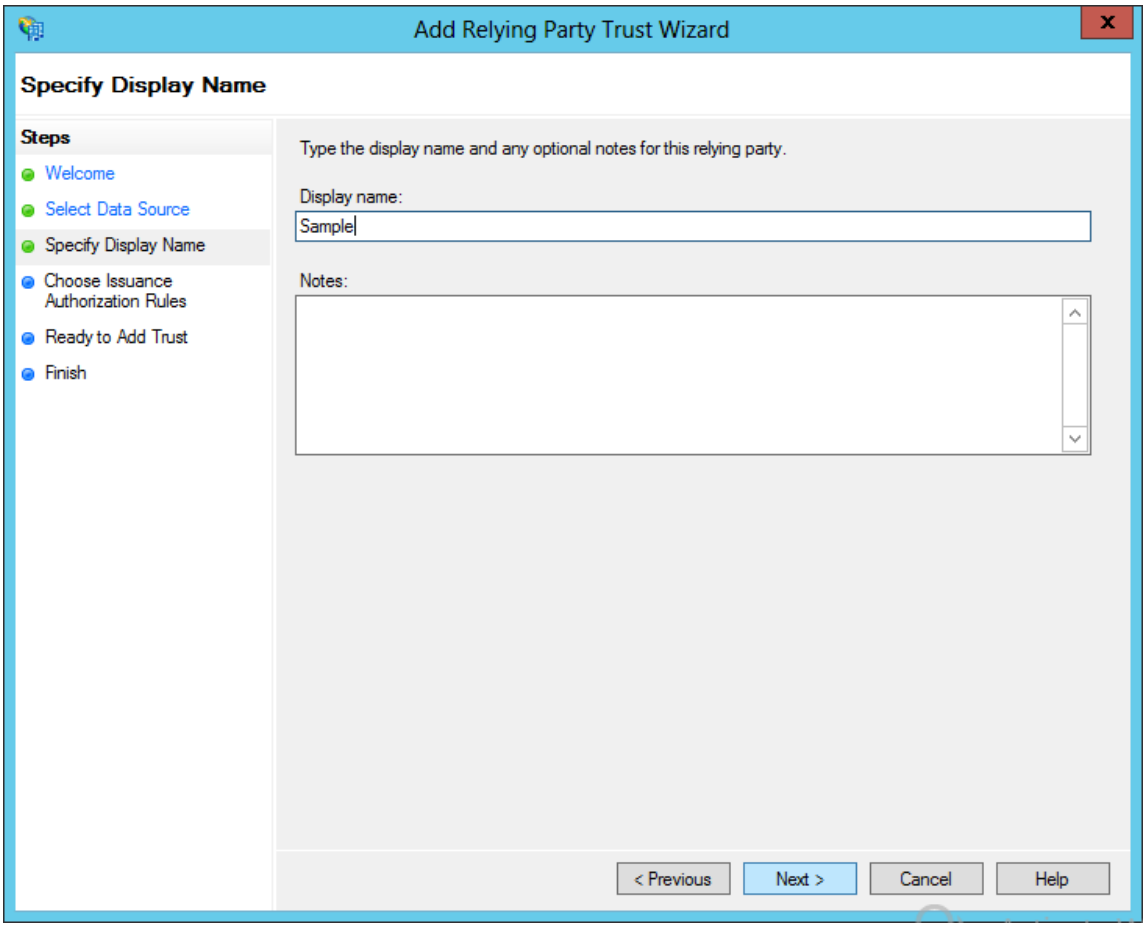


6. Click **Next**. The following message window appears.

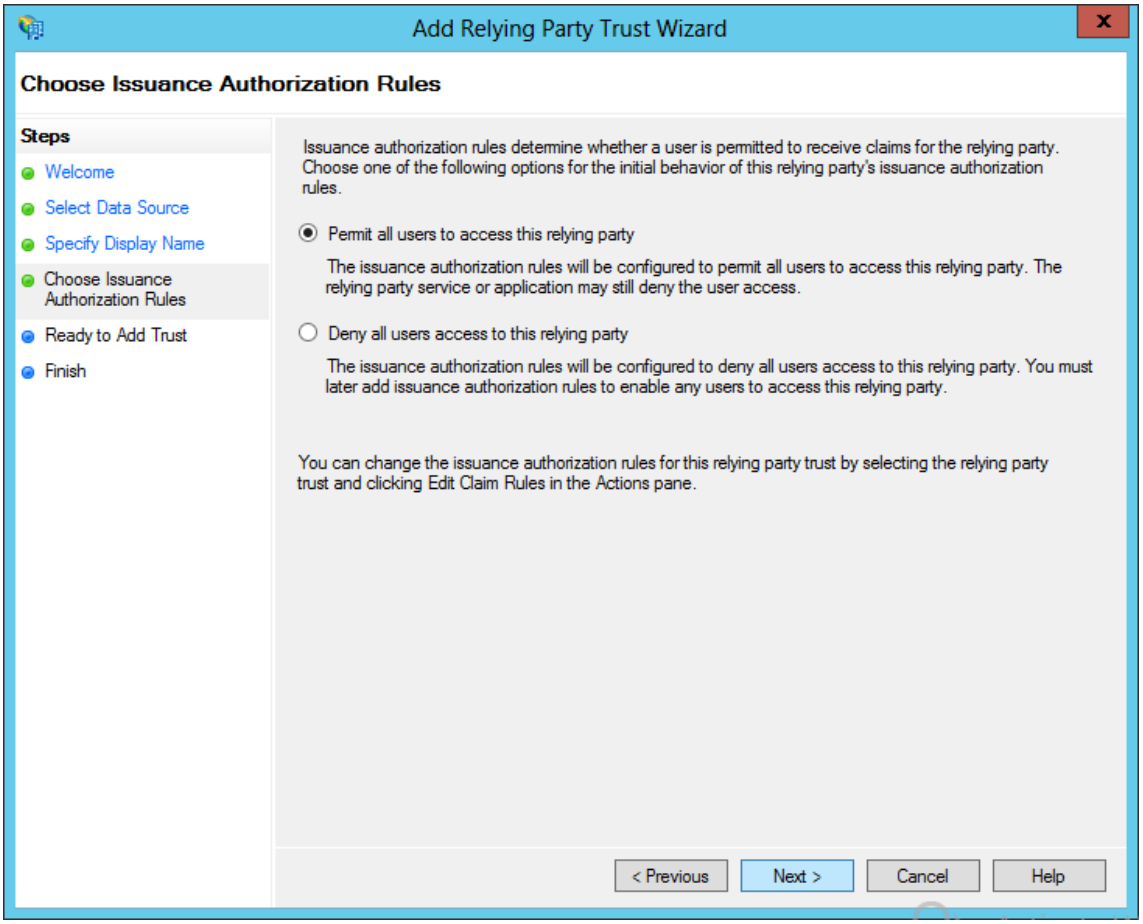


7. Click **OK** to close the message window and to proceed.

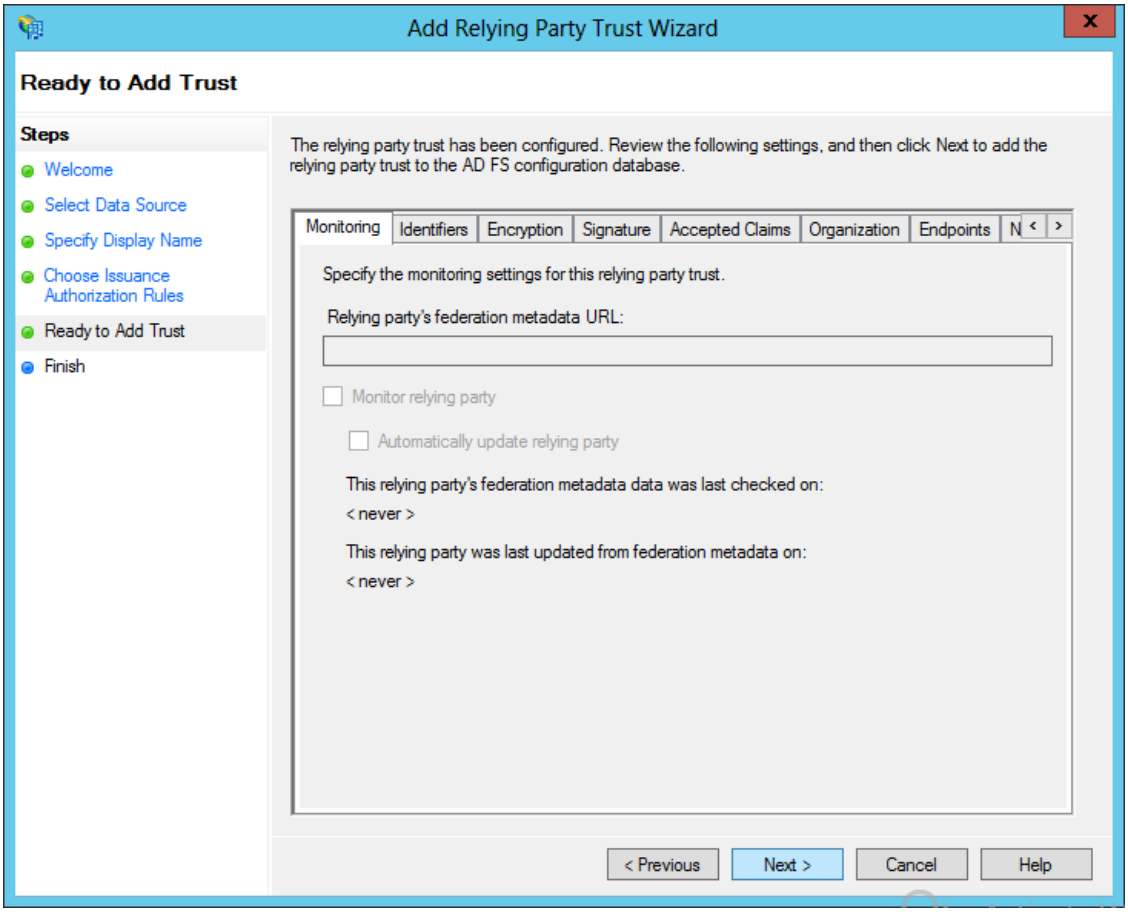
8. In the **Specify Display Name**, enter the name, and click **Next**.



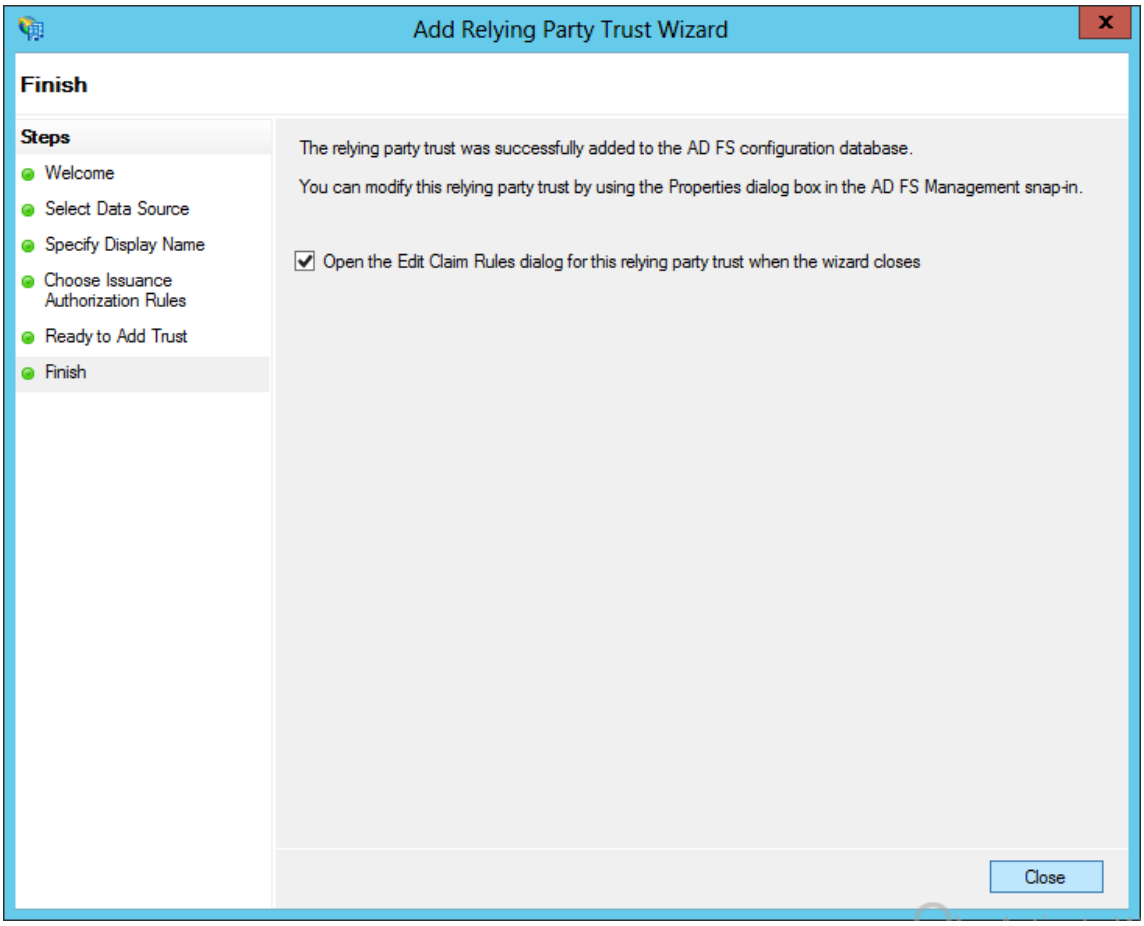
9. Select the **Permit all users to access this relying party** if that option is not already selected, and click **Next**.



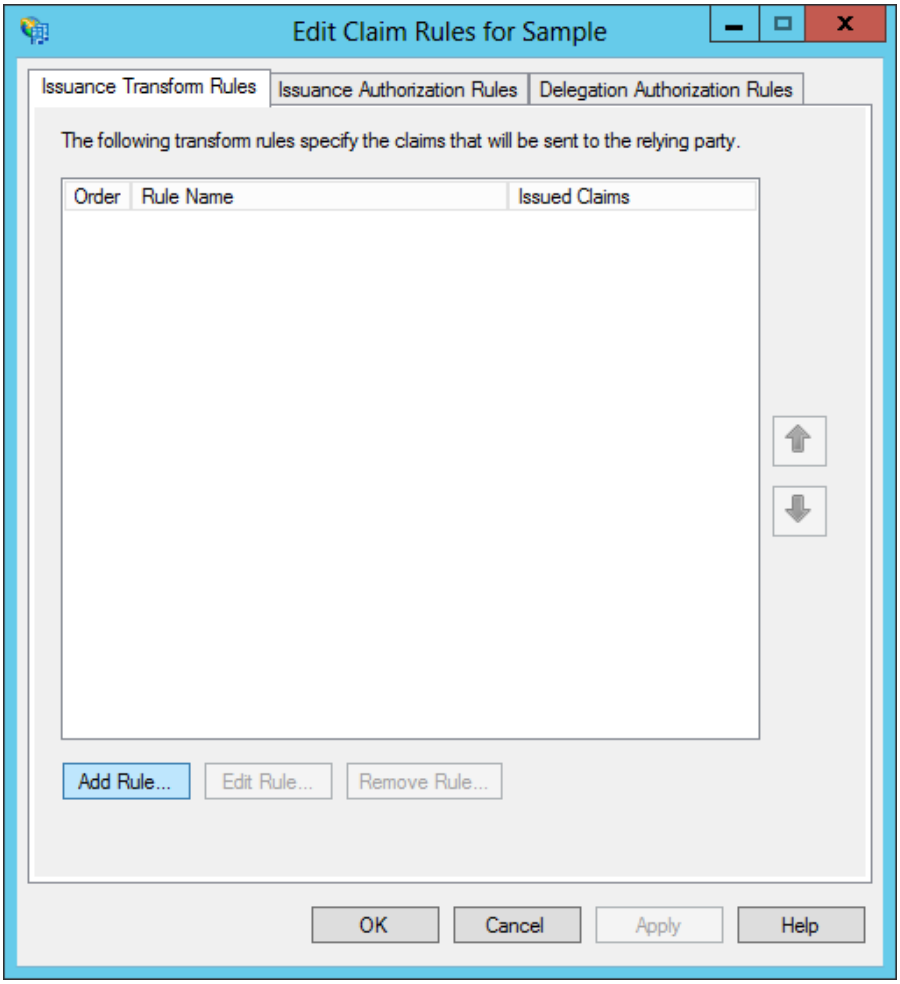
10. In the **Ready to Add Trust**, under the **Monitoring** tab, leave the fields as they are, and then click **Next**.



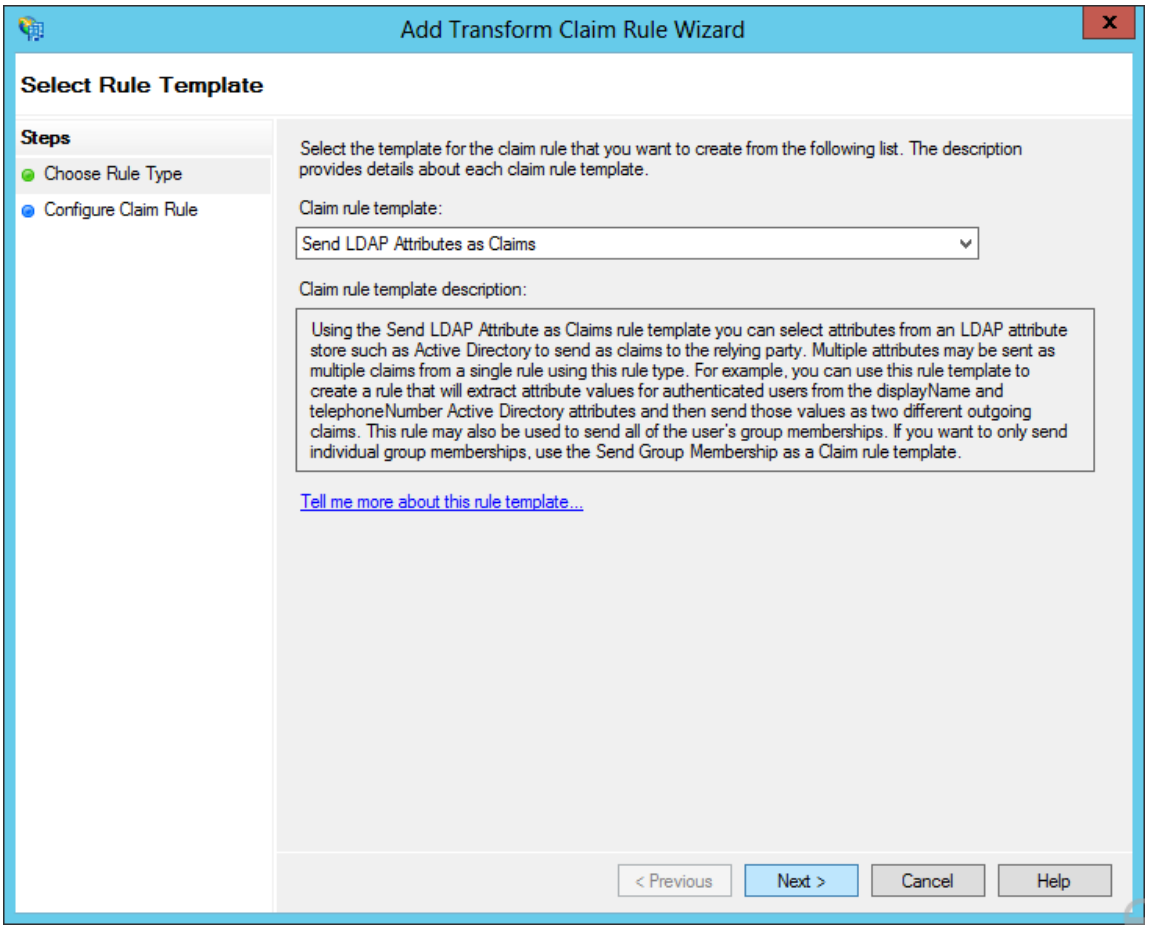
11. In the **Finish**, click **Close**.



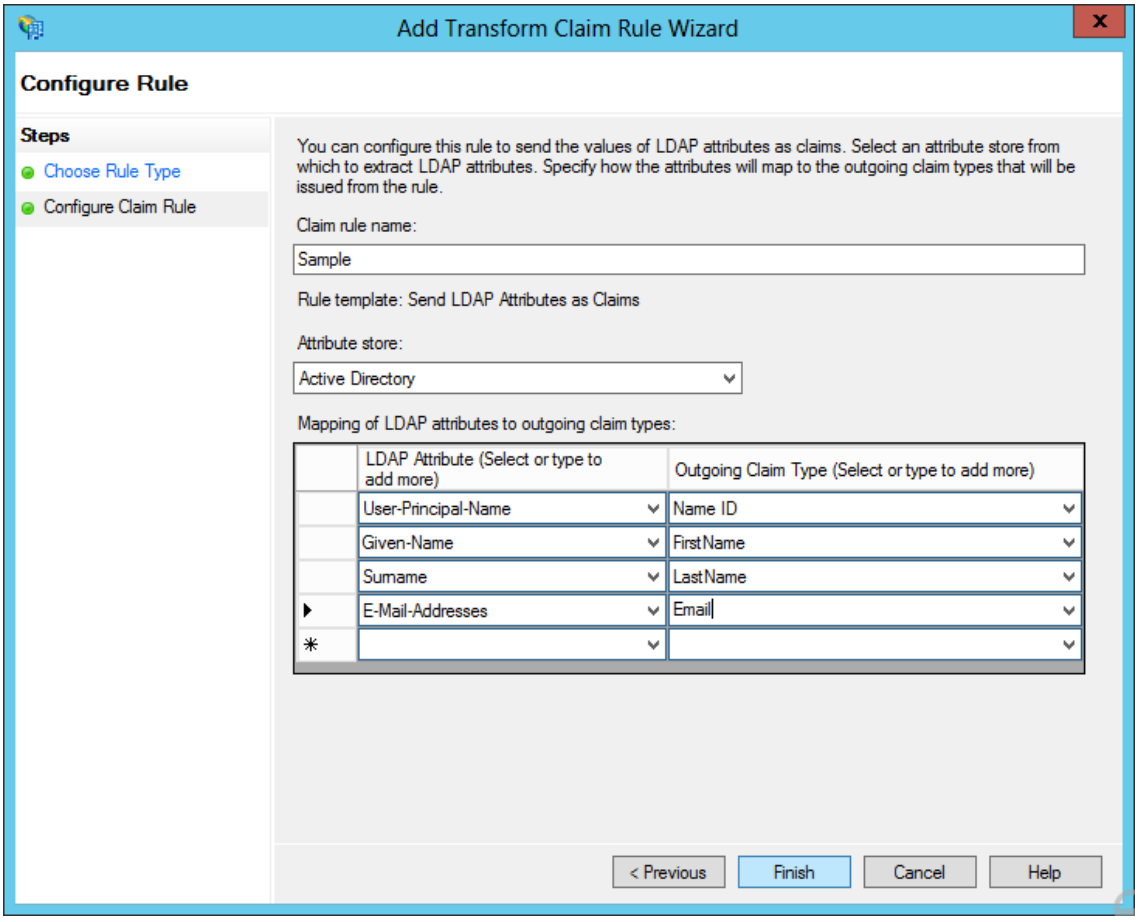
The **Edit Claim Rules** dialog appears. You need to configure the claims that you want to return by AD FS.



12. Click **Add Rule**. The **Add Transform Claim Rule Wizard** dialog appears.



13. From the **Claim rule template** list, select the **Send LDAP Attributes as Claims**, and then click **Next**.



14. In the **Choose Rule Type**, enter the following details:

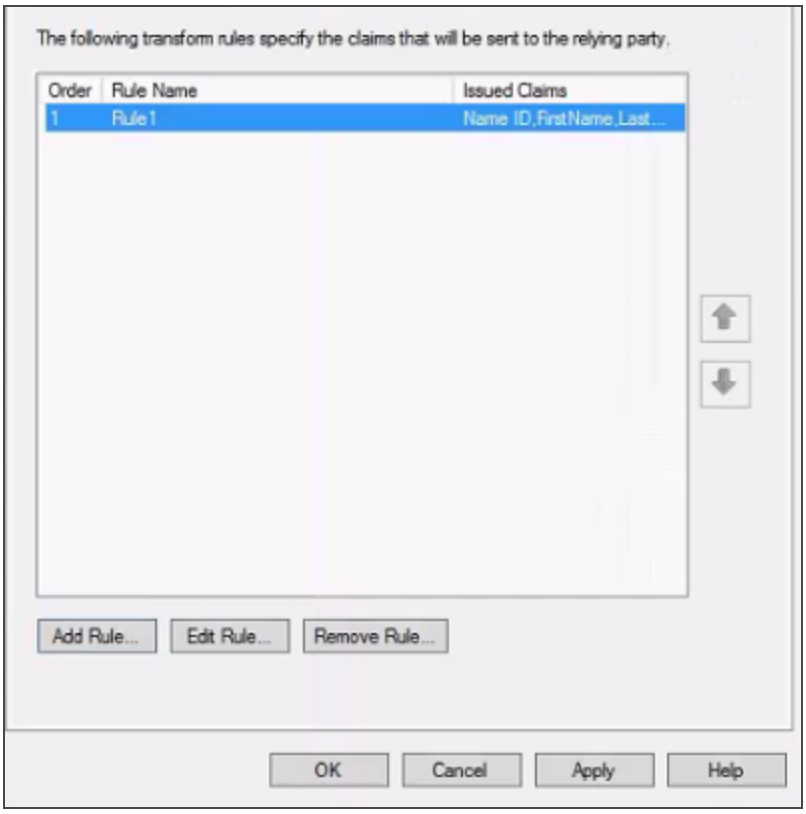
    a. In the **Claim rule name** text box, enter the name for the rule.

    b. From the **Attribute store** list, select **Active Directory**.

    c. In the **Mapping of LDAP attributes to outgoing claim types**, you must map at least one attribute to the **Name ID** as SAML validates the Name ID attribute. If the Name ID is not mapped, the system throws an exception. The Name ID should not be empty - for example, **User-Principal-Name** to **Name ID**.

    Other mappings are optional - for example, Given-Name, Surname.

d. Click **Finish**. The system creates the rule and displays the **Edit Claim Rules** dialog.



15. Click **Apply**, and then click **OK**. The identity provider is configured, and the system displays the **IDP AD FS** dialog.

LDAP/LDAPs

To create Active Directory service using LDAP/LDAPS authentication, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Microsoft Active Directory**.
3. From the **Auth Mode** list, select **LDAP/LDAPS**.

4. Under **Configure Active Directory**, provide the following details:
    a. In the **Domain Name** text field, enter a name.
    b. In the **LDAP URL** field, enter the fully qualified LDAP URL for example: `ldap://myldapserver.com:389`
    c. In the **Root Domain** field, enter the distinguished root domain name. For example, `dc=mycompany,dc=com`
    d. In the **Root Domain Scope** field, enter the scope under which it needs to search for users. For example, `dc=mycompany`, `dc=com`, and `OU=users`.
    If the root domain scope is not defined, the **Root Domain Scope** field will default to the root domain. If the root domain scope is defined, only the scope is considered, and the root domain is ignored.

    > ***Note:*** Base DN for LDAP search. If unspecified, it will default to Root Domain.

    e. In the **Login Attribute**, select the appropriate identifier from the drop-down list.
        - **userPrincipalName (UPN)**: The UPN is an Internet-style login name for the user based on the Internet standard RFC 822. The user logon name format is : `testuser@domainname.com`
        - **samAccountName**: The user logon name format is : `domainname\testuser`.
    f. In the **Federation ID**, select the appropriate identifier from the drop-down list.

5. After entering the above details, click on the **Test Login** button to verify the credentials. The test results are displayed in the **Identity Response** dialog.

6. Click the **Advanced** to provide additional configuration of your service definition:
    - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a

traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.

- **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.
    - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
    - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.
    - **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

      > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

7. After entering the above details, click **Save** to save the service. The system displays the **Identity** page. The new identity service is created for your app.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

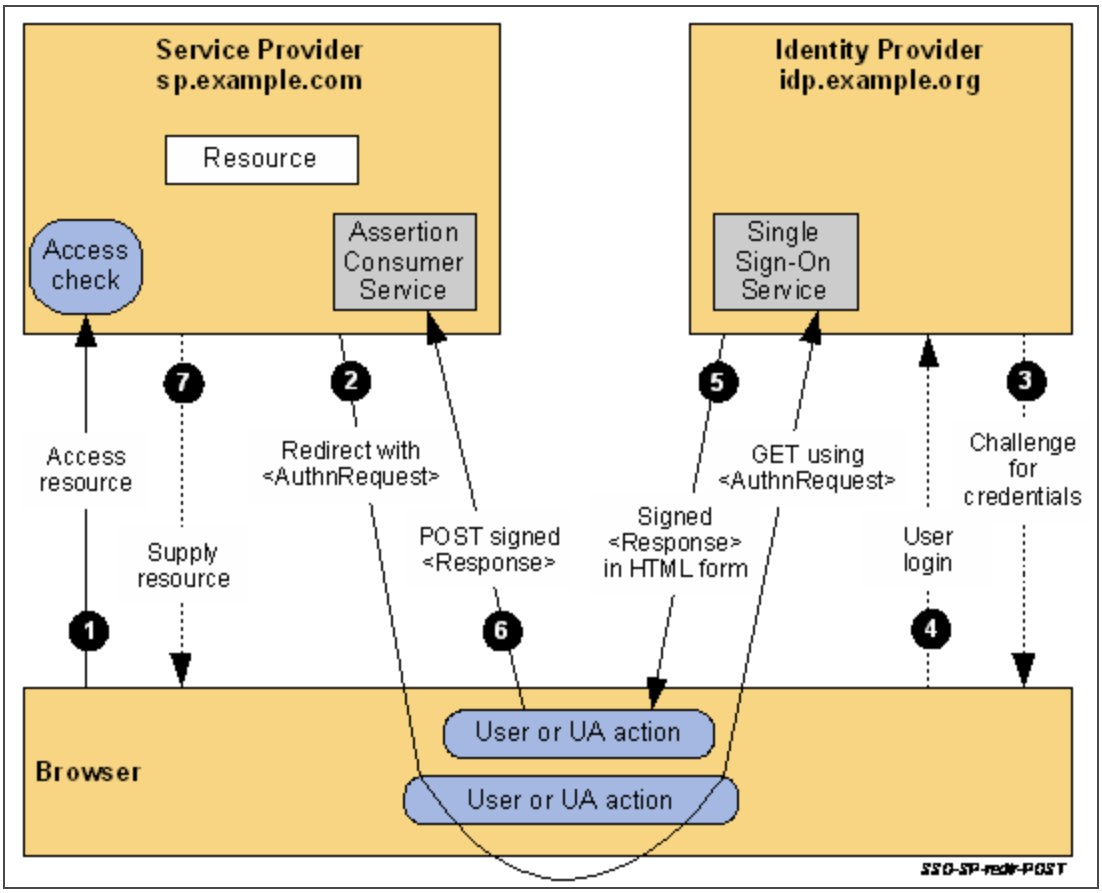
19.4.2 Identity Service Integration with Azure Active Directory

You can configure Azure Active Directory (AD) to act as an identity service to support authentication of users through the SAML protocol. To enable single sign-on (SSO), configure Azure Active directory users and the Kony Fabric app on the Azure portal.

19.4.2.1 How Service Integration with Azure Active Directory Works

A mobile app user attempts to access a resource on the service provider (the Kony Fabric identity service). The user does not have a current logon session on the identity service site. The federated identity that the user needs to access the service provider is managed by the identity provider (Azure AD). Azure AD provides a Unique Name ID that the service provider uses as the federated ID for the user.

The user is sent to the identity provider (Azure AD) to log on. The identity provider responds by sending a SAML web SSO assertion for the user's federated identity back to the service provider. In this case, the service provider uses HTTP-Redirect binding to deliver the SAML AuthnRequest message to the identity provider. The service provider uses the HTTP-POST binding to return the SAML Response message that contains the assertion to the service provider. The following figure illustrates the message flow.



19.4.2.2 Process Overview

The following describes the basic steps for configuring Azure AD to act as an identity service to support authentication of users through the SAML protocol.

1. Create the Azure AD application and configure the identity provider.

   Create the Azure AD app before you create the service provider (the Kony Fabric identity service).

   Configure the App ID URI and Sign-On URL on the Azure AD app. For the App ID URI, you use an ID that is consistent with the format of the entity ID of a service provider. You configure the Reply URL on the Azure AD app after you have created the Kony Fabric identity service.

   Collect the endpoints for federation metadata and the sign-on URL from the Azure AD app.

2. Create the Kony Fabric identity service.

   Use the metadata URL that you collected from the identity provider to create the identity service in Kony Fabric.

3. Copy and save the entity ID for the Kony service provider.

4. Edit the Azure AD app, and change the APP ID URI of the Azure AD app to that of the entity ID of the Kony service provider. Configure the Reply URL by copying the reply URL from the Kony service provider.

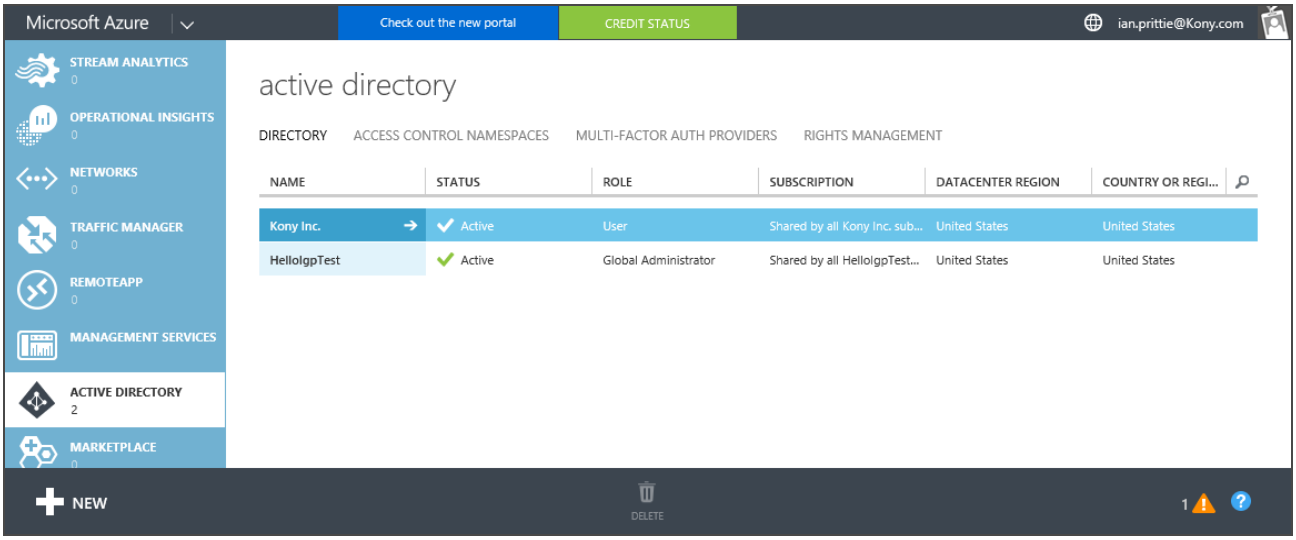
19.4.2.3 Configure the Identity Provider on the Azure Portal

To configure the identity provider on the Azure AD directory, do the following:

1. Sign into the Azure classic portal.

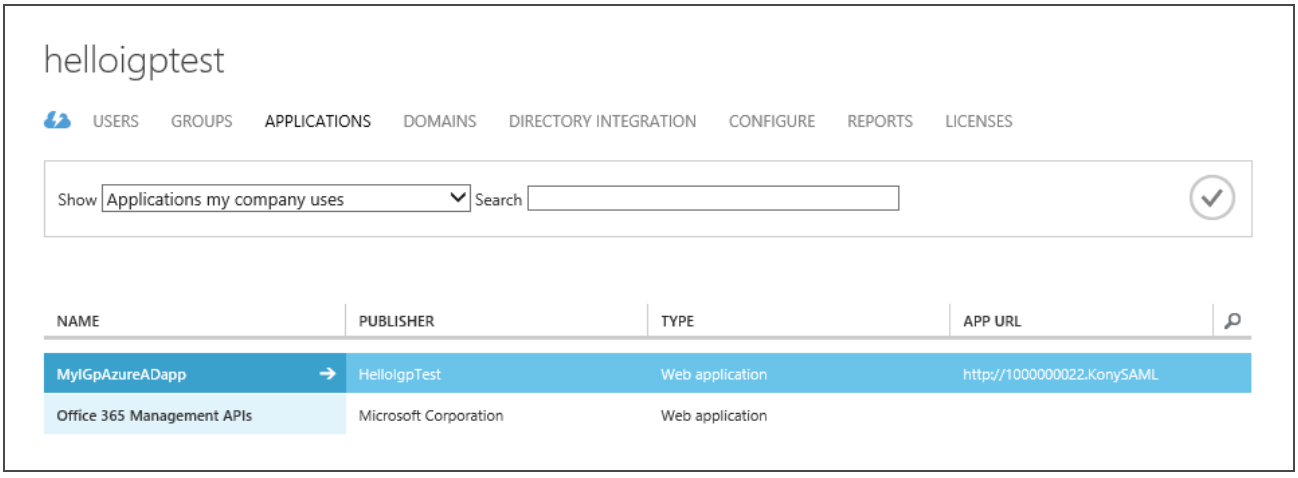
2. Click on the Active Directory icon on the left menu, and then click on the desired directory.

   If you have not created an Azure AD directory, you can add an Azure AD directory in the Azure Management Portal. Select the **Active Directory** extension on the left and click **Add**.



3. On the top menu, click **Applications**.

   If you have not added any apps to your directory, this page shows only the Add an App link.



4. Click on the **Add** button on the command bar.

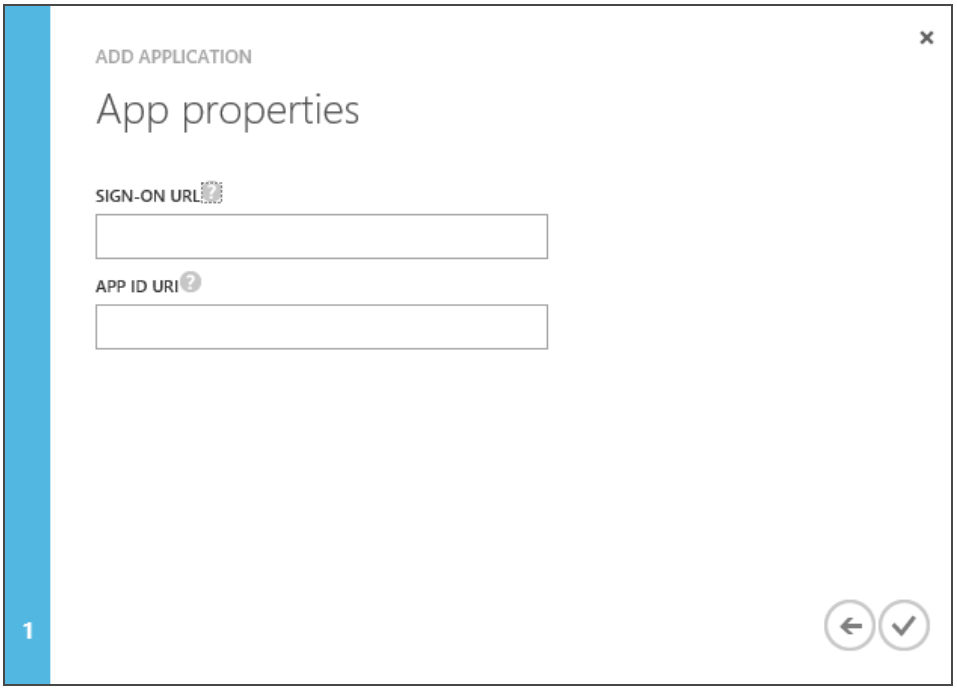
   The **What do you want to do** page appears.

5. Click **Add an application my organization is developing**.

   The **Tell us about your application screen** appears. You can indicate the type of application you are registering with Azure AD. Use the default, Web application and/or Web API.

6. Specify a name for your application, and then click the arrow icon on the bottom-right corner of the page.

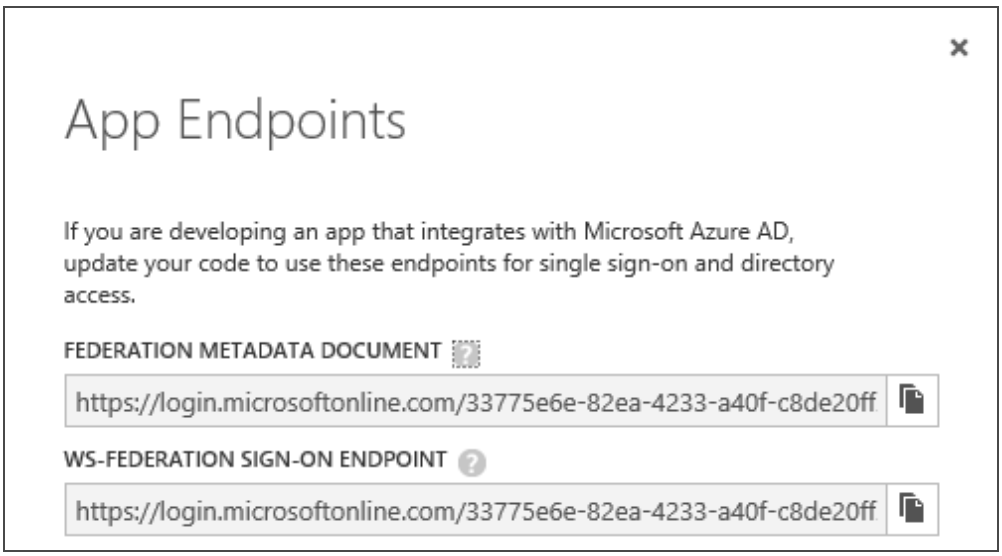
   The **App properties** screen appears.



7. Under **Sign-On URL**, provide the URL where users can sign-in and use your app. You can change this URL later after you create the Kony Fabric identity service.
8. Under **APP ID URI**, provide a unique URI that Microsoft Azure AD can use for this app. You can change this value later.

   Enter a temporary URI that that is consistent with the format of the entity ID of a service provider (the Kony Fabric identity service). For example, http://100000002.KonySAML.

   Later, after you create the Kony Fabric identity service, copy the entity ID for the identity service and enter it as the APP ID URL for the Azure AD app.
9. Click the check box in the bottom-right hand corner of the page.
10. Click **View Endpoints**.



11. Copy and save the **Federation Metadata Document URL**.

    This is the URL for the federation metadata document that your Kony Fabric app uses for authentication through Microsoft Azure AD.

12. Copy and save the **WS-Federation Sign-On Endpoint**.

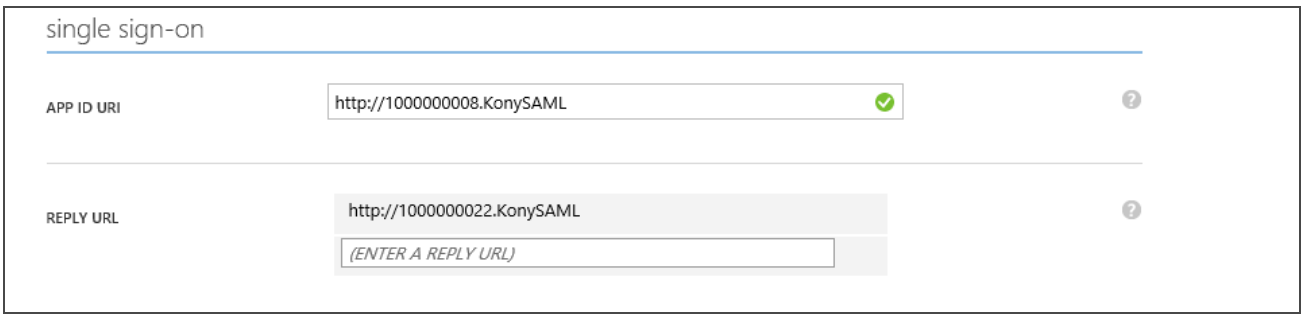
    This is the endpoint that your mobile app should send sign-on and sign-out requests to when using the WS-Federation protocol. Authentication responses will be sent to the Reply URL for the app.

13. Close the App Endpoints window.

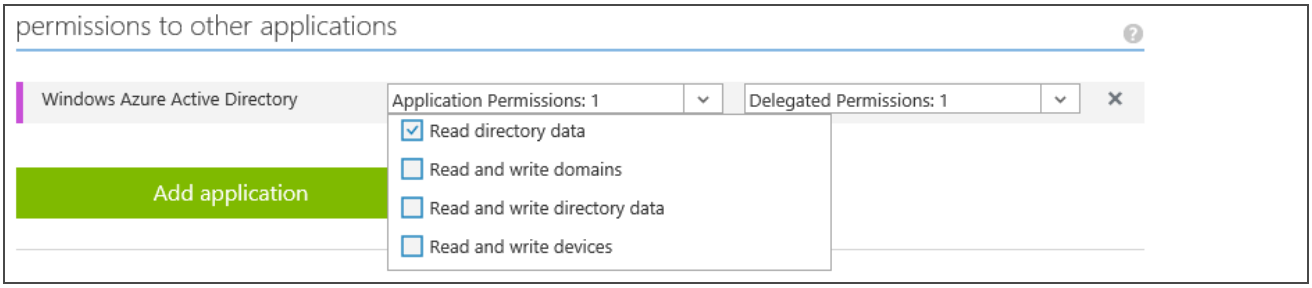
14. Click **Configure**.

15. Under **Single Sign-On**, in **Reply URL**, enter the reply URL, if available, from the Kony Fabric identity service.

    If the Kony Fabric identity service that you create provides a reply URL, copy and use the reply URL from the identity service as the reply URL for the Azure AD app.



16. Under **Permissions to Other Applications**, provide the appropriate application permissions so the Kony Fabric app can access and use Azure AD for purposes of authentication.



17. Click **Save**.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.2.4 Configure the Service Provider on the Kony Fabric Portal

After you create the identity provider on the Azure AD directory, create the service provider on the Kony Fabric portal. The following is the procedure for creating the service provider (the Kony Fabric identity service) that uses Azure AD to act as an identity service to support authentication of users through the SAML protocol.

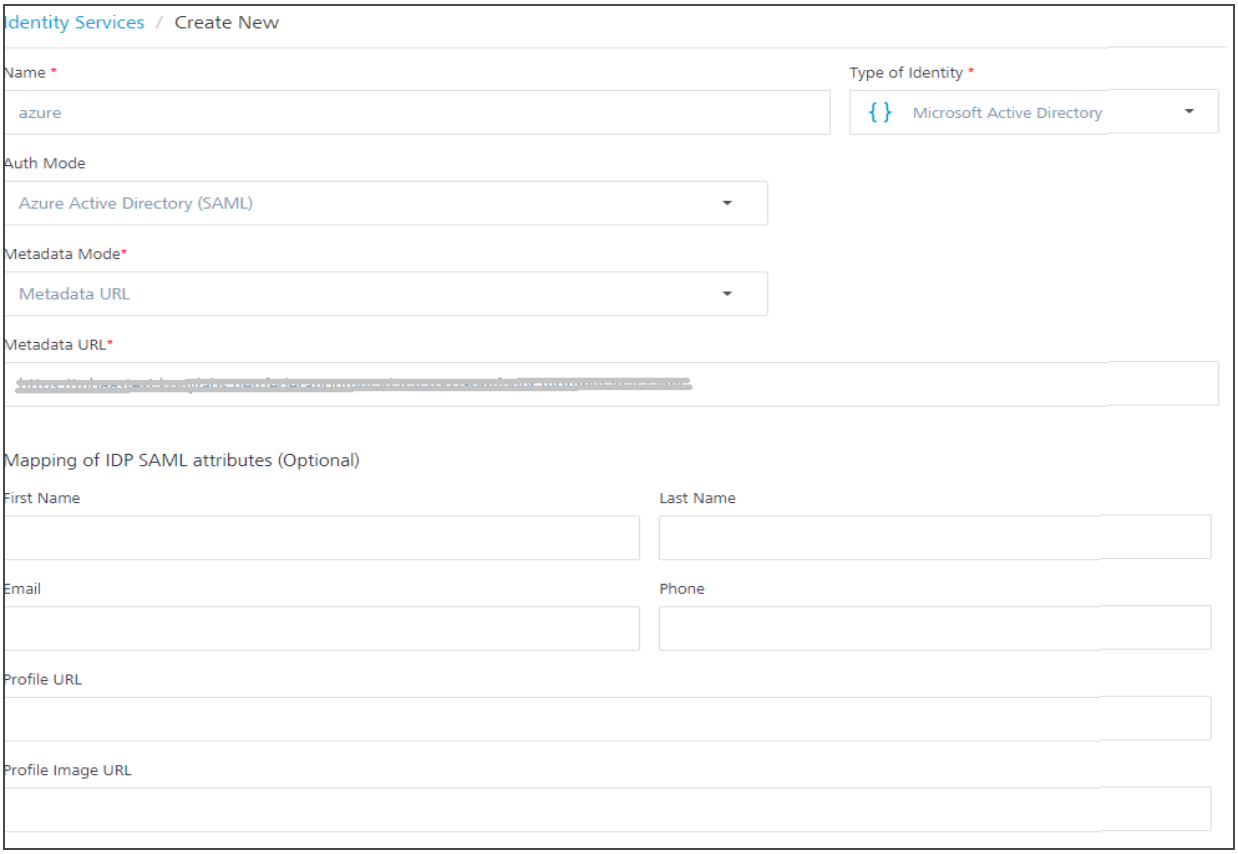
1. After you create an application, in the **Configure Services** tab, click the **Identity** service tab, if not selected.
2. In the Identity page, click **Configure New** to create an identity service.

The identity service designer appears.

3. Type a name for the service in the **Enter Service Name** text box.
4. From the **Type of Identity** list, select **Microsoft Active Directory**.
5. From the **Auth Mode** list, select **Azure Active Directory (SAML).**
6. From the **Metadata Mode**, select **Metadata URL**.

   The Metadata URL text box appears.
7. Enter the URL for the Federation Metadata Document that you copied from your Azure AD app.

   Kony Fabric provides the entity ID and reply URL of the identity service.



8. In the **Mapping of IDP SAML attributes (Optional)**, provide the information if required.

This is used for fetching profile or other information and to retrieve user information from an identity provider while logging in through SAML protocol.

9. Click **Save** to create your identity provider.
10. Publish the app to an environment. The system generates the service provider's metadata for your identity provider.
11. Copy the entity ID and reply URL for the identity service.

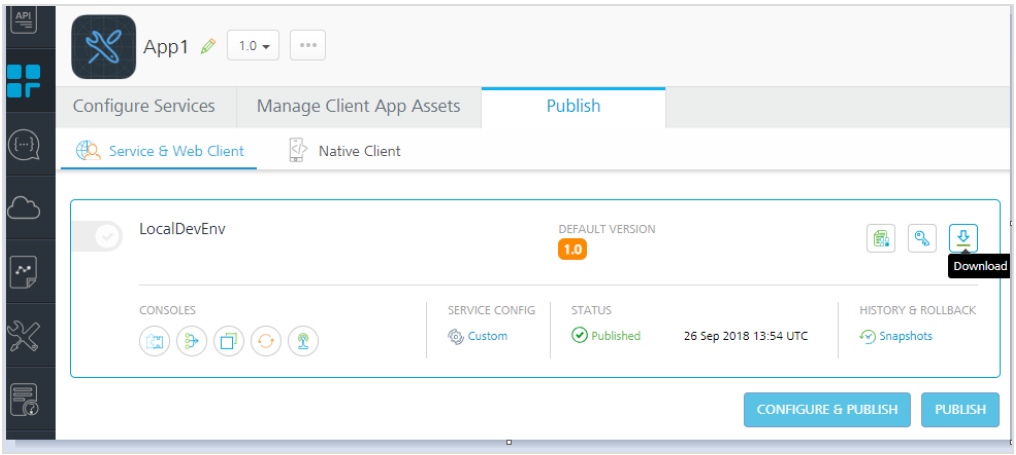
    Change the APP ID URI of your Azure AD app to that of the entity ID for the identity service. Change the REPLY URL of your Azure AD app to that of the reply URL for the identity service.

    Steps to copy the entity ID and reply URL for Identity Service.

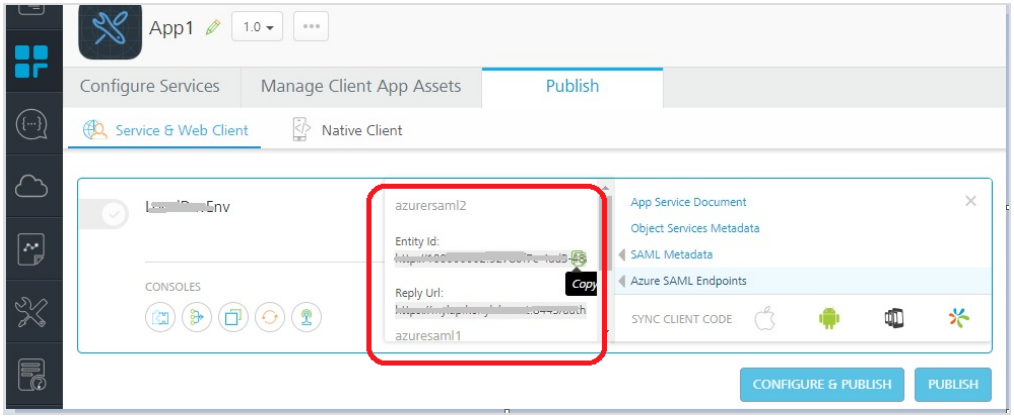
    Pre-Requisite - App must be in published state to copy the steps.

    Steps:

    1. Go to **Publish** tab.
    2. On the relevant **Environment** click **Download**. **Azure SAML Endpoints** option appears.

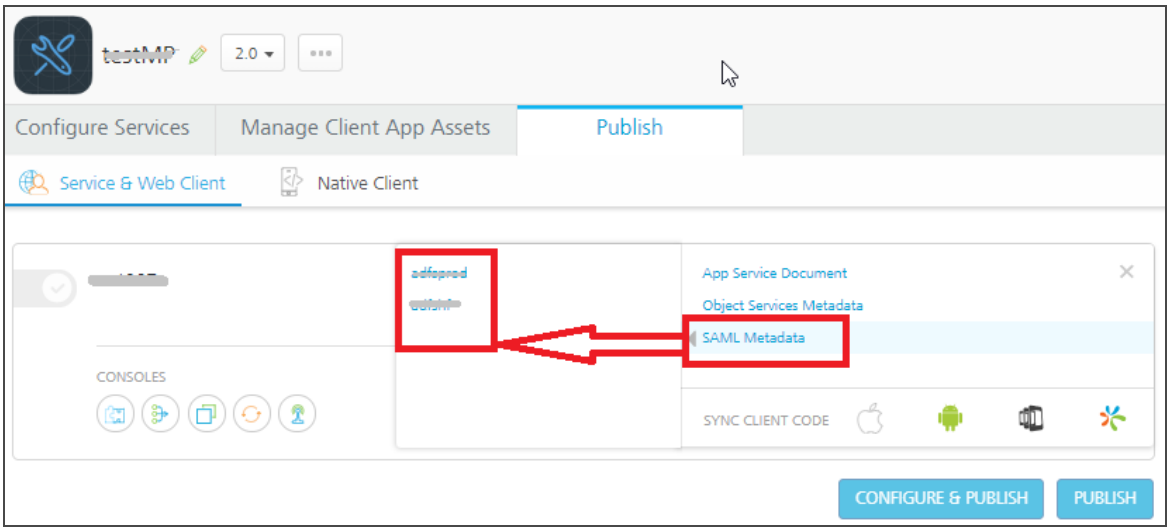


3. Click on it to view the **Entity ID** and **Reply URLs** for all Azure SAML providers linked to the app.



4. Mouse hover on the values and click **Copy** to copy the value.

12. To view the service provider's metadata, click the **Download app documentation** button in the **Published Environment** box. Click **SAML Metadata** and then select the desired metadata from the list.
    The system downloads the metadata file generated by your authentication service (service provider) into your local system. For example, spmetadata.xml.



> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.3 Salesforce Identity Service

Kony Fabric allows your users to authenticate using Salesforce credentials.

> ***Note:*** NTLM authentication is not supported by Salesforce identity service.

To configure Salesforce authentication, follow one of these methods:

- Configuring a New Salesforce service
- Using an Existing Salesforce Service

19.4.3.1 Configuring a New Salesforce Service

The process of configuring your Salesforce service depends on the authentication mode. Kony Fabric supports the following authentication modes:

- OAuth2.0: In this mode, a user is directed to a secure login page of Salesforce portal. After validating the credentials, the user is directed to Kony Fabric page with an authorization code.
- Username and Password: In this mode, users provide the Salesforce credentials. Kony Fabric, in turn, communicates these details to Salesforce. On successful authorization, Salesforce authorizes Kony Fabric to allow the users access the application.

> ***Note:*** For basic authentication on an untrusted network, Salesforce requires you to type the password followed by the security token in the **Password** box. For example, if your password is "password" and your security token is "xxxx," then the password submitted to Salesforce is "passwordxxxx." This type of authentication helps in ensuring that the integrity

> of your credentials is not compromised.
> f you forget your security token, you can reset it by following the steps mentioned in the link: ttps://help.salesforce.com/HTViewHelpDoc?id=user_security_token.htm&language=en_US

OAuth 2.0

To create a Salesforce service using OAuth 2.0 authentication mode, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Salesforce**.
3. From the **Auth Mode** list, select **OAuth (Recommended)**.

   > ***Note:*** **Salesforce URL** and **Callback URL** are pre-populated. In your Salesforce connected app, you need to type this Callback URL.

4. In the **SalesForce Client ID** box, type the client ID provided by Salesforce after you have registered your application.
5. In the **Sales Force Client Secret** box, type the client secret provided by Salesforce after you have registered your application.
6. After entering the above details, click on the **Test Login** button to verify the credentials. The test results are displayed in the **Identity Response** dialog.
7. Click the **Advanced** to provide additional configuration of your service definition:
   - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a

traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.

- **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.

8. Click **Save**.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

Username/Password

To create a Salesforce service using Username/Password auth mode, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Salesforce**.
3. From the **Auth Mode** list, select **Username/Password**.

> ***Note:*** **Salesforce URL** and **Callback URL** are pre-populated. In your Salesforce connected app, you need to type this Callback URL.

4. In the **Salesforce Client ID** box, type the client ID that is provided by Salesforce after you have registered your application.
5. In the **Salesforce ClientSecret** box, type the client secret that is provided by Salesforce after you have registered your application.

6. Click the **Advanced** to provide additional configuration of your service definition:
    - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.
    - **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.
        - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
        - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.
        - **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

            > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

7. After entering the above details, click on the **Test Login** button to verify the credentials. The **Test Login** dialog appears.
    a. Enter the User ID and Password for Salesforce backend.
    b. Click **Sign In**.

The test results are displayed in the **Identity Response** dialog.

8. Click **Save**.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.4 Open LDAP Identity Service

Open Lightweight Directory Access Protocol (LDAP/LDAPS) is an open-source application protocol that is used for single sign-on (SSO) where a user's password is shared among various apps. The following LDAP protocols are supported:

- LDAP without SSL - Credentials are not encrypted before sending them for authentication.
- LDAPS (with SSL) - Credentials are encrypted before sending them for authentication.

To bind connection with an open LDAP, you must provide the details supplied during the Active Directory LDAP configuration and the following information:

- Bind Credentials (bind username and bind password)
- Log-in attribute, federation ID, and object Class.

To create an open LDAP authentication, follow these steps:

1. Under the Identity service designer page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Open LDAP**.
3. Under **Configure Open LDAP**, provide the following details:
    a. In the **Domain Name** text field, enter a name.
    b. In the **Ldap URL** field, enter the fully qualified LDAP URL. For example: `ldap://myldapserver.com:389`

    The default port number is 389. You can change the port, if required.

    For example:

    - The following URLs are the same and valid: `ldap://myldapserver.com:389` and `ldap://myldapserver.com`.
    - The following URLs are not valid. `ldap://myldapserver.com:512` and `ldap://myldapserver.com`.

    c. In the **Root Domain** field, enter the distinguished root domain name. For example: `dc=mycompany,dc=com`

    > ***Note:*** An OpenLDAP administrator can provide you details, such as dc, uid, OU, objectClass, Login attribute.

    d. In the **Bind Username**, enter the bind username. For example: `uid=user,dc=mycompany,dc=com`

e. In the **Bind Password,** enter the bind password.

f. In the **Login Attribute**, enter the log-in attribute.

g. In the **Federation ID**, enter the unique identifier of Active Directory.

h. In the **Object Class,** specify objectClass for the search filter during authentication.

4. After entering the above details, click on the **Test Login** button to verify the credentials. The **Test Login** dialog appears.

   a. Enter the User ID and Password for back-end service.

   b. Click **Sign In**.

   The test results are displayed in the **Identity Response** dialog.

5. Click the **Advanced** to provide additional configuration of your service definition:

   - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.

   - **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.

     - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.

     - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.

- **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

  > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

6. Click **Save** to save the service. The system displays the **Identity** page. The new identity service is created for your app.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.5 SAML 2.0 Identity Service

Kony Fabric identity supports **Security Assertion Markup Language** 2.0 (SAML). SAML is an XML-based open standard data format for exchanging authentication and authorization data between parties, such as an identity provider and a service provider. SAML defines three roles:

- **Service provider** (resource server) – provides you the information.
- **Client** (web browser/user) – interacts with the resource server, like a web app being served through a web browser.
- **Identity provider** (IdP) (authorization server) – owns the user identities and credentials, and authenticates a user.

SAML allows single sign-on (SSO) with web browsers or other clients. With SSO, a user logs in once with a name and password, and can access multiple resources.

When a user logs into an application (either a mobile app or web app), the service provider issues an authentication request to a SAML identity provider through the user agent (usually a web browser.) After the user logs in (as part of SAML identity provider log-in), the IdP generates a SAML token that includes assertions about the user (such as user name, email, or other authorization information). The service provider verifies the SAML token (identity provider of the user information), and provides access to its services or resources. When the process completes, the user can interact with the application/web resources.

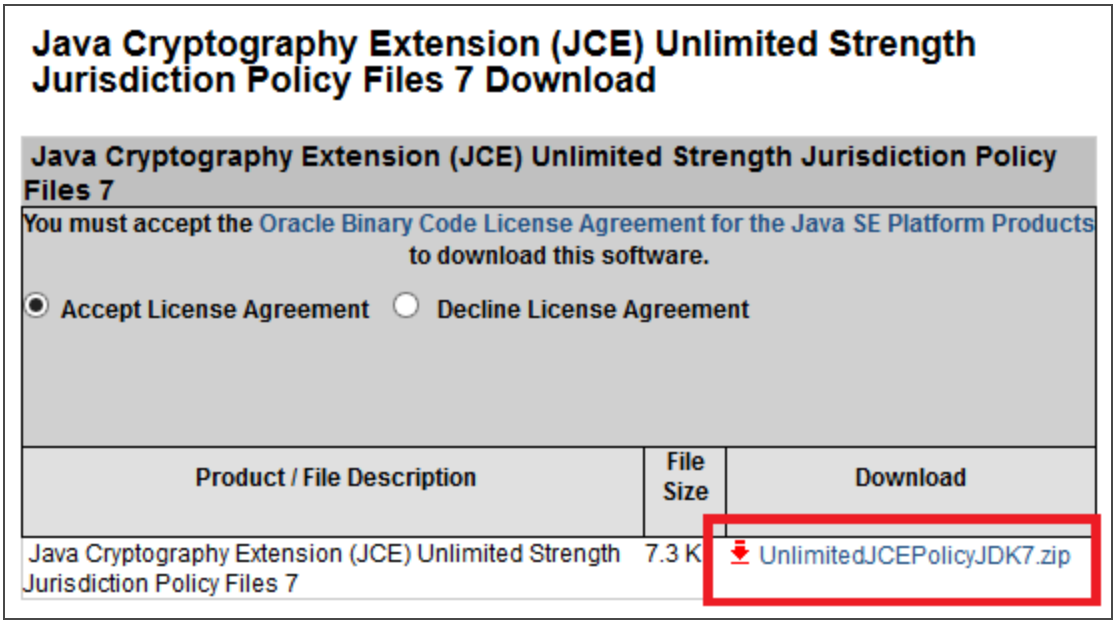
> ***Note:*** NTLM authentication is not supported by SAML identity service.

> ***Note:***

19.4.5.1 Prerequisites

To enable SAML ADFS login, follow these steps:

1. From http://www.oracle.com/technetwork/java/UnlimitedJCEPolicy, download the Java Cryptography Extension (JCE) files for the Java version you are using.



The `UnlimitedJCEPolicyJDK7.zip` (unlimited strength) contains the following files:

- local_policy.jar
- US_export_policy.jar
- README.txt

2. In your Kony Fabric installation folder, go to the `USERINSTALLDIR/jre/lib/security` folder. Replace the policy files with the unlimited strength policy files (local_policy.jar and US_export_policy.jar) that you downloaded from the Oracle website.
3. Restart Kony Fabric server.

The following sections describe how to configure and use a SAML service:

- Configuring a new SAML service
- Using an existing SAML service

19.4.5.2 Configuring a New SAML Service

To create a SAML service, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **SAML**.
3. Download metadata from your identity provider. For more information, see How to download metadata from Salesforce
4. From the **Metadata Mode**, select an option to upload metadata.
    - If you click **Metadata File**, the system displays **Metadata File** option. Click **Browse** to navigate to your identity provider metadata file that you downloaded, and then click

Open. The system uploads your metadata file. For example, `idpmetadata.xml`.

- If you click **Metadata URL**, the system displays **Metadata URL** text box. Enter the URL for the metadata.

5. Under **Choose Assertion Consumer Service Binding**, select one of the following options:
   - **Artifact Binding** - to transmit SAML request and response messages in a single protocol using two different bindings.
   - **Post Binding** - to transmit SAML protocol messages within the encoded content of an HTML form control.

   > ***Note:*** By default, this field is set to Artifact Binding.

6. In the **Mapping of IDP SAML attributes (Optional)**, provide the information if required. This information is used for fetching profile or other information and to retrieve user information from an identity provider while logging in through SAML protocol.
   - For example, In the Mapping of LDAP attributes to outgoing claim types, you must map at least one attribute to the Name ID as SAML validates the Name ID attribute. If the Name ID is not mapped, the system throws an exception. The Name ID should not be empty - for example, User-Principal-Name to Name ID.

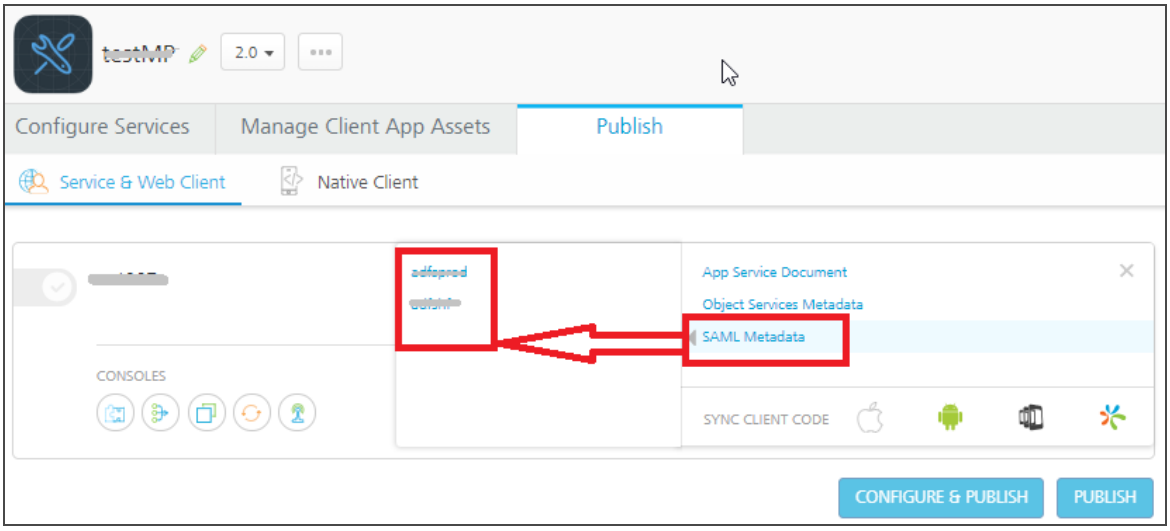
     Other mappings are optional - for example, Given-Name, Surname.

7. Click the **Advanced** to provide additional configuration of your service definition:
   - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.

- **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.
    - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
    - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.
    - **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

      > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

8. Click **Save** to create your identity provider.
9. Publish the app to an environment. The system generates the service provider's metadata for your identity provider.
10. To view the service provider's metadata, click the **Download app documentation** button in the **Published Environment** box.
11. Click **SAML Metadata** and then select the desired metadata from the list.
    The system downloads the metadata file generated by your authentication service (service provider) into your local system. For example, spmetadata.xml.



12. Upload service provider's metadata to your identity provider (Salesforce). For more details, see the topic How to Upload Service Provider's Metadata to Salesforce.

13. In the **Publish** tab, navigate to your published app, and use the app key and app secret of your app to build the app.

14. Build your app by using Kony Fabric SDKs, and deploy the app to a device.

15. From the device, log in to your app by using the SAML identity provider that you configured.

    Once you are authenticated successfully, the system retrieves the profile information from the identity provider. The profile information depends on mapped attributes. If no attributes are mapped, Kony service provider shows an empty profile.

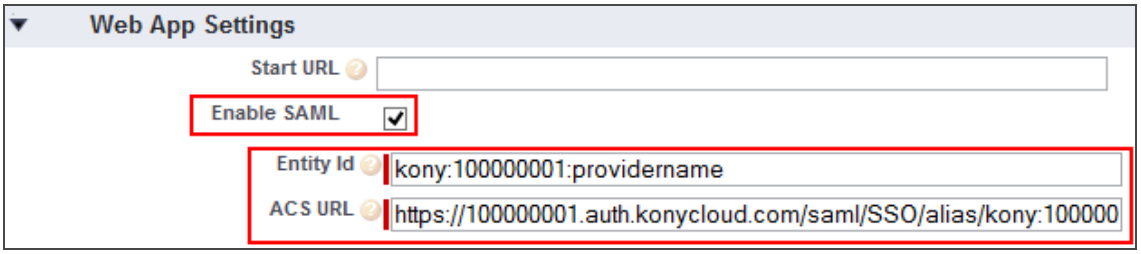
> ***Note:*** **Logout from a browser session not supported for Kony SAML Identity Provider:** When a user logs out from the Kony SAML identity connector, only the identity session is cleared and does not log out from the browser session of the Identity Provider (IdP). Currently, support for the Kony SAML identity connector logout of browser session of IDP is not available. So, this results in the user's IDP session cookies that are created as part of login to SAML IDP are not cleared. After the user logs in again, the Login page is not displayed to the user until the IDP session cookies get expired due to timeout.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

How to Upload a Service Provider's Metadata to Salesforce

To upload your service provider's metadata to Salesforce, follow these steps:

1. Log in to your Salesforce account and create a connected application. For more details about creating a connected app, refer to https://help.salesforce.com/apex/HTViewHelpDoc?id=connected_app_create.htm&language=en_US.
2. Once you create a connected application, in the **Web App Settings** section, select the **Enable SAML** check box to enable your connected app for SAML service provider.
3. From your service provider metadata file you downloaded at in the previous section > Configuring a New SAML Service section, do the following:
   - Copy the value of the `entityID`. For example, **kony:100000001:providername**
   - Copy the value of the `AssertionConsumerService URL`. For example, **https://100000001.auth.konycloud.com/saml/SSO/alias/kony:100000001:providername?provider=provideernam**
4. In the **Web App Settings** section, do the following:



a. In the **Entity Id** text box, paste the value that you copied for `entityID` in Step 3 in this section.

b. In the **ACS URL** text box, paste the value that you copied for `AssertionConsumerService URL` in Step 3 in this section.

5. Click **Save** to save your settings for SAML.

> ***Important:*** While logging on by using the SAML provider, ensure that you have required permission set to access the connected app.

19.4.5.3 How to Use an Existing SAML Service

To use an existing service, follow these steps:

1. On the **Identity** tab, click **Use Existing** to open the **Existing Services** page.
2. Select the check box for the existing service of type SAML from the list.
3. Click **ADD**.

   The service is added and is available in the **Identity** page of your app.

> ***Note:*** The Existing Services page contains a list of services created within the same parent account.

19.4.6 SiteMinder Identity Service

After a developer configures a SiteMinder instance in Kony Fabric Console, the SiteMinder instance handles authentication and authorization of users. When a SiteMinder user logs in Kony Fabric app, the system directs the user to the SiteMinder log-in page. After a SiteMinder user logs into the

SiteMinder instance, the user will be redirected to Kony Fabric Console. Kony Fabric auth service will verify whether SiteMinder headers and cookies exist before granting access to Kony Fabric design-time components. Kony Fabric components include the Kony Fabric Console, Kony Fabric Engagement, Kony Fabric Sync, and Kony Fabric Integration Server.

> ***Note:*** SiteMinder Identity service support is available only for Kony Fabric On-premises.

To configure a SiteMinder service, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **SiteMinder**.
3. In the **Siteminder Session Cookie Name** field, enter a cookie name.
4. Under **User Attribute Selectors from Request Header**, provide the following details:
    a. In the **Federation ID** field, enter a valid ID.
    b. In the **Email ID** field, enter a valid Email ID.
    c. In the **First Name** field, enter your first name.
    d. In the **Last Name** field, enter your last name.
5. Under **User Group Endpoint Details**, in the **URL** field, enter a valid URL.
   For example, **http://localhost:xxxx/customidp/login**
6. Click **Save** to finish configuration of the identity provider.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.7 Kony SAP Gateway Identity Service

You can enable Kony SAP Gateway authentication for your application so that only those users registered with external SAP services can access these services in the application.

The following sections describe how to configure and use a Kony SAP Gateway service:

- How to Configure a New Kony SAP Gateway

19.4.7.1 How to Configure a New Kony SAP Gateway

To configure a Kony SAP Gateway, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Kony SAP Gateway**.
3. In the **Gateway address**, enter **connect.kony.com**.
4. In the **Port** text box, enter a valid port between 1 to 65535.

   Click **Test Connection** to validate the gateway and port details.
5. You can also configure custom login context that is appended at the end of the gateway. To configure custom login contexts. select the **Use Custom Context** check box, and do the following

   a. **Login Context**: Enter the custom login context.

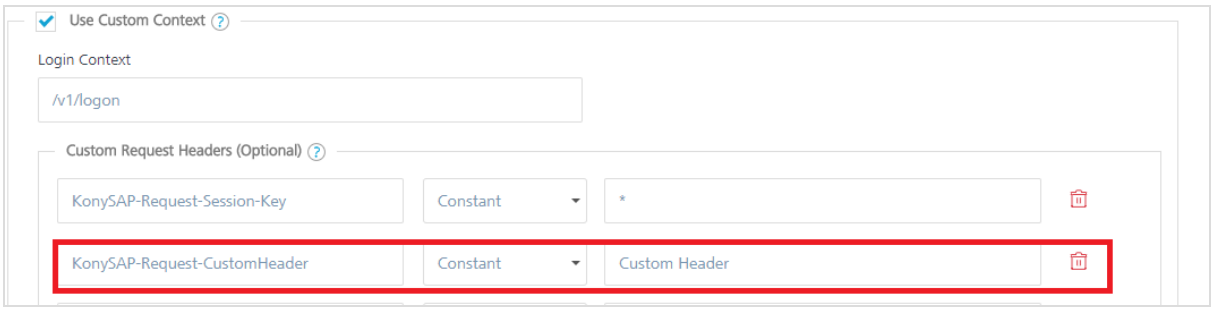
   b. **Custom Request Headers (Optional)**

   - KonySAP-Request-Session-Key
   - Kony Fabric supports multiple ways to configure dynamic headers in **SAP Identity Provider** configuration. There are three modes to configure dynamic headers for

SAP provider in Kony Fabric.

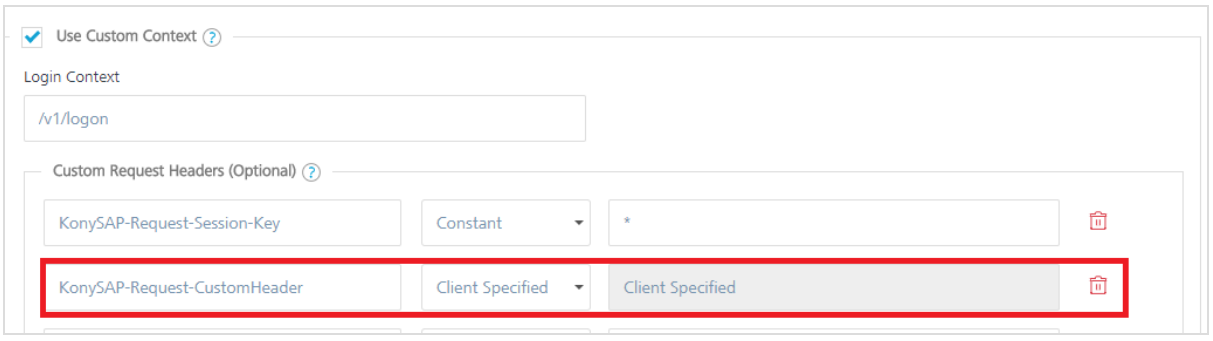
- Constant
- Client Specified
- Identity Session

Constant: This mode allows you to configure the custom header key and its corresponding value to be sent as a part of the backend login request.



The value set as a part of the configuration is sent directly to SAP as a request header in the login request.

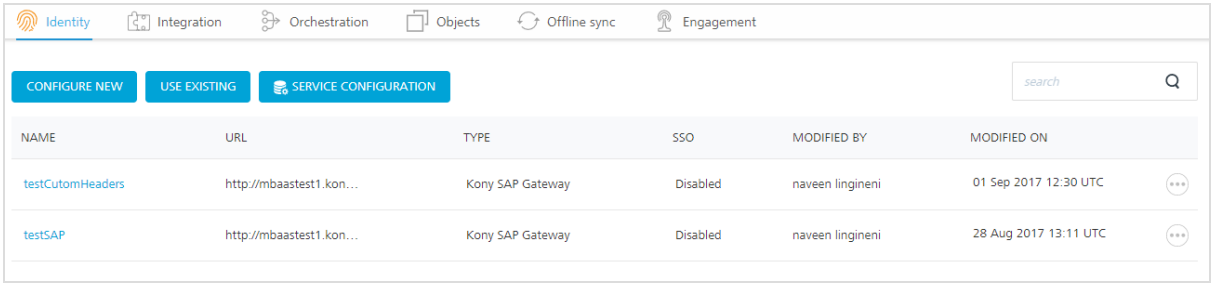
Client Specified: In **Client Specified** mode, Kony Fabric picks up the value of the custom header from the client request.

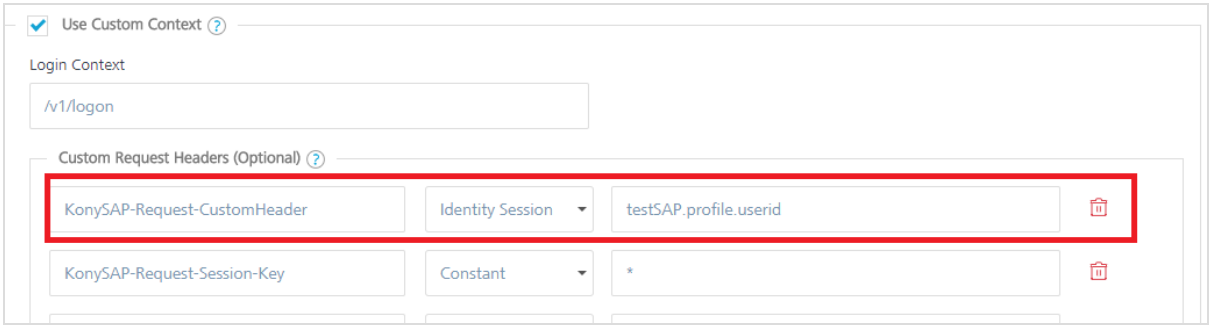


- The value field in the configuration will be disabled. When the client tries to login, the request must contain the header “KonySAP-Request-CustomHeader”, else the header is not sent as a request header to SAP.
- When there is more than one instance of the configured header in the request headers of the login request, the first value sent will be picked up.

Identity Session: In this mode, Kony Fabric picks up the value of the custom header from the identity session. The identity session contains the details of the identity providers logged into by the application. Identity session contains two different types of attributes for each provider: **Profile** and **Security**.

For this mode to work, Kony Fabric should at least have two identity providers associated with it. The value in the provider configuration will be of the following pattern:**`<Provider Name>.<profile|security>.<Attribute Name>`**.





Based on the above configuration, the login request to SAP will pick up the header value from the identity session. It will pick a profile attribute named **userID** associated with the provider of the name**testSAP**.

c. Enter additional headers that need to be present to log in to custom SAP services. The entries for Header are automatically inserted into the login request. You can delete an entry by clicking the **Delete** button next to the entry.

d. **Response Session Key Header**: Enter the header name that contains the session key from SAP response.

6. In the **Header parameter name prefix *** text box, enter **KonySAP**.

7. In **Test SAP Login**, do the following:

   a. In **User ID** and **Password**, provide valid credentials that you created while registering with Kony SAP services.

   b. In the **Default Caller ID**, provide the ID that Kony SAP Gateway uses for logging and auditing.

   c. In the **Default Caller Group**, provide the ID that Kony SAP Gateway uses for logging and auditing.

   > ***Note:*** This information is optional.

   Click **Test Login** to validate the SAP login details.

8. Click the **Advanced** to provide additional configuration of your service definition:

   - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.

   - **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.

- **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
- **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.
- **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

> ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

9. Click **Save**. The identity provider is configured.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.8 Kony User Repository Identity Service

Kony Fabric allows you to use the built-in **Kony User Repository** (User Store) identity service to authenticate your apps. It provides a compact but robust authentication solution for all users that are part of the service. Typically, you can create log-in credentials for app users, so that the service restricts the number of users accessing the application. `Admins` and `Members` are the default groups available in the Kony User Repository. A user must be part of a group. By default, all users are associated to the **Members** group.

Note: Support for Kony User Repository Identity Service is deprecated from V8 SP4 AprilFP onwards. Click here to view the V8 SP4 Document version for Kony User Repository Identity Service

You can create your own user repository by using the enhanced **User Repository** identity service type. With the User Repository identity service, you can create multiple instances in the same account. You can create custom set of users in an instance of the enhanced **User Repository** type and use them for authentication of apps. For more details, refer User Repository Identity Service.

19.4.9 User Repository Identity Service

With the Enhanced **User Repository** identity service, you can create multiple instances in the same account. An instance of the User Repository type can contain a custom set of users. You can use one service for authentication of multiple apps, use them individually, or share the service across multiple apps.

`Admins` and `Members` are the default groups available in each **User Repository** identity service instance. A user must be part of a group in an instance. By default, all users are associated to the Members group. This authentication service works in the same norms as other Kony Fabric services.

19.4.9.1 Use Case 1- Creating multiple instances of User Repository Identity Service

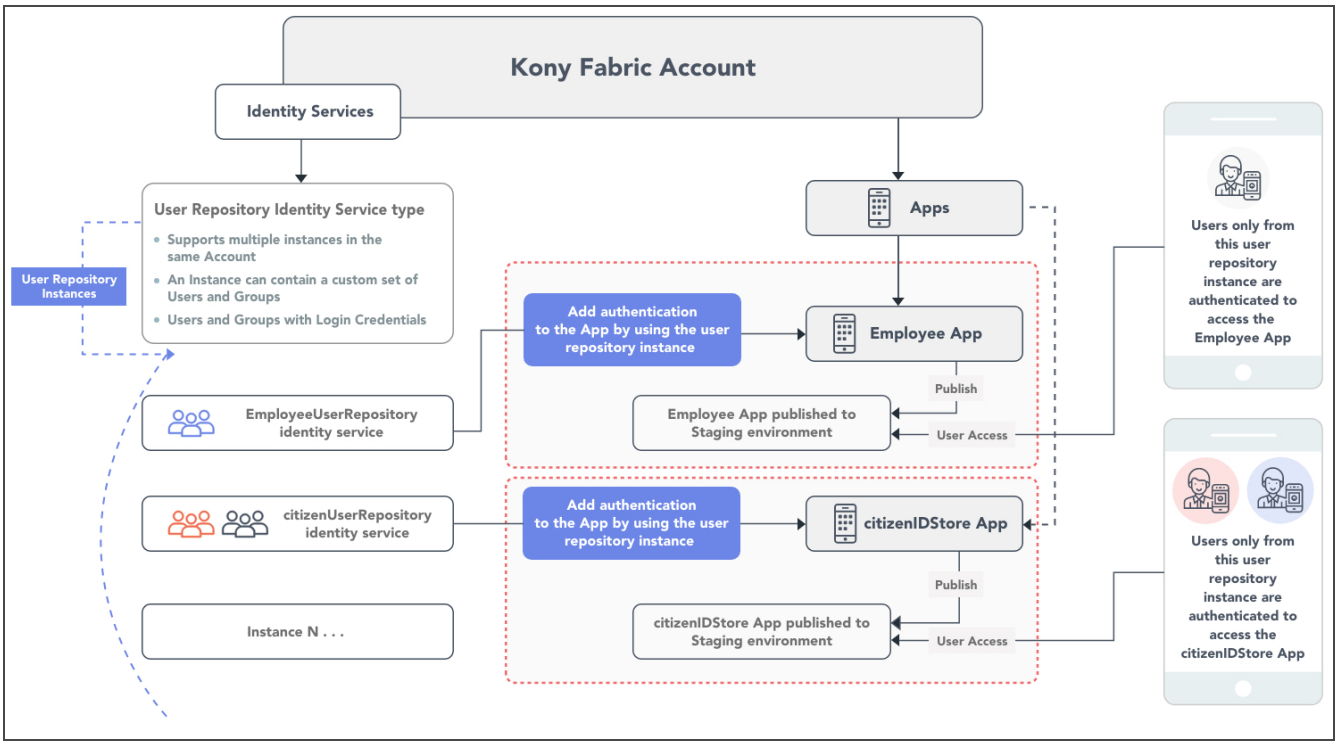
For example, you have created two apps: `Employees` and `citizenIDStore`. Now you want to create separate login credentials for these apps and publish these apps to one environment in the same account. So the individual set of users can access only the authenticated apps from that account. You can use the **User Repository Identity Service** type to achieve this.

- Here, you create `EmployeeUserRepository` identity service instance and associate it to the `Employees` app and publish the app to a `Staging` environment. Now all users in the `EmployeeUserRepository` can only access the `Employees` app from the Staging environment, in that account.
- Similarly, you create another instance of the identity service: `citizenUserRepository` and associate it to the `citizenIDStore` app and publish the app to the Staging environment.

Now all users in the `citizenUserRepository` can only access the `citizenIDStore` app from a Staging environment.

> ***Note:*** Users from a particular User Repository can only access the app associated with it.

The following flow diagram explains the usage of **User Repository** authentication in Kony Fabric.



> ***Note:*** NTLM authentication is not supported by User Repository identity service.

19.4.9.2 Creating a User Repository Identity Service

To configure an identity service using User Repository, follow these steps:

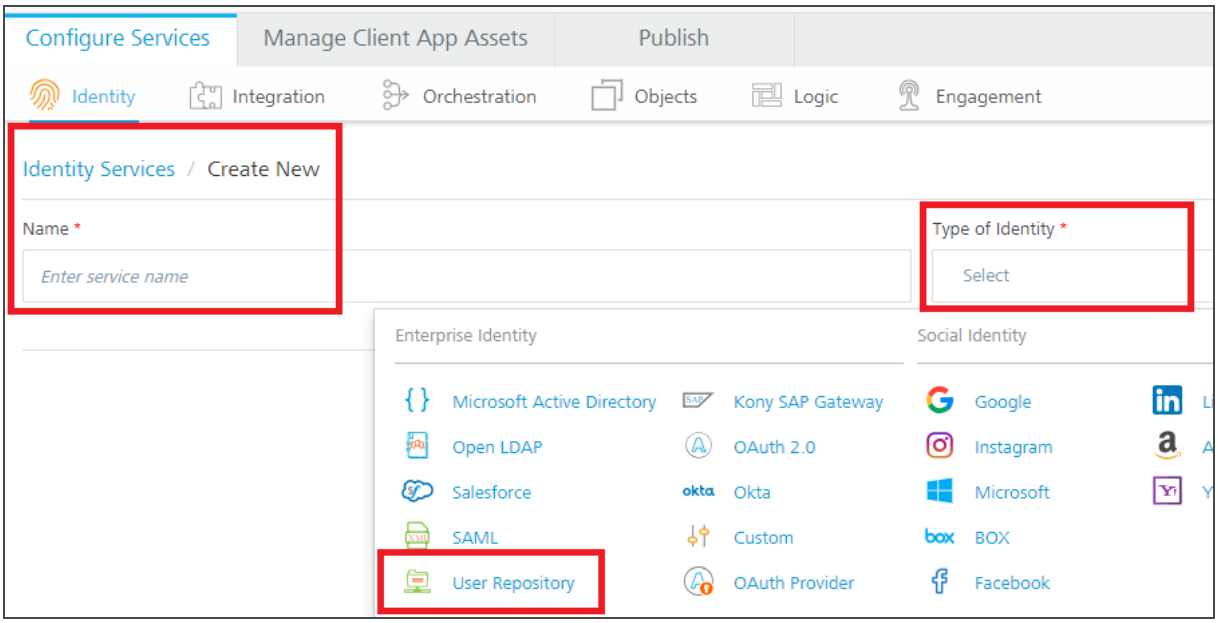
1. In Kony Fabric Console, from the left pane, click the **Apps**.
2. In the **Fabric Apps** page, click **ADD NEW**. By default, the **Configure Services** tab is selected. A new app is added, and you are directed to the **Identity** page of the new app.

3. Click **CONFIGURE NEW**.

> ***Note:*** For more details on Identity Service Designer page, refer to **Identity service designer**.

4. Specify a name for the service in the **Name** text box.

5. From the **Type of Identity** list, select **User Repository**.



6. Click **SAVE**.

7. Click the **Advanced** to provide additional configuration of your service definition:

   - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.

- **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.
    - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
    - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.
    - **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

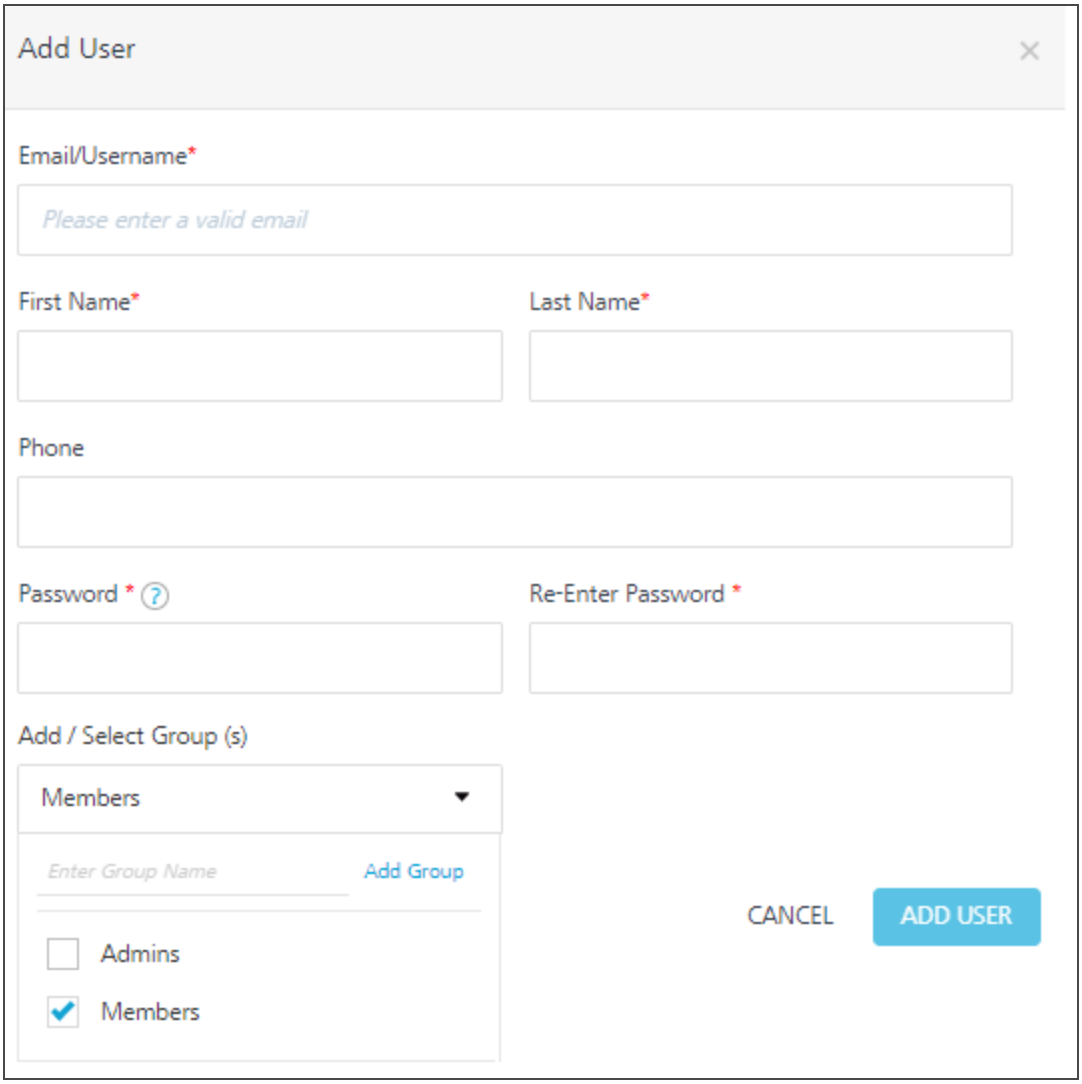
      > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

The **Users List** page appears with the **IMPORT USERS** and **ADD USERS** buttons, which you can use to add users or import users into the User Repository identity service.

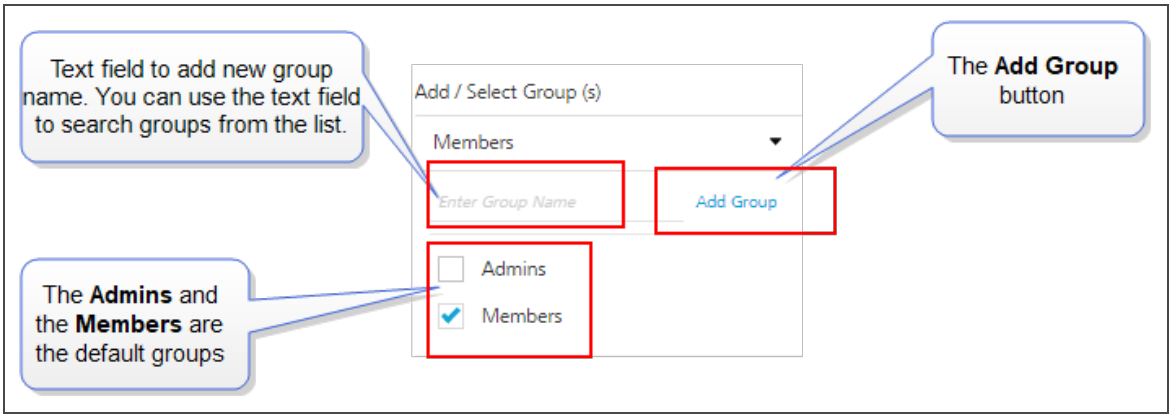
19.4.9.3 Adding a User with a Group to a User Repository

To add a user to the User Repository, follow these steps:

1. After you create a user repository identity service, click **ADD USER**.
2. The **Add User** dialog appears.



3. Provide the required details for the mandatory fields (Email, First Name, Last Name, and Password.)
4. To associate the user to a Group: In **Add / Select Group (s)**, select the required check boxes for the available groups from the list. The default groups are **Admins** and **Members**.



- **To add new group, follow these steps:**
    a. Enter the name for a new group in the text field.
    b. Click the **Add Group**. The new group is created and added to the list.

If you have not selected any group, the user is added to the **Members** group by default.

> ***Note:*** For more details on how to use Groups, refer to Using Groups in an App.

5. Click **ADD USER**.

   You can add another user by repeating these steps above in the procedure.

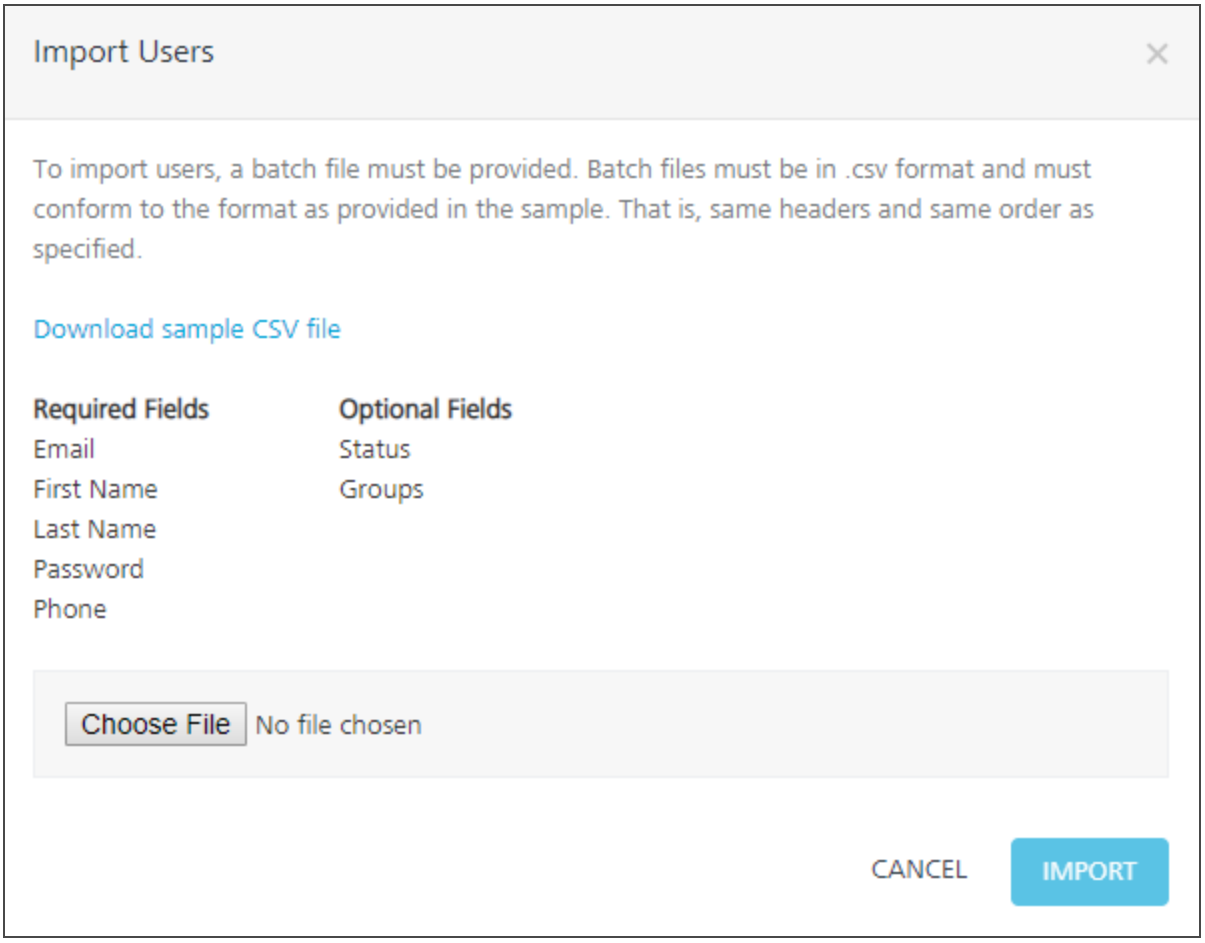
19.4.9.4 Importing users to a User Repository

You can add multiple users to the Kony Fabric console through a CSV file in the **Import Users** window.

To import users to a User Repository, follow these steps:

1. After you create a user repository identity service, click **IMPORT USERS**. The **Import Users** dialog appears.

2. Click **IMPORT USERS**. The **Import Users** dialog appears.



You can import multiple users either by using the sample template provided in a .CSV file format or similar to the `template.CSV` file. The .CSV file must include all the headers such as Email, First Name, Last Name, Password, Phone, Status, and Groups. It is mandatory that the .CSV file contains all the **Required Fields** (Email, First Name, Last Name, Password, and Phone.)

> ***Note:*** Data entered in your .CSV file should have all the following mandatory fields. You can use alphanumeric and special characters to fill the data for each field as follows:

- **Email**: Contains only a valid mail.
- **First Name** and **Last Name**:
  - Cannot contain special characters: **>, <, &, +, |, /, \, ***
  - Min size - 1 (or Empty String) ; Max size - 128
- **Password**
  - Contains only a valid password includes at least one uppercase, one lowercase, and one digit.
  - Min size - 8; Max size - 20
- **Phone**
  - Contains a white space or a valid phone number (digits 0 to 9, #, -, +)
  - Min size - 1; Max size - 20

Note: Data entered in your .CSV file can have all the following optional fields:

- **Status** Contains a white space (space or -) or a valid status such as pending, active, blocked, or disabled
  - **Pending:** Users in the pending state cannot log in or use any of the MBaaS services.

    Users enter the pending state in one of the following ways:
    - A user is imported into authentication via direct user registration to the Kony User Repository but pending confirmation from sysadmin/email verification.
    - A user is added to the system after logging in via an external provider. However, the provider's policy is set to "new users will be in pending state unless confirmed by administrator."
  - **Active:** Users can log in and use services.

    Users enter the active state by an admin/user action or due to a policy on provider as default active when logged in to MBaaS.

- **Blocked:** Users cannot log in or use any service.

  Blocked is typically an automated action by an authentication service based on policies such as nonconsecutive log-ins.

  Users can be reactivated through an admin action, with a policy that is auto-enabled after 24 hours, or via answers to secret questions.

- **Disabled**: Users cannot log in or use any of the services. This status is set by an explicit admin action.

- **Groups**
  - Contains group names (digits 0 to 9, #, -, +)
  - Can contain special characters: **>, <, &, +, |, /, \, ***
  - Min size - 1 (or Empty String) ; Max size - 128
  - If you want to specify multiple groups, use semi-colon to separate the group names. For example, Employees;Admins;Managers

3. To use the sample template CSV file, follow these steps. Otherwise, proceed to the next Step 4.
   a. Click **Download sample CSV file**. The `sample.CSV` file downloads into your local system.
   b. Navigate to the sample.CSV file and open it, and then fill in details. After filling, save the file and then close it.
4. Click the **Browse** button to browse and upload your CSV file.

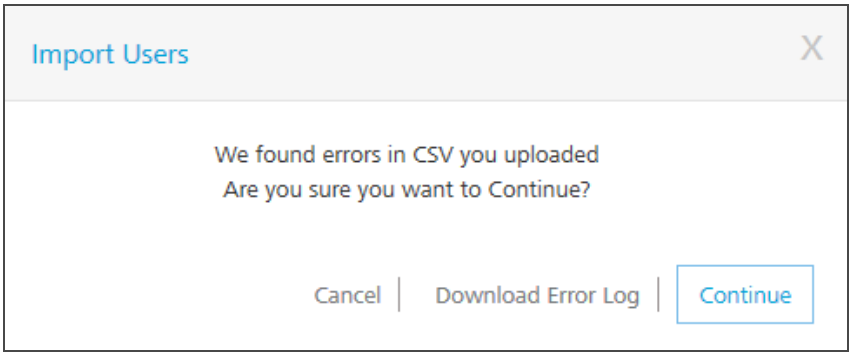
   After you select a CSV file, the system shows the file name. If no file is selected, its status is set to `No file selected`

   > ***Note:*** The **Import** button is made available only after you upload your .CSV file.

5. Click **Import** to import your data.

   The system will validate your CSV file for the following:

- If you select any file type other than a CSV file, an error will display, such as `Unsupported File type.`
- If the file size is greater than 75KB, an error will display, such as the `Selected file is too Hilarge. Size limit is 75KB.`
- If a field in the CSV file contains wrong data, the system will throw an error, as shown below:



6. To view error logs, click **Download Error Log**. The system generates an **error.csv** file, and prompts you to open or save the file.
7. To continue importing users, click **Continue**. The system imports only users with valid data into the console successfully, generates an **error.csv** file, and prompts you to open or save the file. The details of imported users are displayed in the grid list.

   > ***Important:*** If you click **Continue**, the system successfully imports only users with correct data into the console. An error.csv file contains only users fields with invalid data and corresponding error messages.

8. Click **Save**. The system saves the **error.csv** in your browser's default download location. For example, in Firefox, the system prompts you to save or open the file.

   The following is a list of error messages for each field:

Field With Wrong Data	Error Message Displayed
Email - for example, *sample_email*	• Email/User ID contains illegal characters and is invalid
First Name - for example, *&&*	• First Name must be a maximum of 128 characters • First Name contains illegal characters and is invalid
Last Name - for example, *&&*	• Last Name must be at a maximum of 128 characters • Last Name contains illegal characters and is invalid
Phone - for example, *12346753abc*	• Phone field contains illegal characters and is invalid
Password - for example, *passw*	• Password must be between 8-20 characters
Status - for example, my status	• User status **my status** is invalid. Pending, active, blocked, and disabled are valid statuses.

19.4.9.5 Exporting Users from a User Repository

You can export the existing users of the user repository to an .CSV file. The .CSV file contains users with record level data in a tabular form such as Email, First Name, Last Name, Password, Phone, Status, and Groups.

> ***Note:*** When you export users from a user repository, the content of the Password field is not exported.

> ***Note:*** Maximum 10000 users can be exported at a time.

You can import users from the exported .CSV file into another user repository identity service. Before importing the .CSV, ensure that you must fill password for all users in the .CSV file.

To export users of the User Repository to an Excel file, do the following:

1. Go to Fabric Console, and navigate to the app.
2. In the **Configure Services** > **Identity** tab of the app, click the user repository identity service. The User Repository identity services details are displayed.
3. Click **EXPORT**.

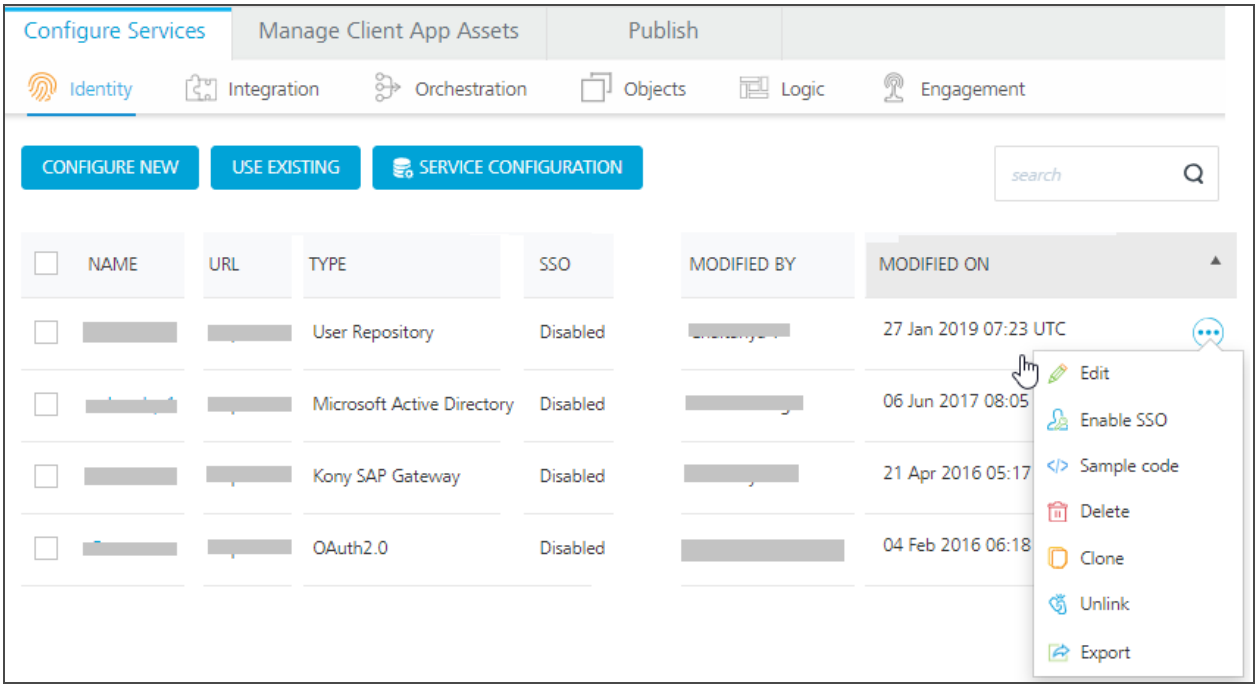
19.4.9.6 Editing or Deleting a User from a User Repository Identity Service

To edit a user from the User Repository, follow these steps:

1. From the **Identity** tab of an app, click the required user repository service. The list of users is displayed.
2. Click the **More Options** button next to the user.
    - To edit a user, do the following:
        a. To edit a user, click **Edit User Details.** The **Edit User** window appears.
        b. Enter the required details.
        c. Click the **EDIT USER** button.
    - To delete a user, do the following:
        a. To delete a user, click **Delete**. The **Delete User** confirmation window appears.
        b. Click the **DELETE** button.

19.4.9.7 Cloning a User Repository Identity Service

1. Go to the **Identity** tab of the app in Kony Fabric. The page lists the existing services (if any).
2. Click **More Options** button next to the identity service type of the User Repository.



3. Click **Clone**: When you clone a user repository identity service, all users added in the first service are present in the cloned service as well.

> ***Important:*** When you click **Clone**, the system generated new name appears for the cloned identity service, in the list. The new name remains in the edit mode until you click anywhere else on the screen. If you want, you can rename it. Changes made to a cloned identity service will not impact the original service.

19.4.9.8 Reset Password

If you have used User Repository identity service to authenticate and build your app by using MF-SDKs, and if you forgot your password to access the app, you can reset your password based on your registered email ID. Refer to Reset Password for Authentication based on User Repository Identity Service

> ***Note:*** For more information on how you can integrate Kony OAuth Provider, User Repository, and OAuth 2.0 Identity services to create a basic login form, refer to a Base Camp article: Exploring Kony OAuth Provider.

19.4.9.9 Reset Password for Authentication based on User Repository Identity Service

For apps built based on Kony Fabric SDKs and User Repository identity service, an app user can now reset the password of the app.

- For example, you use User Repository identity service for authentication in for app and build the app by using Kony Fabric SDKs. In this case, if an app user forgets the password to log in to the app, the user can reset the password based on the registered email ID.

> ***Note:*** From Kony AppPlatform V8 SP4, the forgot password functionality is available only for Kony Cloud Users.

Prerequisites

You must meet the following prerequisites:

- The client app must have a User Repository Identity Service.

How to Reset password based on Authentication for User Repository Identity Service

To reset a password, do the following:

1. After an app is published successfully, open the service document file of your app. The service document contains all configured services.

   The following is a sample code for Login, which contains meta data of an identity service including the following details:

   - `forgot_pswd_submit_userid`: URL of your reset password request.
   - `url`: URL of your Kony Cloud account

```
  "login": [
  {
   "forgot_pswd_submit_userid":
"https://00000000.auth.konycloud.com/forgot_password/submit_
userid/",
   "reset_pswd": "https://000000000.auth.konycloud.com/forgot_
password/reset_password/",
   "alias": "myRepo1",
   "type": "basic",
   "prov": "myRepo1",
   "url": "https://000000000.auth.konycloud.com"
  },
```

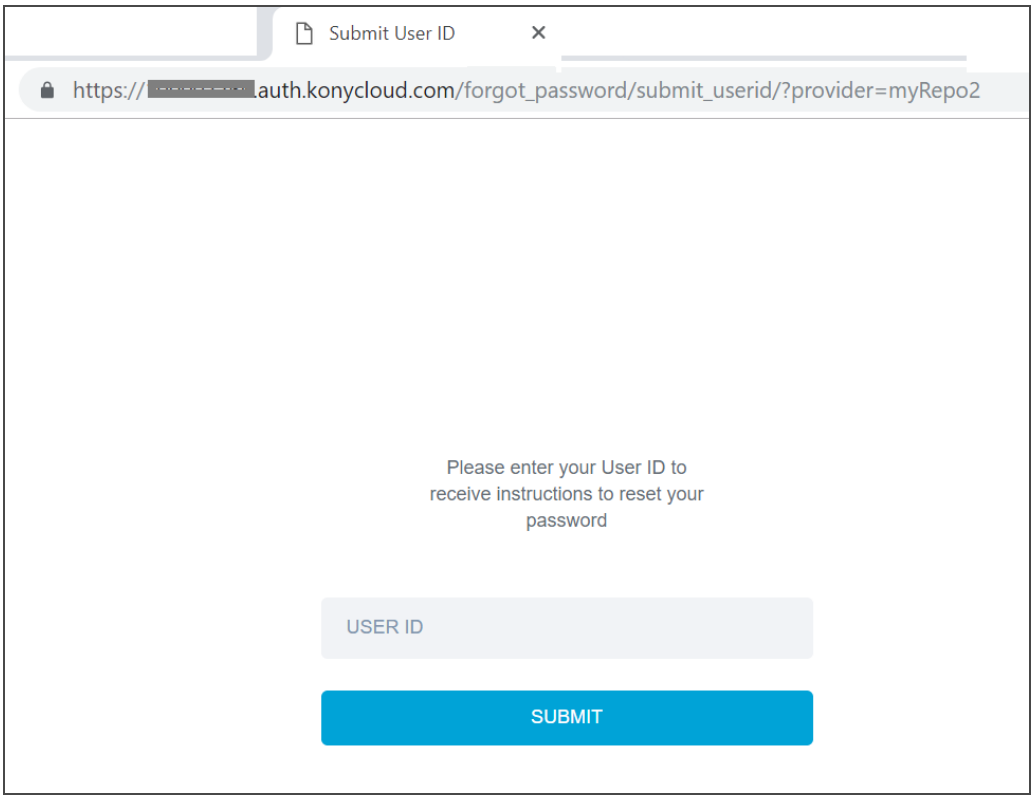
2. Specify the identity **providername** as a query parameter to the URL.

   For example, if an app user forgets the login password, the user can reset the password using the URL set for the Key: `forgot_pswd_submit_userid` from app service document.

```
 https://<mytenantID.auth.konycloud.com/forgot_password/submit_
userid/?providername=userstore
```

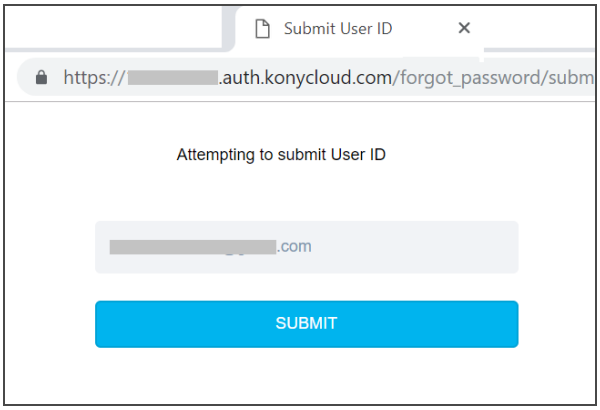
- **Userstore** is the name of an identity service.
- If you create a user repository identity service, you can provide the name of the service as **providername** in the query parameter.
  For example, `"prov": "myRepo1"`

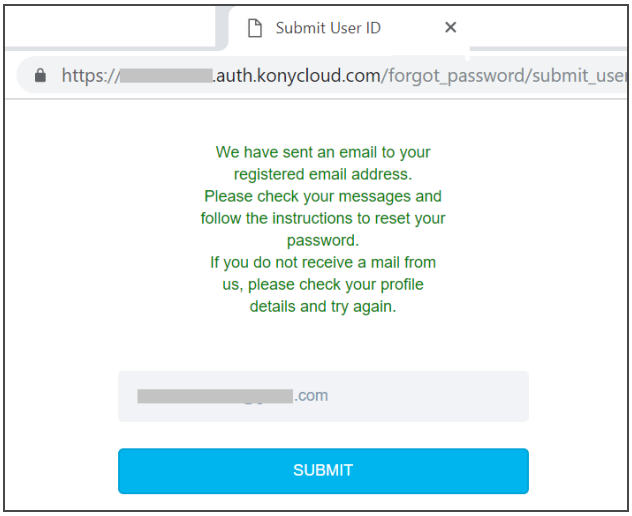
3. Go to the **reset_pswd** URL by using a browser. The **Submit User ID** screen appears.



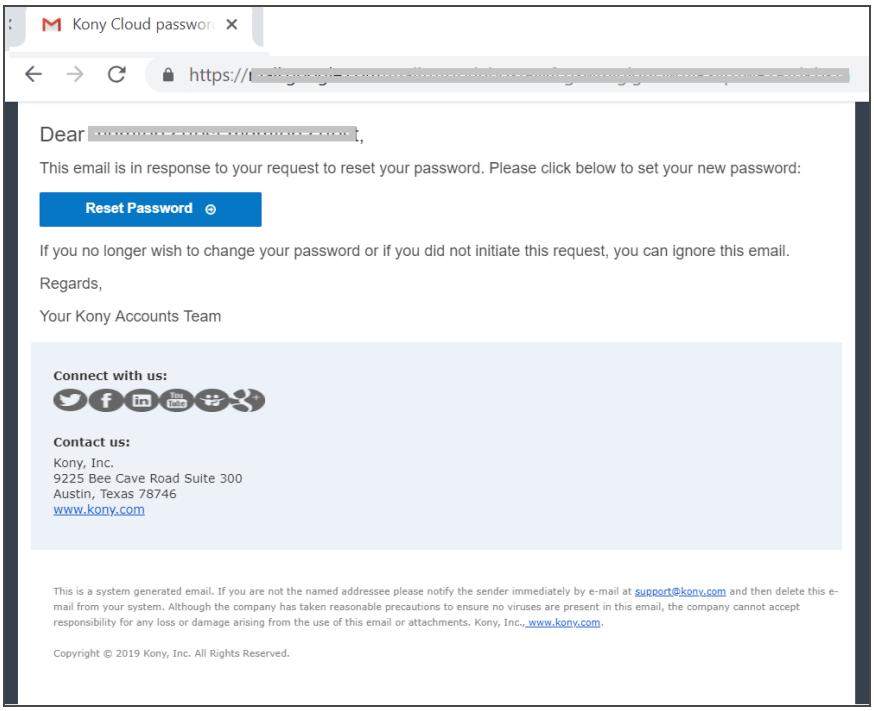
4. Specify your User Id.

5. Click **SUBMIT**. The reset password mail process starts and sends an email to your registered email account.

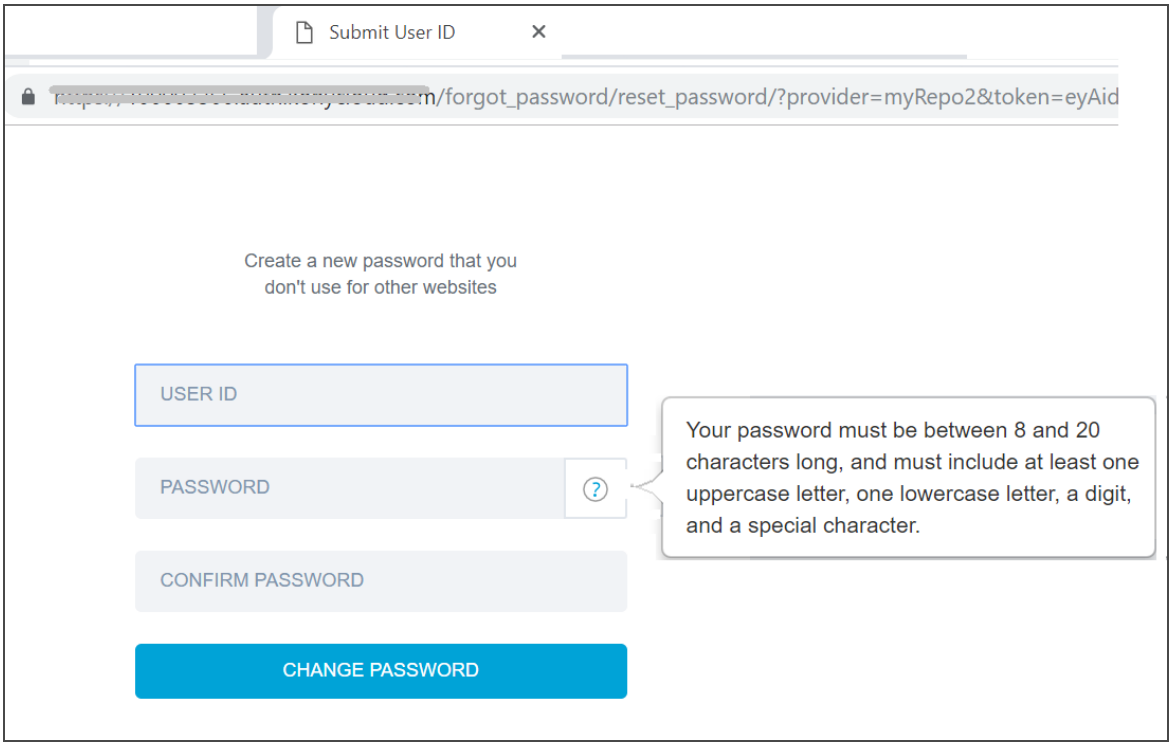




6. Click the **Reset Password** link from the email that you received.



The new password window appears.

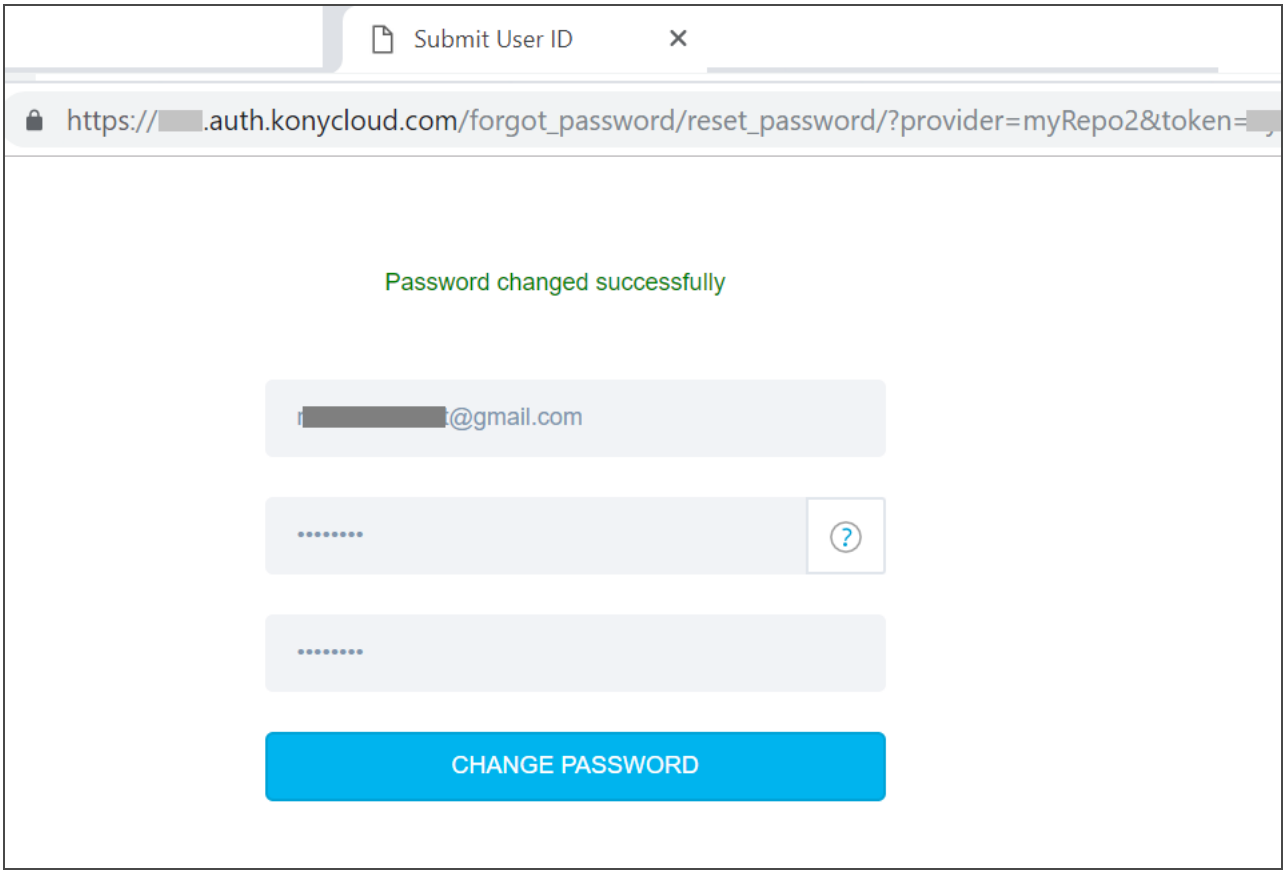


7. Specify the following details:
    - **User ID**: Email of a user.
    - **Password**: Enter a new password for your account pertaining to the password criteria.

      > ***Note:*** Your password must be between 8 to 20 characters long, and must include at least one uppercase letter, one lowercase letter, a digit and a special character.

    - **Confirm Password**: Re-enter your new password for the confirmation.

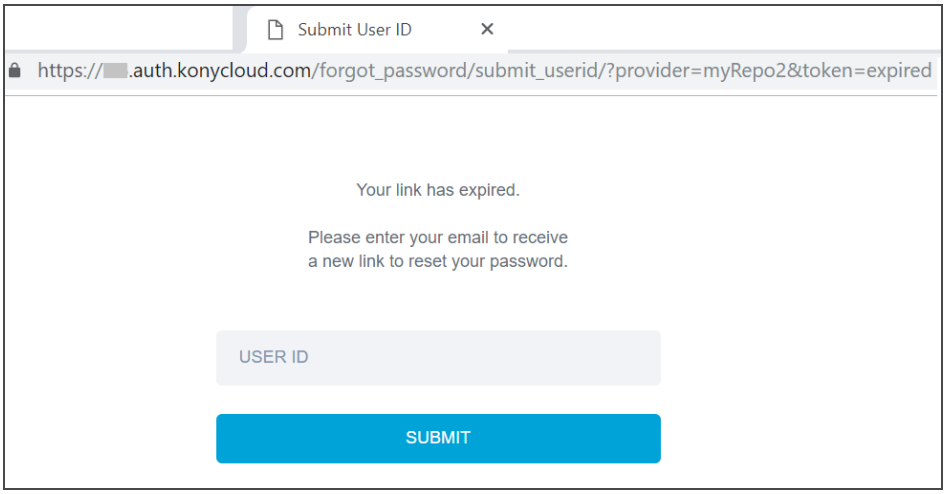
8. Click **CHANGE PASSWORD**. A confirmation window is displayed saying, **Password Changed Successfully**.



Characteristics of Reset URL

The following are the characteristics of the reset URL:

- Forgot Password URL is valid for one-time use.
- Forgot Password URL is valid for one hour. Using old forgot password links throw an error:



- Forgot Password URL must not be formed with the user request input.

Limitations

- A Reset Password link is sent to your registered email ID. The link is valid for 1 day. A reset password link can be used only once.

19.4.9.10 Sign-Up for Kony User Repository

Sign-up is a new capability of Kony User Repository that allows the users to signup or register to the application by using their e-mail ID. Until V8 SP4, only an admin can add the users in the Kony User Repository. From V9 onwards, the users can directly signup by providing their user details.

Configuration of the Sign-Up User Flow

You can configure the sign-up feature in two ways.

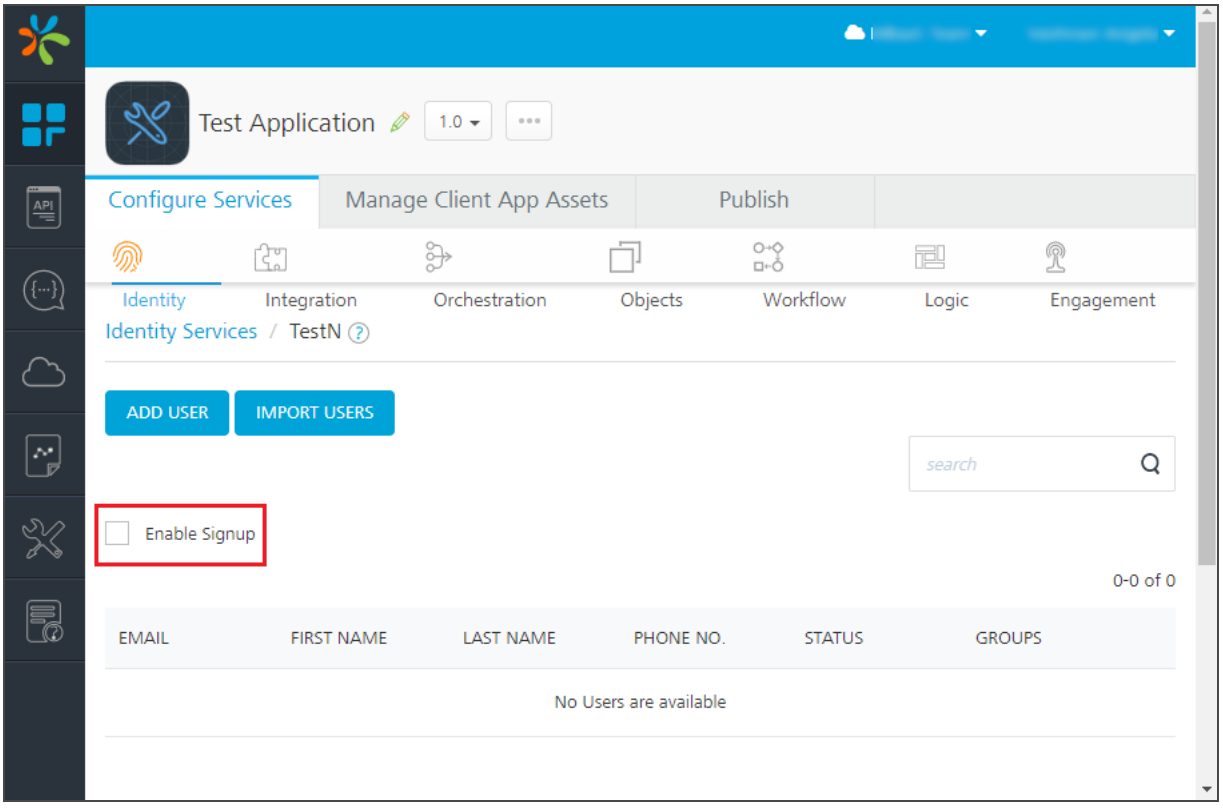
- Invoke the Sign-up API in the Visualizer application.
- Configure the User Store Adapter.

Invoking the Sign-Up API in the Visualizer Application

To enable the sign-up feature by invoking the Sign-up API, follow these steps.

1. Sign in to Kony Fabric Console.
2. Create a User Repository Identity Service.

   For more information about how to create an Identity service by using the User Repository identity type, click here.
3. In the Identity tab, select the **Enable Signup** check box and publish the application.



4. Now, create a project in Kony Visualizer.
5. Link your Fabric Application to your Kony Visualizer project.

6. To enable the sign-up feature, invoke the `identitySvc.register` API with the user details in Visualizer with the user details.

19.4.9.11 Register API

The Register API registers or signs up a new user asynchronously and executes the given callback.

Syntax

```
register (params, successCallback, failureCallback, options);
```

Parameters

Parameter	Type	Description	Required
params	Object	Details of the user such as userID, password first name, and last name.	Yes
successCallback	Function	Method invoked on success.	Yes
failureCallback	Function	Method invoked on failure,	Yes
options	Object	Map for optional parameters.	Optional

Sample Code:

```
//Sample code to register a user to an Identity Service.
//Currently available for Kony User Repository i.e. userstore
var serviceName = "userstore";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var registerParams = {
    "userid": "userID",
    "password": "password",
```

```
    "first_name": "<user's-first-name>",
    "last_name": "<user's-last-name>",
    "phone": "<user's-phone-number>"
};
identitySvc.register(registerParams, function(response) {
    kony.print("User Registration Successful: " + JSON.stringify
(response));
}, function(error) {
    kony.print("User Registration Failed: " + JSON.stringify(error));
});
```

Configuring the User Store Adapter

To enable the sign-up feature by configuring the User Store Adapter, follow these steps.

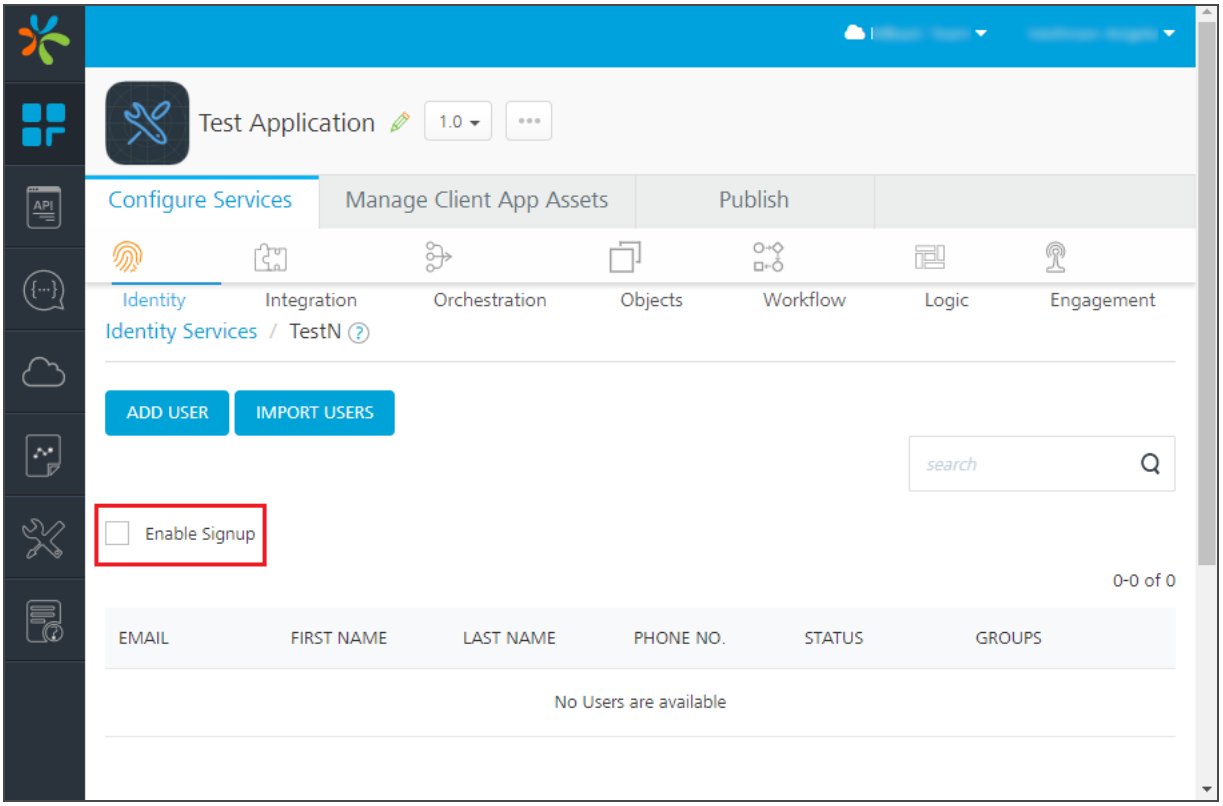
1. Import the User Store Adapter from Kony Marketplace to your Kony Fabric Console.

   For more information on how to import an adapter from Marketplace, click here.

2. Create a User Repository Identity Service.

   For more information about how to create an Identity service by using the User Repository identity type, click here.

3. In the Identity tab, select the **Enable Signup** check box.



4. Configure an Integration service by using the adapter and type your User Repository Service Name in the Provider Name. The functionalities of the user store adapter are available as Integration service operations.

5. In the sign-up operation, pass the user details and publish the application.

User Flow to Signup to the Application

1. User chooses the Signup option. A User Details page appears.

2. The user types the following details and clicks Submit.
   - First Name
   - Last Name

- E-mail ID
- Password

3. Now, the user receives an activation email to the provided email ID. The email contains an activation link.
4. After the user clicks on the activation link, he/she becomes an active member in the User Store and can sign in to the application using their credentials.

Error Codes

The user might encounter the following errors while signing up to the application:

Error Code	Error Message	HTTP Status	Description
-12	UnSupported Operation	400	This error occurs when the API is invoked in the On-prem case.
-4	Invalid User Credentials	400	This error occurs if the user details are invalid or missing.
-70	Multiple users with same username are found	409	This error occurs when the registered user receives the activation email again.
-7	Invalid Provider	400	This error occurs if the provider is not found.
-119	Provider type is not userRepository	400	This error occurs if the provider of non-user repository is mentioned.
-27	given provider not exist in app services	400	This error occurs if the validation of the token fails.
-5	Token invalid	401	This error occurs if the token is invalid.

Error Code	Error Message	HTTP Status	Description
-6	Token expired	401	This error occurs if the token is expired.
-1	Uncategorized Failure occured	500	This error occurs in case of any unexpected failure.
-120	The Attempt to send email has failed	500	This error occurs if the email is not sent due to some reason.

19.4.10 Using Groups in an App

You can now create groups by using User Repository identity service. Admins and Members are the default groups available in the User Repository identity service. Users can create custom groups as well. When a user is added, the user must be part of a group. By default, all users are associated to the **Members** group.

Based on the groups you create in User Repository and the response in identity service, you can now directly handle your logic/code in controllers created in MVC (modules in non-MVC) platform, for apps. Your logic/code can include selected functionality targeted to a specific set of users in a group.

> ***Note:*** For configuring groups for other identity service types, refer Groups Support for Identity Services.

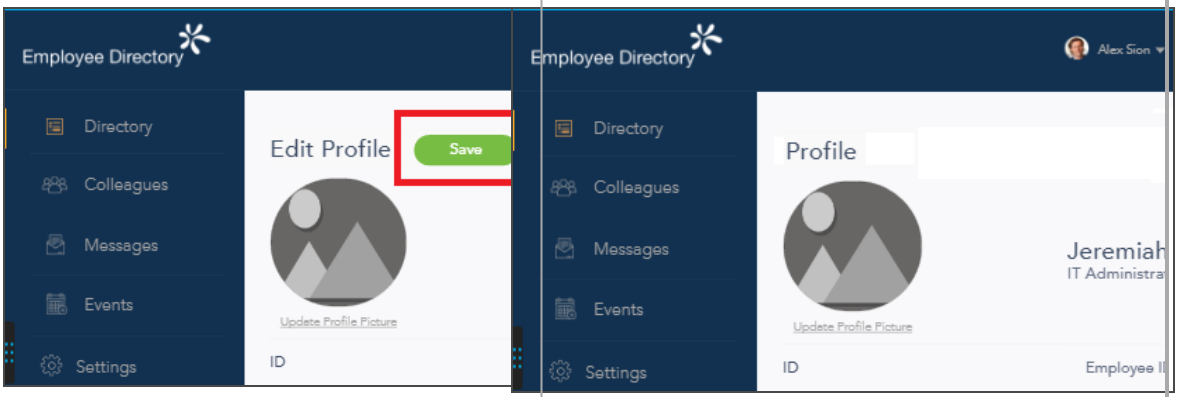
19.4.10.1 Use Case

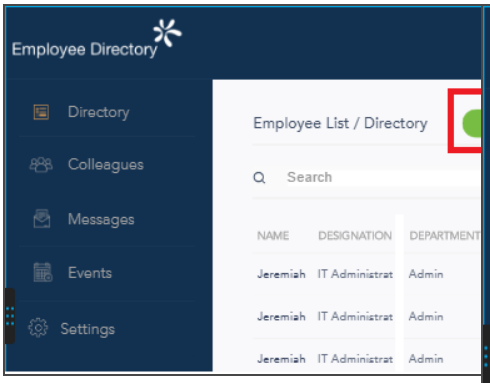
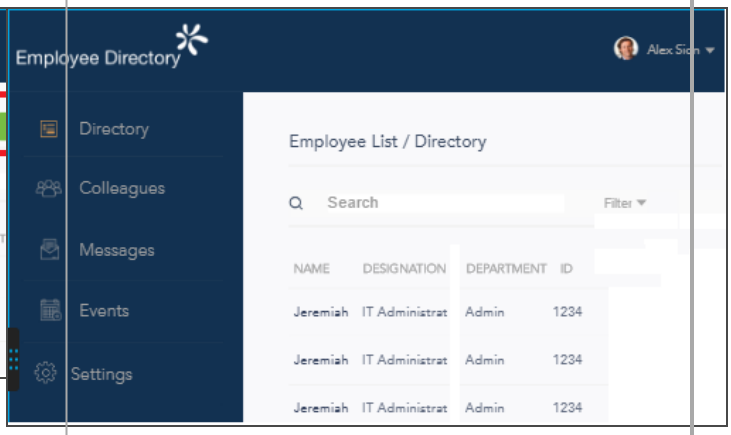
To understand the functionality of **Groups** feature, let us take an example of the sample app KonyEmployeeDirectory. You will learn how to handle groups to authenticate CRUD operations in user forms in the app.

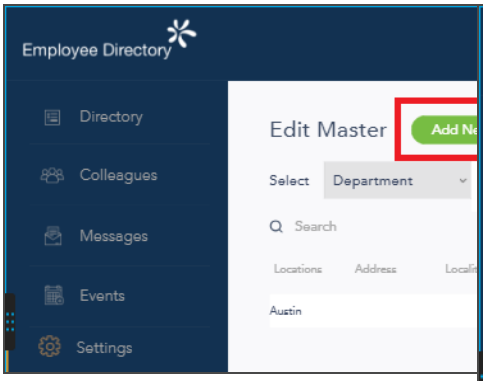
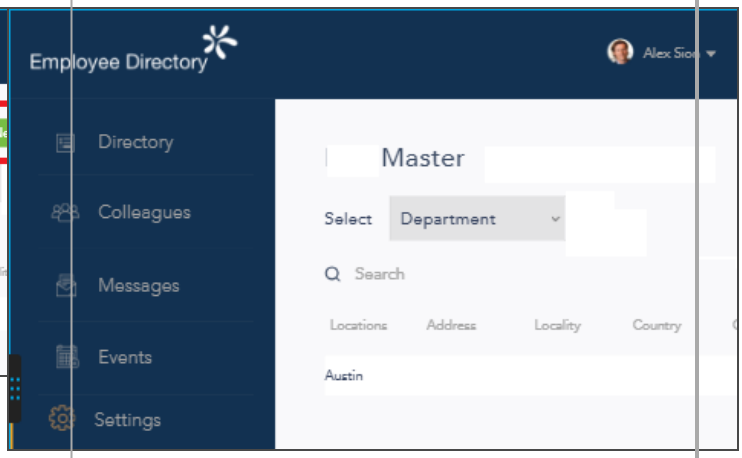
- **Desktop web version (Users of Member group log in/Client app)**: The app comes pre-configured with Login, Menu, Filter and List/Detail interface with Search and Dictionary functionality. Users in the Member group can search, discover and know more about your co-workers.

- **Desktop web version (admin only)**: In addition to the capabilities of the member role, users in the Admin group can create, edit, save, and delete employee records. The responsive web app comes with an Admin Console which lets the user to perform CRUD operations.
    - For example, in the sample app: **KonyEmployeeDirectory** is linked to the User Repository identity service. You create one set of users and associate them to a group: Members, and another set of users to the Admin group. Based on the response from the identity service, now you can write your logic/code for admin app and client app.

The following table details the forms used in the KonyEmployeeDirectory sample app versions authenticated and authorized based admin and member groups.

Forms available to users in the **Admin** group (allows **CRUD** operations)	Forms available for users in the **Member** group (allows **Read-only** operations)
 Form: Employee Details Navigation: Directory; Default launch screen • to allow **CRUD** operations to manage the User profiles	

Forms available to users in the **Admin** group (allows **CRUD** operations)	Forms available for users in the **Member** group (allows **Read-only** operations)
  Form: Employee List / Directory Navigation: Directory; Default launch screen • to allow **CRUD** operations to manage users in the Employee List	

Forms available to users in the **Admin** group (allows **CRUD** operations)	Forms available for users in the **Member** group (allows **Read-only** operations)
 Form: Edit Master Navigation: Directory; Default launch screen • to allow **CRUD** operations to manage details for department, designation, and location in the Employee List	

19.4.10.2 How to Use Groups in Apps (Based on Sample App - KonyEmployeeDirectory)

1. Import the KonyEmployeeDirectory app into your project in Kony Visualizer.
2. Navigate to app to the Kony Fabric services by using the Data panel.

   By default, Kony User Repository (Userstore) identity service is linked to the app.
3. Create a user (User1) and associate the user to the default group **Member**.

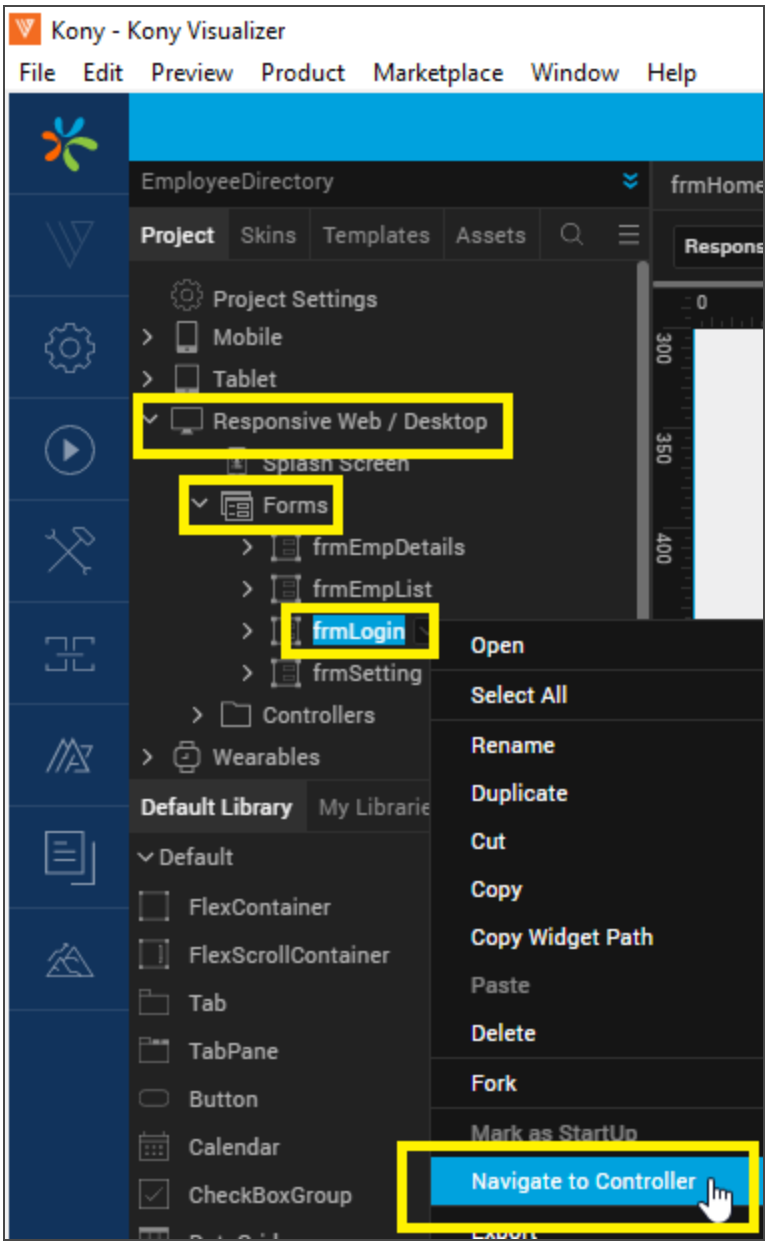
4. Create another user (User2) and add the user to the new group **Admin**.

   Sample Test Response:

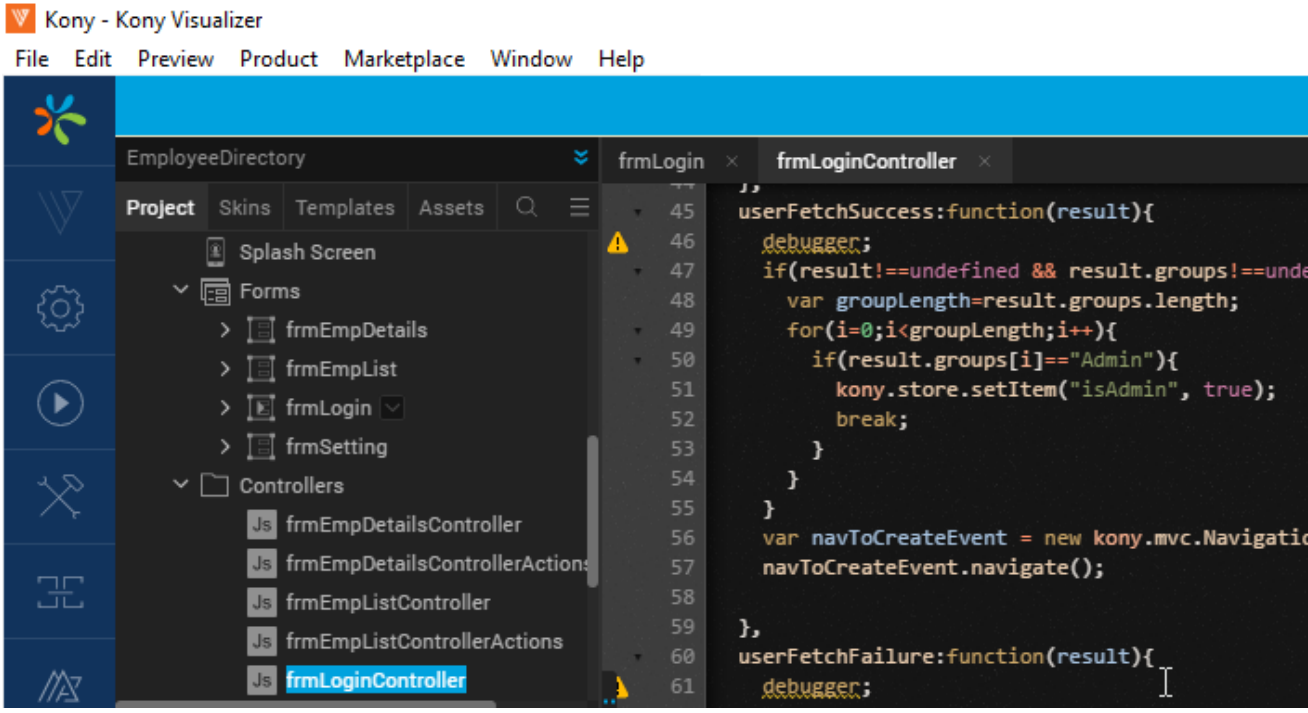
```
{
   "profile": {
       "userid": "User1@kony.com",
       "email": "User2@kony.com",
       "firstname": "User_Admin",
       "lastname": "Das",
       "user_attributes": {
           "user_id": "User1@kony.com",
           "groups": [
               "Admin"
           ]
       },
       "profile_attributes":
   },}
```

5. In **Kony Visualizer**, link the Kony Fabric app to your project.
   a. Open the project.
   b. Navigate to **Responsive Web/Desktop > Forms**, and right-click **frmLogic**.

c. Click **Navigate to Controller**.



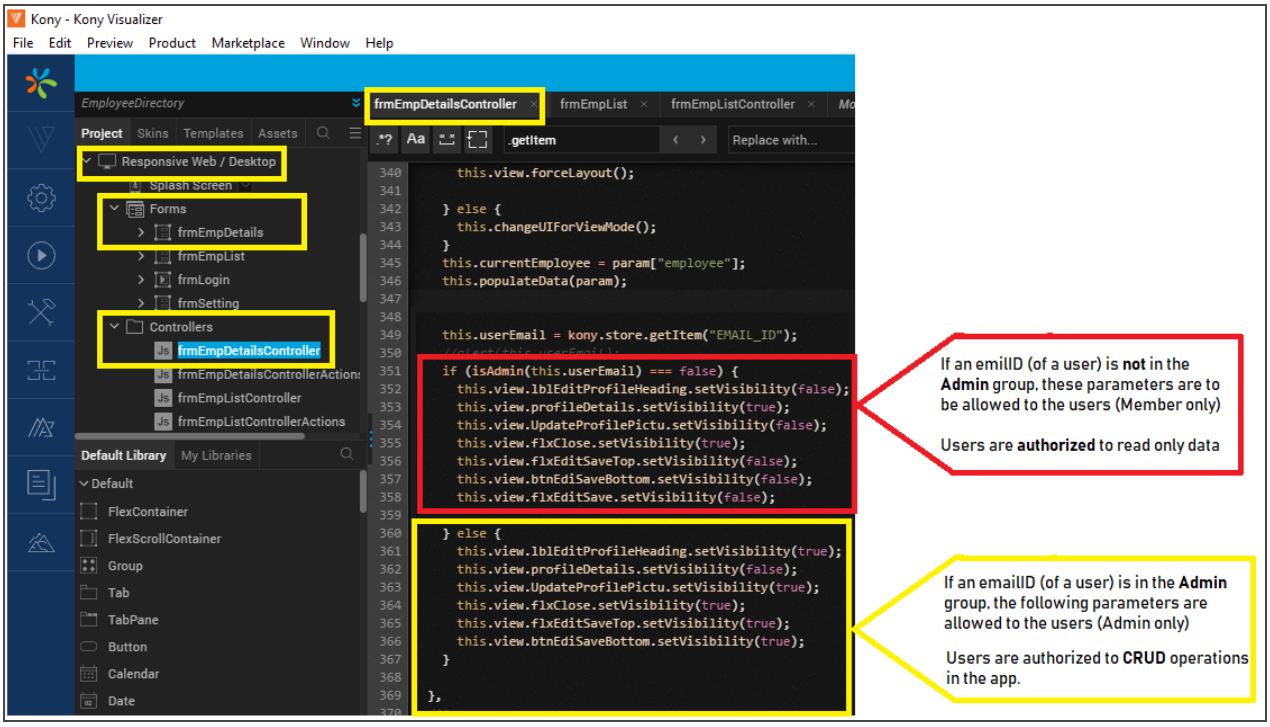
The **frmLoginController** module is selected. The frmLoginController details sample code to fetched groups details that you configured in Kony Fabric User Store:



Sample Code	Description
`if (result.group[i]=="Admin") kony.store.setItem("isAdmin", true);`	The Admin is the group in User Repository, which can contain users. You can write your logic to allow these users to perform CRUD operations in the Admin app version.
• In this case, `(result.group[i]=="Admin")`, the logic checks if the result in the response from user repository contains **Admin** under the `groups` tag, • In this case: `(kony.store.setItem("isAdmin", true)`, if you set the group is true, the group details are stored in the array: **group[i]**	

6. Navigate to Responsive **Web/ Desktop > Forms**, and right-click **frmEmpDetails**. The **frmEmpDetailsController** module contains your code/logic to access CRUD operations than

the user (Admin) is authorized to perform, such as view user details, create a new, update user details, and delete a user.



7. Build the app.

19.4.11 Facebook Identity Service

With a Facebook identity service, a user can access Facebook resources through Kony Fabric applications. Kony Fabric supports OAuth2.0 authentication mode for Facebook identity service.

For example, when a user logs into a Kony Fabric application by using the Facebook identity provider, the Kony Fabric application redirects the user to log in to a Facebook account. After the user logs into Facebook, Facebook issues an authorization code to the user. The user sends the authorization code to the application. Kony exchanges the authorization code for the access token and issues Kony token. By using Kony token, the user can access the application or web resources.

19.4.11.1 How to Configure a Kony Facebook Identity Service

To configure a Facebook identity service using OAuth 2.0 authentication mode, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Facebook**.

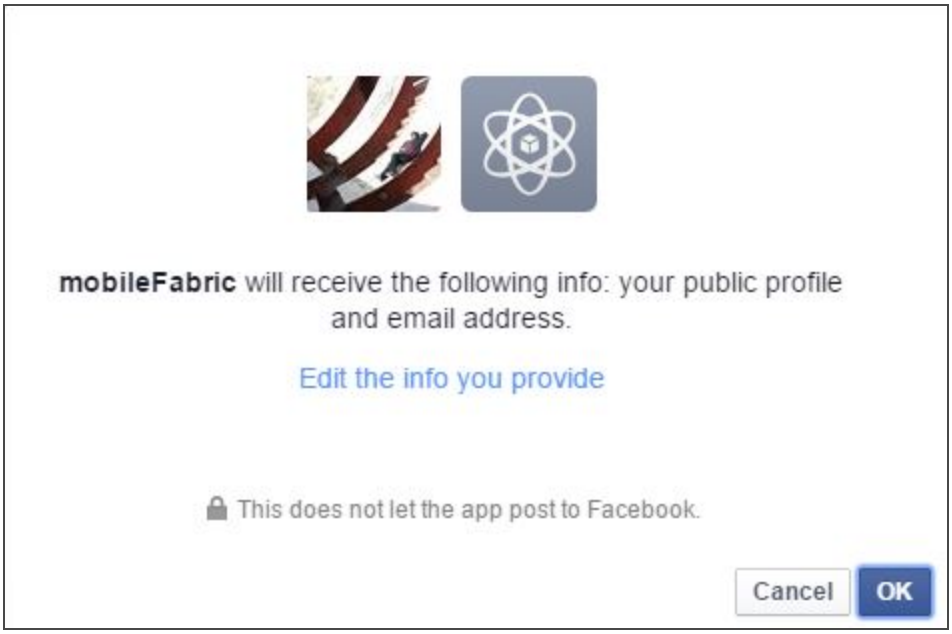
   > ***Note:*** Fields marked with an asterisk are mandatory.

3. In the **Client Id** text box, enter the app key of the app instance that you created on Facebook.
4. To move the cursor to the next text box, press the **Tab** key or click in the text box. In the **Client Secret** text box, enter the app secret key of the app instance that you created on Facebook.
   - In the **Login URL** text box, the default URL (https://graph.facebook.com) is displayed. You cannot modify these details.
   - In the **Callback URL** text box, the default URL is displayed. You cannot modify these details.
     `https://<accountID>.auth.konycloud.com/OAuth2/Callback`.

     > ***Important:*** Configure your Facebook app to accept requests from authentication service by typing the `auth.konycloud.com` in the **App Domain** text field.

5. In the **Scope** text box, enter the list of permissions that a user needs to provide while in the user Facebook log-in page. The list can contain more than one permission. The list must contain a comma-separated valid permission. The list of permissions accepted by Facebook does not contain spaces. For the full list of permissions, refer to Permissions with Facebook Login.

For example, if you configure the `email` permission as scope, after you log in Facebook for the first time, Facebook displays the following **Log in with Facebook** dialog with the configured permissions. Click **OK** to share your permissions (public profile and email) with Kony Fabric:



6. After entering the above details, click the **Test Login** button to verify the credentials. The test results are displayed in the **Identity Response** dialog.
7. After entering the details in the **Scope** field, click **Save** to save the service. The system displays the **Identity** page.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.12 Kony Custom Identity Service

Kony Fabric identity service supports federating authentication and authorization with external identity services over standard protocols.

> ***Note:*** A **Federated Identity** is an electronic identity that maps attributes of a person across multiple identity services.

To authenticate users before accessing Kony Fabric services, Kony Fabric supports various types of identity services, such as Microsoft Active Directory, Salesforce, SAML, SAP, Kony SAP Gateway and User Repository. Each of these identity services agrees on a specific input and output format for authentication. The input and output formats differ based on back-end providers.

With Kony Fabric's custom authentication, a user can log in to any back-end provider's custom protocols by providing any parameters that the back-end provider supports, such as a userid and password, or a secret key.

> ***Important:*** Custom identity service APIs should support application/ x-www-form-urlencoded.
>
> For example, when you send user input (username and password) to an auth service, the input is sent as JSON input, shown below:
> { "userid" : "user@user.com",
> "password" : "password" }
>
> For a custom identity service, user inputs are sent as / x-www-form-urlencoded.

> ***Important:*** SDK support for custom auth provider is available from Kony Fabric 6.0.4 onwards.

You can configure a custom auth service by pointing to an end point that implements the custom identity protocol. A custom identity protocol can be implemented by using Kony Fabric integration service or by an external service.

> ***Note:*** A custom identity protocol is implemented by a third-party back-end provider, or configured as an integration service by using the Kony Fabric Console.

- **Custom identity protocol is implemented by an external service:**

  A third-party back-end provider directly authenticates Kony Fabric users configured for a custom identity service based on an agreement. Based on Kony Fabric's API contract, a back-end provider must agree to implement `login` and `logout` APIs in a format compatible with Kony Fabric's auth service. The login and logout APIs accept parameters that a back-end provider supports, such as a userid and password, or a secret key for configuring a custom identity service.

- **Custom identity protocol is configured as an integration service:**

  If a back-end server **does not support** login and logout APIs, you can configure a custom auth service by pointing to an end-point.
  For example, A Kony Fabric user creates an integration service that has a login and logout operations. Based on the input and output parameters configuration, Kony Fabric fetches a response for a user for login and logout operations. To get custom back-end server's error codes and error messages in the login/logout response, a developer must map error codes and messages with the parameters `backend_error_code` and `backend_error_message` in the Integration service **Response Output**. The following steps explain how to configure a custom auth service by pointing to an end-point.

- **If the back end expects headers from an input request to be passed, follow these steps:**

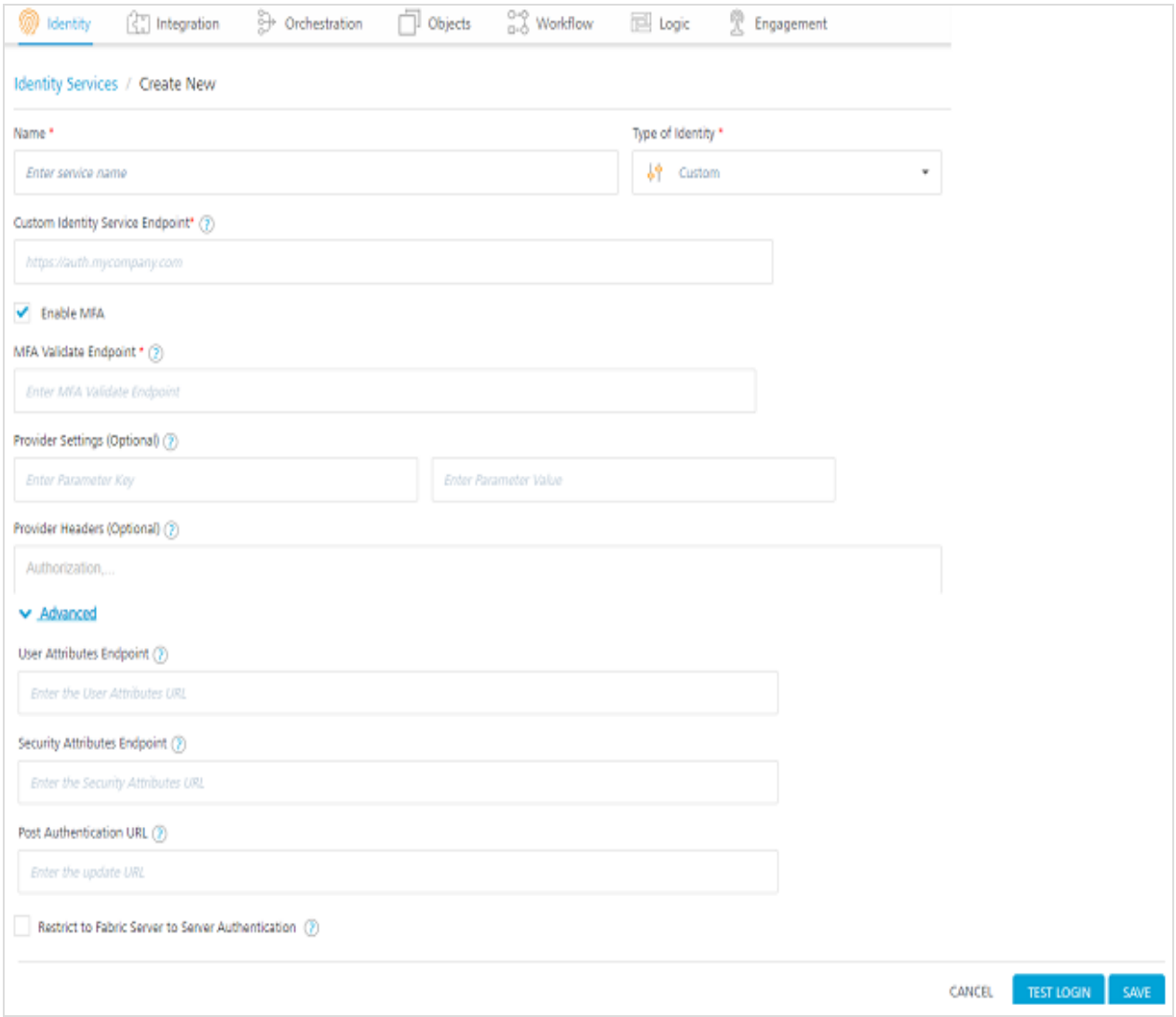
  Currently, Kony Identity service cannot pass the headers from input request directly to the back end. Any headers apart from the appkey, appsecret, and other X-Kony headers will be dropped by the Kony Identity service. To solve the issue of dropped headers, follow these steps:

  1. Configure integration service operation (for example, login) to talk to the back end and define the headers in the service definition.
  2. In the custom identity creation, define the integration service URL as login endpoint.
  3. Pass the same headers (that you defined in step 1) as body params instead of sending them as headers in the runtime app while making a call to the Kony Identity service.

19.4.12.1 How to Configure Kony Custom Identity Provider

To configure Kony custom identity provider, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Enter Service Name** text box.
2. From the **Type of Identity** list, select **Custom**.



> ***Note:*** Fields marked with an asterisk are required.

3. In the **Custom Identity Service Endpoint** text box, choose one option:
   - Enter a custom identity endpoint implemented by an external service. The endpoint should be in compliant with the custom identity provider API contract. The custom identity protocol implemented by an external service is the direct URL of a back-end provider. For more details, see the Custom identity protocol by an external service topic.
   - Enter a custom identity endpoint configured as an integration service. A custom endpoint URL (Kony Fabric's custom identity protocol) is available in the `App Service Document` after you publish an app. The app should contain an integration service with login and logout operations. For more details, refer Configuring a Custom Identity service by using an Integration service.

     > ***Important:*** A custom endpoint URL is available from a successfully published app. You must use a custom endpoint URL for a custom identity service within the same app.

4. The **Enable MFA** check box enables you to configure MFA for the Custom Identity Service. However, when configured, whether MFA is used for a particular identity service call is determined at runtime based on the response for the login service.
   This check box is selected by default. This helps you in the Test Login flow in design time environment to test the status of the short-lived known_user token (whether it is issued or not). You can uncheck it if MFA is not required.

5. In **MFA Validate Endpoint**, type the URL to which the Identity Service should send the request to validate the MFA key and get the authenticated token. This feature is linked to the **Enable MFA** check box. If the **Enable MFA** check box is selected, this field is mandatory. If the **Enable MFA** check box is unchecked, this field is disabled.
   If you give the MFA endpoint details, this adapter will act as a Multi Factor Authentication (MFA) adapter in design time environment. For more information, refer Custom Identity MFA Flow.

6. In the **Provider Settings (Optional)** section, click the **Add** button.

   > ***Note:*** When you enter text in the **Enter Parameter Key** text box, a new row is automatically added below the first row.
   > You can also delete an entry by clicking the **Delete** button that appears next to each entry.

7. In the **Enter Parameter Key** text box, enter the parameter you want to configure as an additional attribute. For example, <Backend-config-key>.

8. To move the cursor to the next text box, press the **Tab** key or click in the text box. In the **Enter Parameter Value** text box, enter the value for the parameter. For example, <backend-config-key-value>.

   > ***Important:*** You can add any key-value pairs. When an identity service makes a login call to a back-end provider, the identity service sends key-value pairs along with the call request. Confidential key-value pairs can be added under the **Provider Settings (Optional)** section instead of by adding these details in a client app.

9. In the **Provider Headers (Optional)** section, enter the headers from the login request to be stamped on the requests made to the endpoint.

10. In the **Advanced** section, you can do the following:

    - In **User Attributes**, type the URL that you want to invoke after an identity session is created to get the custom user attributes created for the user who is trying to login to the client app. The custom user attributes can be any information related to the user like DoB, Gender, and more. This field is optional.
      If you provide the URL, any additional user attributes that were configured will be appended to the respective section in the login response after a successful login

    - In **Security Attributes**, type the URL that you want to invoke after an identity session is created to get the custom security attributes created for the user who Is trying to login to the client app. The security attributes are stored in Kony Fabric and not shared with the users. This field is optional.

Any secure info like credit card number, SSN which need not be sent to client app, but is required for backend processing, must be sent as security attribute.

- In **Post Authentication URL**, enter the URL to be invoked after an identity session is created. After this URL is invoked, the Identity response is returned to the Client.
- Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.
- **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.
    - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
    - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.
    - **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

      > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

11. After entering the above details, click the **Test Login** button to verify the credentials. The **Test**

Login dialog appears.

a. Enter the required details in headers and body for custom identity provider.

The entries for Header and Body are auto-inserted into the login request. You can delete an entry by clicking the **Delete** button next to the entry.

b. Click **Sign In**.

The test results are displayed in the **Identity Response** dialog.

12. Click **SAVE** to save the service. The system displays the **Identity** page. The custom identity service is configured.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.12.2 Custom Identity Agreement

The following APIs must be implemented by a custom identity hosting service as the `endpoint_url` configuration parameter:

- Login API
- Logout API
- MFA
- User Attributes
- Security Attributes

Login API for a Custom Identity Service

The Login API authenticates user access to a custom identity provider. This API should comply with the custom identity protocol mentioned in this section.

Method

POST

Headers

X-Kony-RequestId: <Request Id>

Content-Type: application/formurl-encoded only

Accept: application/json or application/xml

Sample Request Body:

```
 {
    “<custom_key1>” : <custom_value1>,
   “<custom_key2>” : <custom_value2>,
    …
}
```

Details:

- Custom key-values (optional) - Sometimes, back ends require additional parameters for authentication such as SAP-SKY requiring a callerID to be passed. Caller IDs will be passed via additional name-value pairs from the identity service to the custom identity

Sample Success Response if MFA is enabled

The response contains `is_mfa_enabled` and `mfa_meta` fields if the back-end supports MFA. The user and security attributes are displayed after MFA validation. Following is the sample success response:

HTTP 200 OK

```
{
...
"is_mfa_enabled":true,
"mfa_meta":{"otp":2},
```

```
...
}
```

> *Important:* The "mfa_meta" field varies depending on the type of MFA set in the backend server. The login service uses it to indicate to client app the type of MFA to be used like OTP, Security questions or even to invoke the security questions. This meta is available for client app to help the client app logic determine the actual mechanism of getting the MFA delivered.

Sample Success Response if MFA not enabled:

If an end user does not require MFA or has not enabled it, then the `is_mfa_enabled` field is set to false and the user profile information and claims token are available in the response. There will be no further MFA validation in the current service call instance. Following is the sample success response:

HTTP 200 OK

```
{
"is_mfa_enabled":false,

“security_attributes” : {
“session_token”: “<session_token>”,
“session_ttl”: <>, /* -1 : if ttl is not available.
Else TTL in msecs */
“<optional key1>”: “<value1>”,  /* ex: refresh_token
“<optional key2>”: “<value2>”,
    },
   “user_attributes”:
{
"user_id” : “<userID_number>", /* federation ID of the user from
custom identity service
“<attr1>” : “<value1>”, /*ex: “first_name”: “<firstname>” */
“<attr2>” : “<value2>”, /*ex: “user_id”: “<user_id>” */
“<attr3>” : “<value3>”, /*ex: “role”: “<role>” */
```

```
    }
}
```

> ***Note:*** If "is_mfa_enabled" field is not there in the server response, then the backend server does not support MFA.

> ***Important:*** The user_id is a mandatory field in the log-in response.

Sample Success Response with httpStatusCode:

HTTP 200 OK

```
 {
“security_attributes” :

{ “session_token”: “<session_token>”, “session_ttl”: <>, /* -1 : if
ttl is not available. Else TTL in msecs */ “<optional key1>”:
“<value1>”, /* ex: refresh_token “<optional key2>”: “<value2>”, }
,
“user_attributes”:

{ "user_id” : “<userID_number>", /* federation ID of the user from
custom identity service “<attr1>” : “<value1>”, /*ex: “first_name”:
“<firstname>” */ “<attr2>” : “<value2>”, /*ex: “user_id”: “<user_id>”
*/ “<attr3>” : “<value3>”, /*ex: “role”: “<role>” */ }
,
"httpStatusCode" : 200 /* please note this is an integer. if this is
not 200 custom login will fail */
}
```

Possible Failure Responses

- HTTP 401 : Invalid/incorrect credentials
- HTTP 400 : Insufficient/wrong parameters
- HTTP 500 : Internal Server Error

Failure response when Custom identity protocol is configured as an integration service

```
{
    "details": {
        "message": "Middleware Service returned a HTTP Response Status [401], OpStatus [8009], Error Message [Request unsuccessful for service login, server responded with status code 401]",
        "errcode": 123,
        "errmsg": "backendErrorMessage"
    },
    "httpstatus": "Unauthorized",
    "requestid": "b5bb895b-7cc7-40db-8cfc-a2d69bfd31f9",
    "domain": "AUTH",
    "code": -14,
    "mfcode": "Auth-14",
    "message": "Configured IdP Service Unavailable"
}
```

To map the `backend_error_code` and `backend_error_message` with the `errcode` and `errmsg` of the Identity Service, the two values should be mapped to the integration service response output.

After mapping, the Integration Service response for the failed authentications is displayed as below:

```
 {
    "backend_error_message": "backendErrorMessage",
    "opstatus": 8009,
    "errmsg": "Request unsuccessful for service login, server responded with status code 401",
    "backend_error_code": "123",
```

```
    "httpStatusCode": 401
}
```

The Identity service extracts `backend_error_code` and `backend_error_message` from the integration response and maps them to the `errcode` and `errmsg` and sends the response.

```
 {
    "details": {
        "message": "Middleware Service returned a HTTP Response Status
[401], OpStatus [8009], Error Message [Request unsuccessful for
service login, server responded with status code 401]",
        "errcode": 123,
        "errmsg": "backendErrorMessage"
    },
```

Logout API for a Custom Identity Service

The Logout API clears the session of a logged-in user. The Logout API is optional for a custom identity service. You need to configure the logout API only if a logged-in user is associated with a session. You can configure the logout API in the Kony Fabric Console.

Method

POST

Headers

X-Kony-RequestId: <Request Id>

Content-Type: application/formurl-encoded only

Accept: application/json or application/xml

Sample Body Request:

```
{
  "session_token" : "<Session Token>"
}
```

Details:

session_token (optional)

Session Token of the user session in custom identity service

Sample Success Response:

- HTTP 200 OK, No Body

Sample Failure Response:

- HTTP 401 : In case of invalid credentials
- HTTP 500 : In case of internal server error

```
{
"domain" : "custom",
"code" : <error code>,
"message" : <Error Message>,
"details" : {
"message" : <Detailed Error Message>,
   },
 "requestid" : <Request Id>,
}
```

MFA API

If MFA is enabled, the client app should have the necessary logic to validate the MFA with any configured endpoint(s) such as an integration service to generate and validate OTP, or validate security questions entered in client app. Post the validation, the integration service should return an MFA key (mfa_key) to the client app, which should be passed using the Fabric SDK API to MFA validation endpoint along with known_user token. The endpoint should be configured to confirm the

authenticity of the mfa_key and accordingly send a successful response or failure. This 2 step process is to ensure that the backend generating MFA is not implicitly trusted by identity, but validates the mfa_key, before generating an authenticated session for the same. The MFA validation endpoint is invoked with the following parameters.

Sample Request Body

```
{
    “<mfa_key>” : <mfa key>,
    …
}
```

> ***Note:*** To know more about other parameters that are added by SDK, refer to the SDK document.

Sample Success Response

HTTP 200 OK

```
{
“security_attributes” : {},
“user_attributes”:
{
/* Fetched from Custom Identity Endpoint*/
"user_id” : “<userID_number>", /* federation ID of the user from
custom identity service
“<attr1>” : “<value1>”, /*ex: “first_name”: “<firstname>” */
“<attr2>” : “<value2>”, /*ex: “user_id”: “<user_id>” */
“<attr3>” : “<value3>”, /*ex: “role”: “<role>” */

/*Fetched from User Attributes Endpoint*/
“<attr4>” : “<value4>”, /*ex: “mobile_number”: “<mobile_number>” */
“<attr5>” : “<value5>”, /*ex: “department”: “<department>” */
“<attr6>” : “<value6>”, /*ex: “job_title”: “<job_title>”  */
```

```
}
}
```

User Attributes Endpoint API

The User Attributes Endpoint API retrieves additional user profile data which will passthrough integration service.

Method

GET

Headers

X-Kony-RequestId: <Request Id>

X-Kony-Authorization: <claims _token>

Content-Type: application/formurl-encoded only

Accept: application/json

Sample Success Response:

HTTP 200 OK

```
“user_attributes”:
{

/* Fetched from Custom Identity Endpoint*/
"content_type": "application/json",
"user_id” : “<userID_number>", /* federation ID of the user from
custom identity service
“<attr1>” : “<value1>”, /*ex: “first_name”: “<firstname>” */
“<attr2>” : “<value2>”, /*ex: “user_id”: “<user_id>” */
“<attr3>” : “<value3>”, /*ex: “role”: “<role>” */

/*Fetched from User Attributes Endpoint*/
“<attr4>” : “<value4>”, /*ex: “mobile_number”: “<mobile_number>” */
```

```
"<attr5>" : "<value5>", /*ex: "department": "<department>" */
"<attr6>" : "<value6>", /*ex: "job_title": "<job_title>"  */
…

}
```

Sample for Failure Response:

If invocation of optional User Attributes Endpoint fails and returns a successful response, then the response will have the following failure information:

HTTP 200 :

```
"user_attributes":
{

"content_type": "application/json",
"user_id" : "<userID_number>", /* federation ID of the user from
custom identity service
"<attr1>" : "<value1>", /*ex: "first_name": "<firstname>" */
"<attr2>" : "<value2>", /*ex: "user_id": "<user_id>" */
"<attr3>" : "<value3>", /*ex: "httpStatusCode": "<200>" */
…

"<httpStatusCode >" : "<200>", /*ex: "httpStatusCode": "<200>" */
"<opstatus>" : "<value4>", /*ex: "opstatus": "<integer value>" */
"<errmsg>" : "<value5>", /*ex: "errmsg": "<error details>" */

}
```

Security Attributes Endpoint API

The Security Attributes Endpoint API retrieves additional security attributes data which will passthrough integration service. These are server only attributes and are accessible from the custom code in server (pre/post processors).

Method

GET

Headers

X-Kony-RequestId: <Request Id>

X-Kony-Authorization: <claims _token>

Content-Type: application/formurl-encoded only

Accept: application/json

Sample Success Response:

HTTP 200 OK

```
“user_attributes”:
{
"content_type": "application/json",
"user_id” : “<userID_number>", /* federation ID of the user from
custom identity service
“<attr1>” : “<value1>”, /*ex: “session_token”: “<session_token>” */
“<attr2>” : “<value2>”, /*ex: “refresh_token”: “<refresh_token>” */
“<attr3>” : “<value3>”, /*ex: “_provider_token”: “<_provider_token>”
 */
…

“<attr4>” : “<value4>”, /*ex: “session_auto_extend”: “<session_auto_
extend
>” */
“<attr5>” : “<value5>”, /*ex: “session_idle_timeout”: “<session_idle_
timeout>” */
…
}
```

Sample Failure Response

If invocation of optional Security Attributes Endpoint fails and returns a successful response, then the response will have the following failure information:

HTTP 200:

```
“user_attributes”:
{
“<attr1>” : “<value1>”, /*ex: “session_token”: “<session_token>” */
“<attr2>” : “<value2>”, /*ex: “refresh_token”: “<refresh_token>” */
“<attr3>” : “<value3>”, /*ex: “provider_token”:“<provider_token>”*/
…

“<opstatus>” : “<value4>”, /*ex: “opstatus”: “<integer value>” */
“<httpStatusCode >” : “<httpStatusCode >”, /*ex: “httpStatusCode”:
“<200>” */
“<errmsg>” : “<value5>”, /*ex: “errmsg”: “<error details>” */

}
```

19.4.12.3 Custom Identity MFA Flow

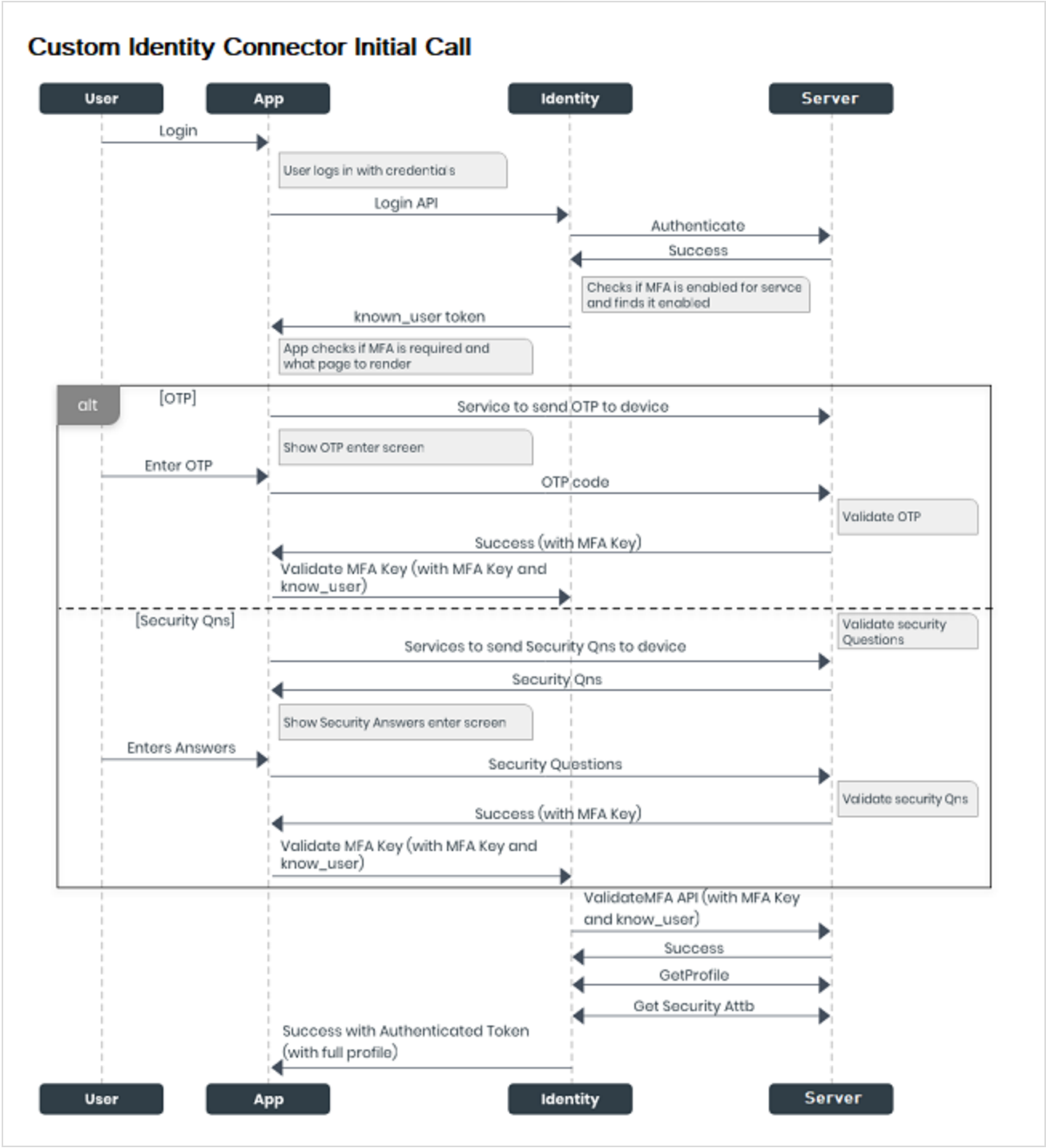
Initial Call Process

When a user signs in to the application for the first time, the following calls occur among the app, Identity service, and server:

- The login request is sent to the Kony Fabric Identity service through the Login API.
- The Identity service sends the login request to the server to be authenticated.
- On successful authentication of the user from the server, the Identity service verifies the MFA status. By default, MFA is enabled. If MFA is not enabled, the Identity service returns the authenticated token.
- If MFA is enabled, the Identity service sends a “**known_user token**” to the application.
- The app contains the custom logic internal to its implementation to fetch MFA. The configured

MFA can be of any type, namely: OTP, and/or security questions, and/or any custom logic to obtain an MFA key.

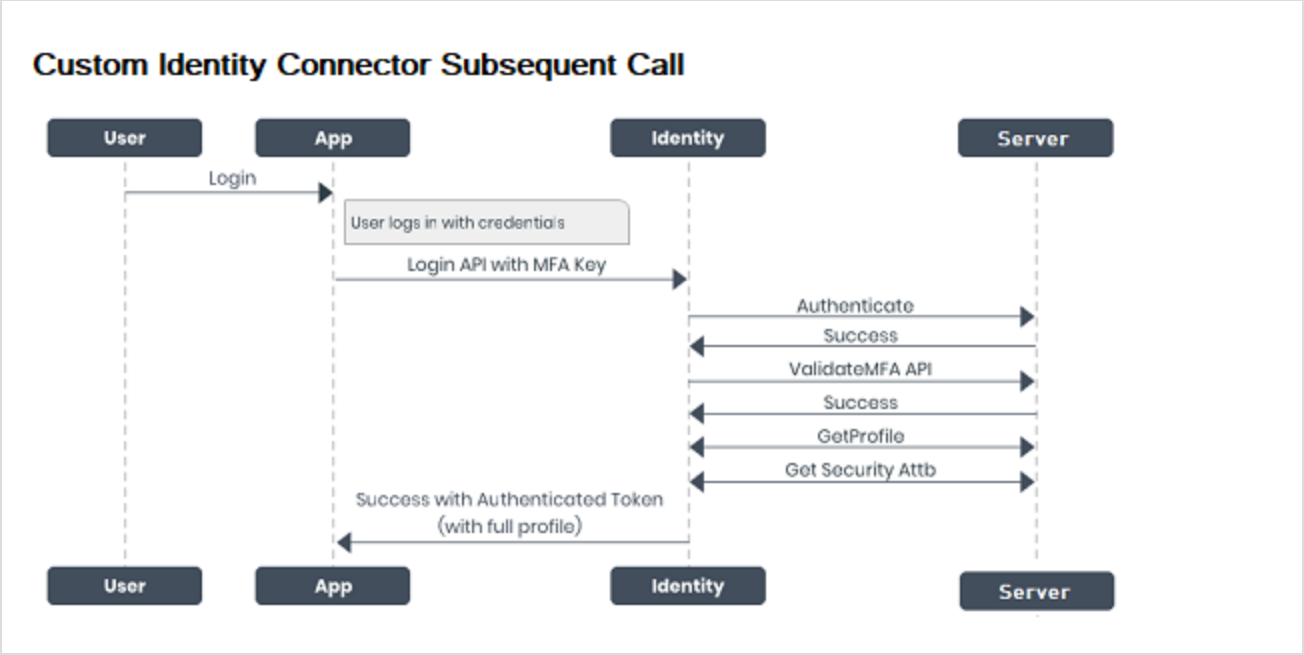
- **OTP flow**
    - If the bank or user enables OTP as the type of MFA, the app displays the OTP screen. And the app invokes a service that sends an OTP to the user's registered device.
    - After the user type the OTP, it is sent to the server to be validated.
    - On successful validation, the server sends an MFA key to the application.
    - The app sends the MFA key along with the Known User token to the Identity service.
- **Security Questions Flow**
    - If the bank or user enables Security Questions as the type of MFA, the app displays the Security Questions screen. And the app invokes a service that requests the server to send the security questions.
    - After the user provides the answers to the questions, they are sent to the server to be validated.
    - On successful validation, the server sends an MFA key to the application.
    - The app sends the MFA key along with the Known User token to the Identity service.
- The Identity service sends the MFA key and the Known User token to the server to validate the user who is trying to access the application.
- On successful validation, the server sends the profile and security attributes of the user to the Identity service.
- The Identity service sends the authenticated token with full profile details to the application.



Subsequent Call Process

On subsequent logins, the following calls occur among the app, Identity service, and server:

- The application invokes the Login API along with the stored MFA key, and sends the details to the Identity service.
- The Identity service sends the MFA key and the known_user token to the server to validate the user who is trying to access the application.
- On successful validation, the server sends the profile and security attributes of the user to the Identity service.
- The Identity service sends the authenticated token with full profile details to the application.



19.4.12.4 How to Configure a Custom Identity Service by using an Integration service

A custom identity service is configured by pointing to an endpoint, which follows a custom identity protocol/contract.

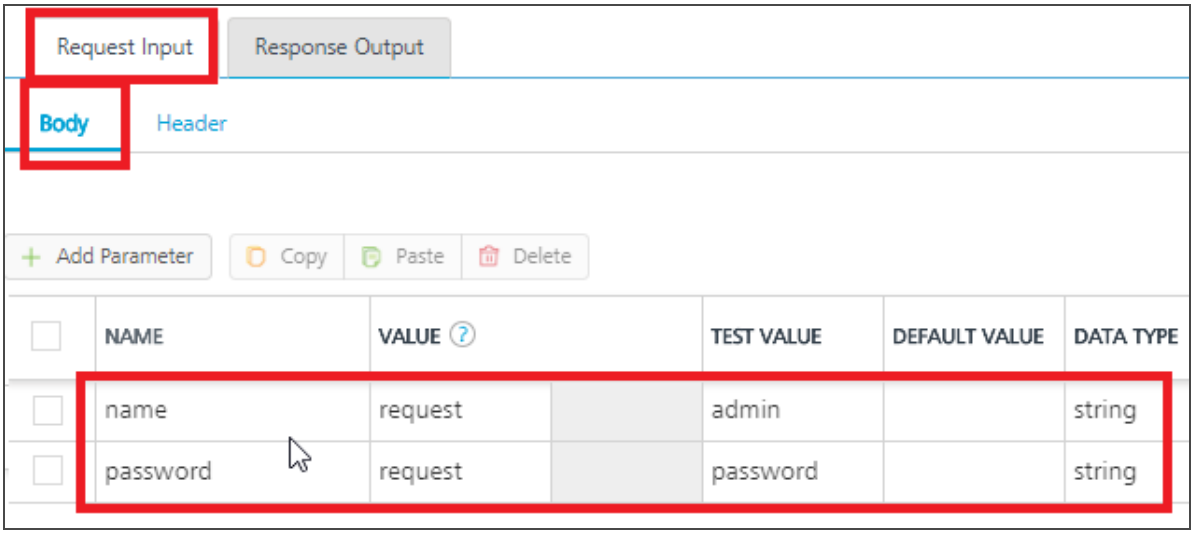
An external back-end server can be used to configure a custom identity service, if it implements the identity protocol/contract. Otherwise you can make use of Kony Fabric Integration Service, which internally points to the authentication/login API of the back-end server.

For example,

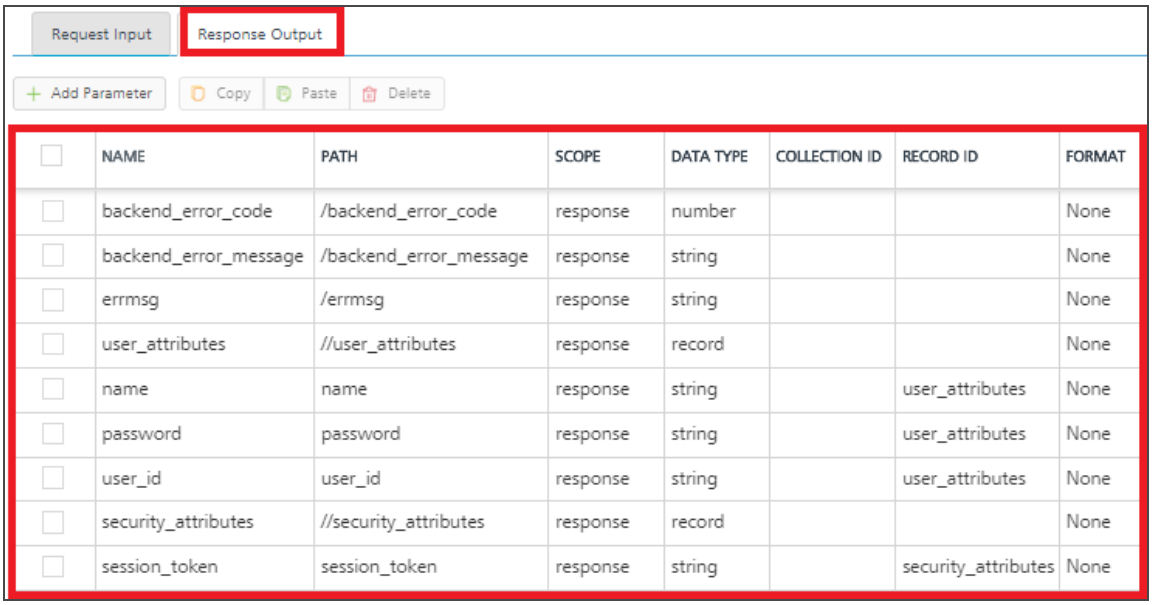
- If a back-end server **does not support** login and logout APIs, you can configure a custom auth service by pointing to an end-point.
  For example, A Kony Fabric user creates an integration service that has a login and logout operations. Based on the input and output parameters configuration, Kony Fabric fetches a response for a user for login and logout operations. To get custom back-end server's error codes and error messages in the login/logout response, a developer must map error codes and messages with the parameters `backend_error_code` and `backend_error_message` in the Integration service **Response Output**. The following steps explain how to configure a custom auth service by pointing to an end-point.

To create a custom auth service by pointing to an end-point that, follow these steps:

1. Create a **JSON** integration **service** with the custom back-end server's base URL.
2. Create an operation by name **`login`**.
   a. In the **Target URL** field of the operation, enter the **`login/authentication`** API of the custom back-end server.
   b. In the **Operation Security Level** list, select one of the **Authenticated App User** option. The **Authenticated App User** option restricts the access to clients who have successfully authenticated using an Identity Service associated with the app.
   c. In **Request Input**, configure the login parameters for **Body**, **Header**, and **Request Template** (for example, name and password.)



d. In the **Response Output**, configure the following parameters.



3. Similarly create an operation by name `logout` as in the previous step.
4. Publish the app.
5. Copy the `service URL` from the App Service Document in the **Publish > Environment** page.

```
"services_meta": {
 "custom": {
  "type": "integsvc",
  "version": "1.0",
  "url": "https://test11.testtest.net:11111/services/custom"
 }
}
```

6. Now, create a custom identity service.
7. In the **Custom Identity Service Endpoint**, paste the `service URL` that you copied from the **App Service Document**.

   You can now test the login from your authentication server of the custom back-end.

19.4.13 Kony Fabric OAuth 2.0 Identity Service

Kony Fabric identity supports OAuth 2.0 protocol for authenticating back-end identity providers that support `RFC6749`. For more details, refer to https://tools.ietf.org/html/rfc6749

With **Kony Fabric OAuth 2.0** identity service, a user can access some of the external OAuth service providers such as Salesforce, Google, Amazon, Microsoft, Instagram, Yahoo, and Box for authentication.

Based on the user's request made by using SDKs, the Kony Fabric OAuth 2.0 identity service selects the Web server authorization grant flow (3-legged OAuth) for performing user authentication.

- **Example for 3-legged OAuth**: When a user logs into a Kony Fabric application by using the Google OAuth endpoint, the authorization service (for example, Google OAuth endpoint) redirects the user to Google login page. During this stage, the callback URL is also set. The user enters login details such as userID and password. After successful authentication, the user is logged into Google account. Based on the Kony Fabric OAuth 2.0 identity configuration, identity service filters the user profile data from Google and stores the details in the identity session.

User Profiles in Kony Fabric OAuth 2.0

Different service providers implement user profiles as per their own standards. Kony Fabric OAuth 2.0 identity service retrieves user attributes from a user profile and saves these attributes in Kony Fabric Identity Sessions after successful login response.

> ***Note:*** A profile endpoint provides the profile of the logged in user.

Kony Fabric OAuth 2.0 helps users to configure authorization provider to access User Profile data in one of the following ways:

- User profile data is provided as part of the response of GET on a fixed URL.
- User profile data is provided as part of the response of GET on a URL, but the URL itself is not fixed. However, the URL is available either in the:
    - Response of another call
      or
    - In the token response
- User profile data is available as part of the token response.

Advantages of Kony Fabric OAuth 2.0 identity service

- You can use the OAuth 2.0 identity service to retrieve and save user attributes in Kony Fabric identity sessions after a successful login response, and then use the attributes as client filters during Offline Sync calls.

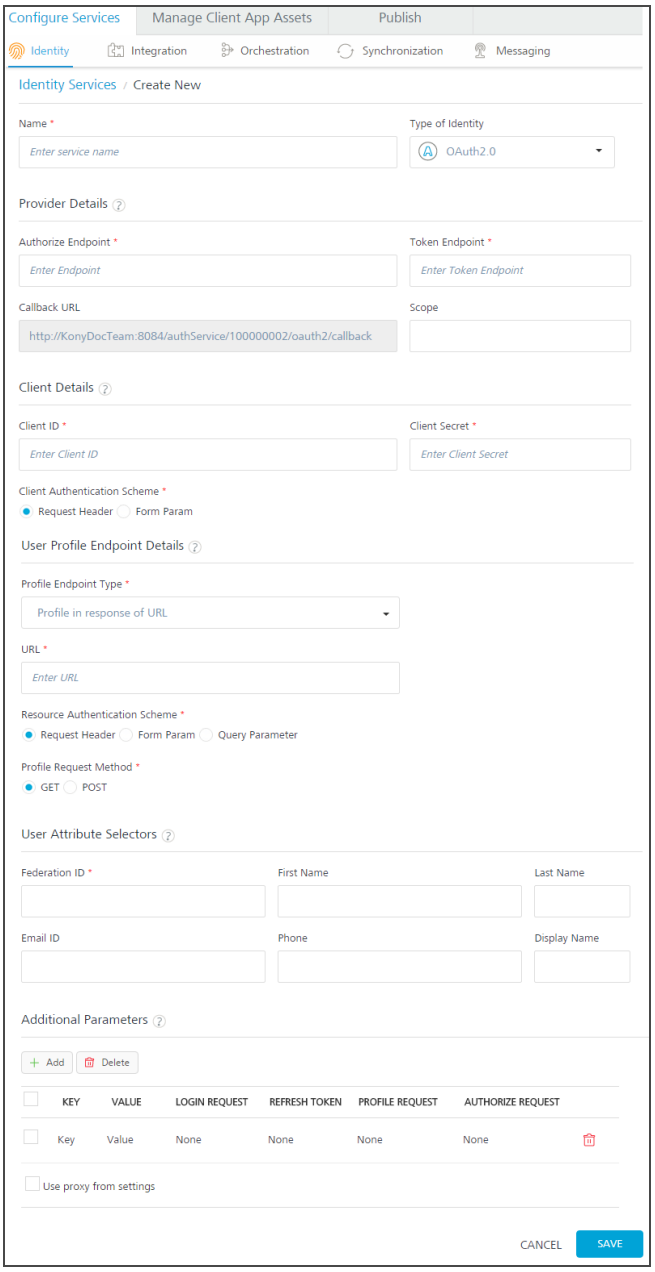
  > ***Note:*** If you enable synchronization capability in your app, Kony Fabric OAuth 2.0 identity service uses the user attributes retrieved from the user profile as client filters during Offline Sync calls.
  >
  > For example, the logged in user's role (such as Manager or Employee of an organization) received as part of a User Profile after a successful OAuth login can be used as a client side filter for Offline Sync. For more details, see the Synchronization > client side filters topic.

> ***Note:*** To configure Okta identity service, you can the same steps provided for the Auth 2.0 Identity Service section as well.

19.4.13.1 How to Configure a Kony Fabric OAuth 2.0 Identity Service

To configure an identity service using Kony Fabric OAuth authentication mode, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Name** text box.



2. From the **Type of Identity** list, select **OAuth2.0**.

> ***Note:*** Fields marked with an asterisk are required.

3. In the **Provider Details** text box, configure the following endpoints:
    a. **Grant Type**: A new authorization grant type is defined by providing a grant type parameter in Kony Fabric Console. Select the required grant type from the drop-down menu. The following are the different grant types you can select:
        i. **Authorization code**: An authorization code grant type is used if the client wants to request access to protected resources on behalf of another third-party user. When the client requests for the authorization, the authorization server redirects to the third party URL. The client enters the authorization code and gains access to the protected resources.
        ii. **Password**: On selecting this grant type, the client needs to provide the **Token Endpoint**. The client provides their access credentials to access the URL provided in the Token Endpoint. These credentials are validated by the back-end server. If the credentials are valid, it redirects back to the Console.
        iii. **JWT Bearer**: When the JWT bearer grant type is selected, the provider picks the existing identity provider and pass the values as input to JWT bearer provider for the response. Apart from client specified parameters, the system picks the existing MF auth token and gets the relevant values from the session. The values are passed as input to the current JWT bearer provider login. The existing identity session is selected from the **Token Identity provider** drop-down menu.
        iv. **Extension**: When the extension grant type is selected, the system displays the **Extension Grant** text box to enter the grant type value. For extension grant type, the oAuth does not have any fixed value. The grant type value is dynamic, which is a client specified parameter.
        v. **Client Credentials**: When the client credentials grant type is selected, the client can request an access token using only its client credentials when the client is requesting access to the protected resources under its control, or to those of another resource owner that has been previously set up with the authorization server.

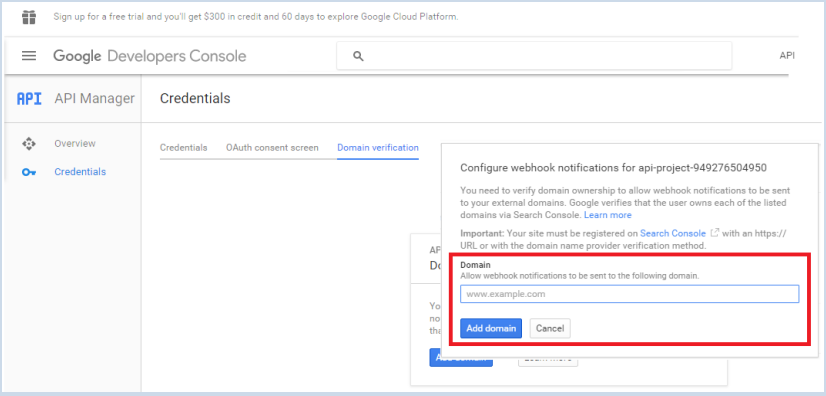
b. **Authorize Endpoint**: Enter the URL that is provided by the endpoint service provider. For example, Salesforce or Google.
Sample authorize endpoint for Google:
`https://accounts.google.com/o/oauth2/auth`

c. **Token Endpoint**: Enter the token URL that is provided by the endpoint service provider.
Sample token endpoint for Google:
`https://accounts.google.com/o/oauth2/token`

d. In the **Callback URL** text box, the system displays the default URL generated by an identity service. You need to configure the callback URL details on the authorization server against your app. You cannot modify these details.

Sample callback URL:
`https://<accountID>.auth.konycloud.com/OAuth2/Callback`

> ***Note:*** Provide domain as a service URL.
>
> For example, configure your Google app to accept requests from authentication service by typing the `auth.konycloud.com` in the **App Domain** text field.



e. In the **Scope** text box, enter the list of permissions that a user needs to agree to while in the user log-in page of the OAuth 2.0 service provider - for example, Gmail. The list can

contain more than one permission and depends on the authorization server. For a sample of full list of permissions, refer to OAuth 2.0 standards.

For example, if you configure the `email` permission as scope, after you log in Google for the first time, Google displays the **Log in with Google** dialog with the configured permissions. Click **OK** to share your permissions (public profile and email) with Kony Fabric.

4. Under **Client Details**, configure the following parameters. The client details are used by a service provider to identify which app the authorization service is trying to access:
    a. In the **Client ID** text box, enter the client ID (for Google, app key) of the app instance that you created in service provider's developer console. For example, Google Developers Console.
    b. In the **Client Secret** text box, enter the client secret (for Google, app secret key) of the app instance that you created in service provider's developer console. For example, Google Developers Console.
    c. In the **Client Authentication Scheme**, select one of the following options to the client details as headers or form parameters:
        - **Request Header**: To send client details as a header.
        - **Form Param**: To send client details as form parameters.
5. Under **User Profile Endpoint Details**, configure the following parameters to get the user profile from a service provider - for example, Google.
Different service providers implement user profiles as per their own standards. Kony Fabric OAuth 2.0 identity service retrieves user attributes from a user profile and saves these attributes in Kony Fabric identity sessions after a successful login response.

> ***Note:*** If you enable synchronization capability in your app, Kony Fabric OAuth 2.0 identity service uses the user attributes from the user profile as client filters during Offline Sync calls. For example, User Role (one of the attributes of the user profile) received as part of User Profile after a successful OAuth 2.0 login can be used as client side filter for Offline Sync. For more details, refer to Synchronization > client side filters.

a. Kony Fabric OAuth 2.0 supports four types of user profile endpoint configurations. Select one of the following profile endpoint types from the **Profile Endpoint Type** drop-down list:
   - If you want to skip profile configuration for the provider, you can select **None**.
   - Profile in response of URL
   - Profile endpoint in response of URL
   - Profile in Token response
   - Profile endpoint in Token response.

b. Configure the following details based on the selected profile endpoint type:
   - To configure the **Profile in response of URL** as profile endpoint type, follow these steps:
     i. Enter the **URL** of the endpoint.
     ii. Under **Resource Authentication Scheme**, select one of the following authentication schemes:
        - **Request Header**: To send access token to profile endpoint as a header.
        - **Form Param**: To send access token to profile endpoint as form parameters.
        - **Query Parameter**: To send access token to profile endpoint as query.
     iii. Under **Profile Request Method**, click the **GET** or **POST** option.

- To configure the **Profile endpoint in response of URL** as profile endpoint type, follow these steps:
    i. Enter the **URL** of the endpoint.
    ii. In the **Profile endpoint selector**, enter the profile endpoint selector.
    iii. Under **Resource Authentication Scheme**, select one of the following authentication schemes:
        - **Request Header**: To send access token to profile endpoint as a header.
        - **Form Param**: To send access token to profile endpoint as form parameters.
        - **Query Parameter**: To send access token to profile endpoint as query.
    iv. Under **Profile Request Method**, click the **GET** or **POST** option.
- If you select the **Profile in Token response** as profile endpoint type, follow the steps from User Attributes Selectors section.
- To configure the **Profile endpoint in Token response** as profile endpoint type, follow these steps:
    i. In the **Profile endpoint selector**, enter the profile endpoint selector.
    ii. Under **Resource Authentication Scheme**, select one of the following authentication schemes:
        - **Request Header**: To send access token to profile endpoint as a header.
        - **Form Param**: To send access token to profile endpoint as form parameters.
        - **Query Parameter**: To send access token to profile endpoint as query.

iii. Under **Profile Request Method**, click the **GET** or **POST** option.

6. Under **User Attribute Selectors**, configure the following fields to extract the required information from the response:
   a. **Federation ID**: Enter the federation ID of the user. The federation ID is a mandatory field and it is unique ID of the user at the service provider.
   b. **First Name**: Enter the JSON path to extract the first name from the user profile information.
   c. **Last Name**: Enter the JSON path to extract the last name from the user profile information.
   d. **Email ID**: Enter the JSON path to extract the email ID from the user profile information.
   e. **Phone**: Enter the JSON path to extract the phone number from the user profile information.
   f. **Display Name**: Enter the JSON path to extract the display name from the user profile information.
   g. **Custom User Attribute Selectors**: Enter custom attributes to add more attributes in addition to the ones defined above. For example, `groups=user.groups` to define a custom attribute with name `groups` and maps to `user.groups` JSON path in backend profile response.

> ***Note:*** The parameters in **User Attribute Selectors** are supported in JSON path format.

```
// Sample profile response in JSON format

{
      "id": "0001", // federation ID of the user
      "name":
```

```
                {
                        "first":"John", // first name of the user
                        "last":"Doe", // last name of the user
                        "display":"john.doe" // display name of the user
                },
        "email": // email ID of the user
                [
                        "john.doe@gmail.com",
                        "john.doe@hotmail.com"
                ],
        "phone": "1234567890", // phone number of the user
        "picture": "picture", //"https://my-
company.com/profile/picture/1002"
}
```

```
 // Sample profile attributes selectors for the above response

"federation id": "id"
"first name":"name.first"
"last name": "name.last"
"display name":"name.display"
"phone":"phone"
"email":"email[0]"
```

```
 // Sample custom attributes selectors for the above response

"federation id": "id"
"first name":"name.first"
"last name": "name.last"
"display name":"name.display"
"phone":"phone"
"email":"email[0]"
"picture": "https://my-company.com/profile/picture/1002"
```

7. **Redirect URL on successful authentication**:
    - **Any URL**: Select this to use any URL on successful authentication.
    - **Allowed URL list**: Select this to use a specific set of URLs on successful authentication.
        - **URL**: In the URL text box, enter the allowed URLs.
8. Under **Additional Parameters**, configure the following additional parameters. Some service providers require additional parameters based on their standards. These additional parameters' name-value pairs will be sent to a service provider along with `authorization request`, `token request`, or `profile request` as header, body, or query parameters.

    > ***Note:*** You can add an entry by clicking the **Add** button if entries for the input and the output tabs do not exist.
    > To delete an entry, click the **Delete** button at the end of that entry.
    > To delete a group of entries, select the check boxes for the entries, and then click the **Delete** button under the **Additional Parameters** section.

    a. Click **Add** to add an entry (row) and do the following:

    b. Under the **KEY** field, click and enter the parameter name.

    c. Under the **Source** field, click and select one of the following options from the drop-down list. By default, this field is set to **Constant**.

        i. **Constant**: This mode allows you to configure the value to be sent as a part of the selected request.

        The value set as a part of the configuration is sent in the selected request.

        ii. **Identity Configuration**: In this mode, Kony Fabric picks up the value of the custom header from the identity configuration. The supported values are Client ID, Client Secret, Scope, and Grant Type.

iii. **Client Specified**: In **Client Specified** mode, Kony Fabric picks up the value of the `value` from the client request.

The value field in the configuration will be disabled. When the client tries to login, the request must contain the parameter in the request body.

iv. **Identity Session**: In this mode, Kony Fabric picks up the value of the custom header from the identity session. The identity session contains the details of the identity providers logged into by the application. Identity session contains two different types of attributes for each provider: **Profile** and **Security**.

For this mode to work, Kony Fabric should at least have two identity providers associated with it. The value in the provider configuration will be of the following pattern:`<Provider Name>.<profile|security>.<Attribute Name>`.

Based on the above configuration, the value is picked up from the identity session.

d. Under the **VALUE** field, click and enter the parameter value.

e. Under the **LOGIN REQUEST** field, click and select one of the following options from the drop-down list. By default, this field is set to **None**.

- Header
- QueryParam
- FormParam

f. Under the **REFRESH TOKEN** field, click and select one of the following options from the drop-down list. By default, this field is set to **None**.

- Header
- QueryParam
- FormParam

g. Under the **PROFILE REQUEST** field, click and select one of the following options from the drop-down list. By default, this field is set to **None**.
   - Header
   - QueryParam
   - FormParam

h. Under the **AUTHORIZE REQUEST** field, click and select the following options from the drop-down list. By default, this field is set to **None**.
   - QueryParam

9. After entering the above details, click on the **TEST LOGIN** button to verify the credentials.

   If you have not logged in to your the social identity service (for example: Gmail), the Console redirects you to the back-end identity provider's log-in page. Enter your credentials as required.

   The test results are displayed in the **Identity Response** dialog.

10. Click the **Advanced** to provide additional configuration of your service definition:
    - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.
    - **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.

11. After entering the above details, click **SAVE** to save the service. The system displays the **Identity** page. The Kony Fabric OAuth 2.0 identity service is configured.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.13.2 How to Test a Kony Fabric OAuth 2.0 Identity Service

If you configure an OAuth 2.0 identity service in Kony Fabric for a mobile app, the mobile app needs to use OAuth to authorize and authenticate users before they access any APIs. You can test the login for the OAuth identity provider in the OAuth Identity service definition tab. As a response to the login, you can view the identity response and the profile and token response that you get from the backend. Also, you can use the information from the backend profile response to help you configure the user profile under User Attribute Selectors.

The primary benefit of testing the login is that you will know if there is a problem with the URL or other configuration errors during the configuration of the OAuth 2.0 connector. You can use the details captured from the response to troubleshoot and resolve any problems with the OAuth 2.0 connector.

To test a Kony Fabric OAuth 2.0 identity service, do the following:

1. In the Identity service configuration page, click **Test Login**.

   A sign in screen for the OAuth 2.0 service appears. For example, the sign in screen for your Google account.

2. Enter your credentials and click **Sign in**.

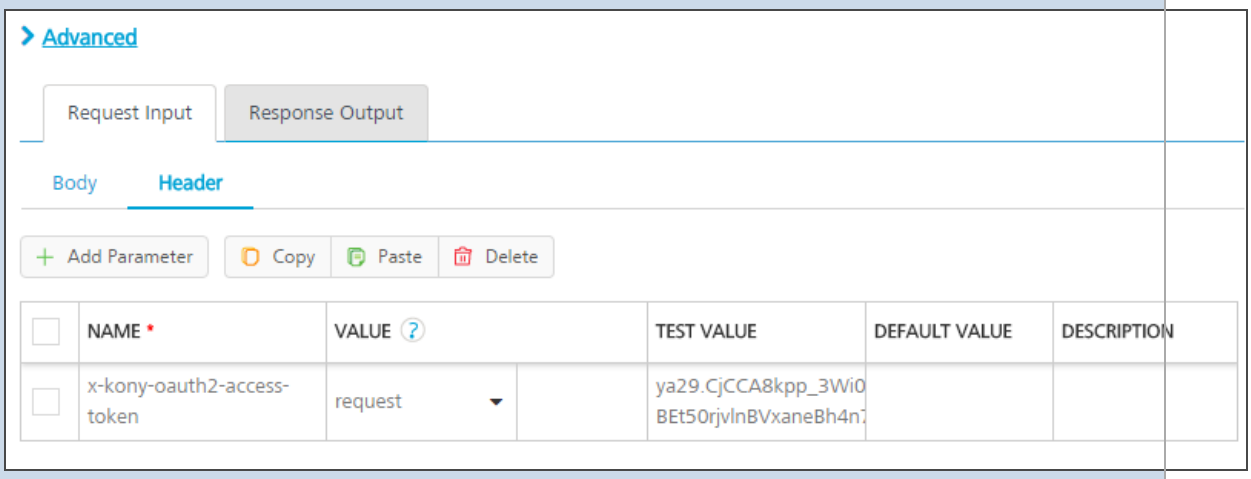
   An alert indicates a successful login and a pane appears that has tabs for Backend Token Response, Backend Profile Response, Identity Response. If your sign in failed, the error message that appears provides information about the failure.

> ***Important:*** If a custom integration service (for example, MongoDB or RAML) is linked to an OAuth2 identity service, while testing an operation of the integration service from Kony Fabric Console, you must pass the `x-kony-oauth2-access-token` as a header and

`access_token` as a header value.

Also, If a custom integration service (for example, MongoDB or RAML) is linked to an OAuth2 identity service, while testing an operation of the integration service from Admin Console, you must pass the `x-kony-oauth2-access-token` as a header and `access_token` as a header value.

For example:



Note: For more information on how you can integrate Kony OAuth Provider, User Repository, and OAuth 2.0 Identity services to create a basic login form, refer to a Base Camp article: Exploring Kony OAuth Provider.

19.4.14 OAuth Provider Identity Service

With **OAuth Provider** identity service, a user can configure an OAuth provider for authentication.

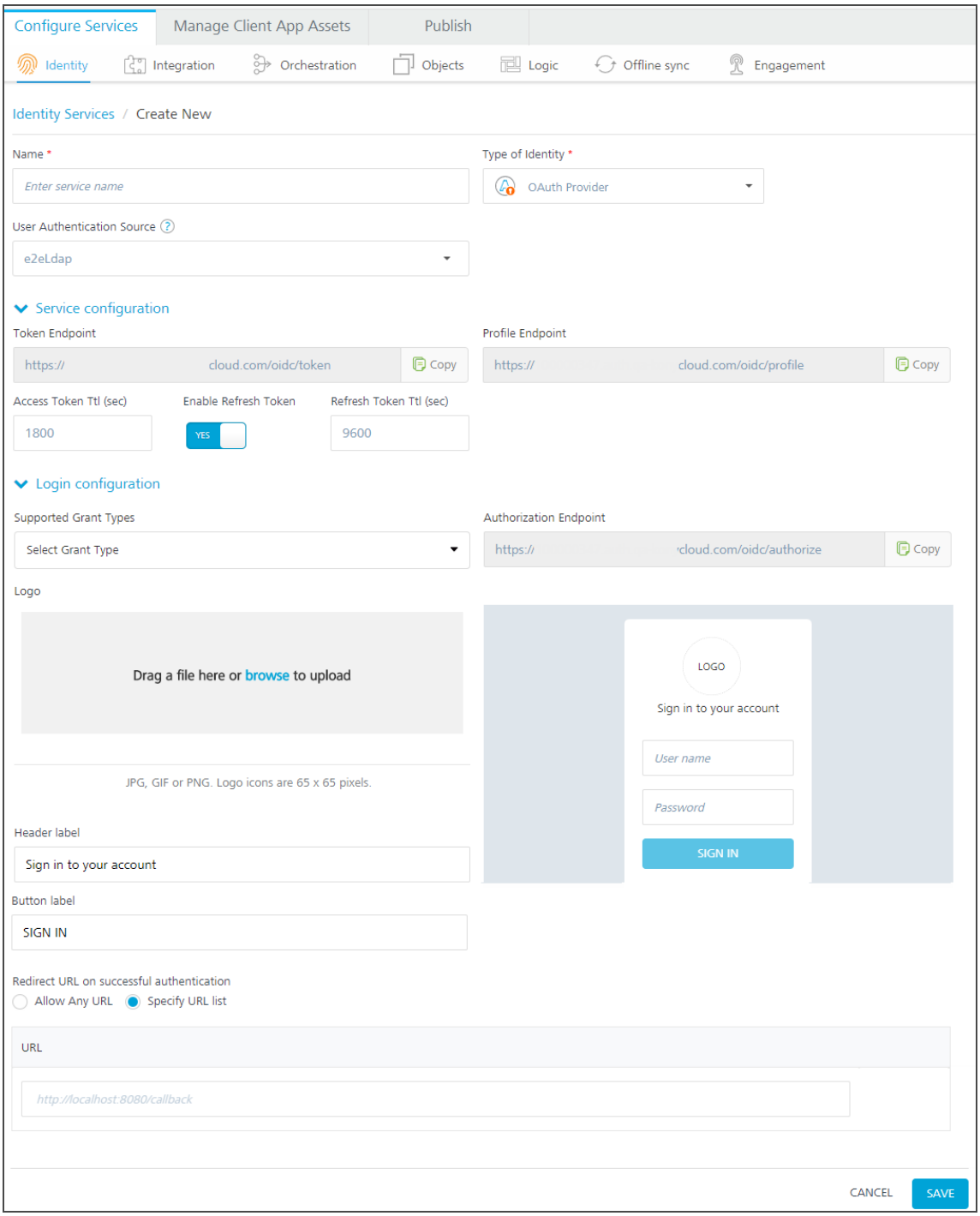
Service Configuration in Kony Fabric OAuth Provider

Different service providers implement service configuration as per their standards. Kony Fabric OAuth Provider identity service retrieves service configuration from the selected User Authentication Source and saves these attributes in Kony Fabric Identity Sessions.

19.4.14.1 How to Configure a Kony Fabric OAuth Provider Identity Service

To configure an identity service using Kony Fabric OAuth Provider authentication mode, follow these steps:

1. Under the **Identity service designer** page, type a name for the service in the **Name** text box.



2. From the **Type of Identity** list, select **OAuth Provider**.

   > ***Note:*** Fields marked with an asterisk are required.

3. From the **User Authentication Source** list, select an identity service that to be used as the authentication source provider for this OAuth provider.

   > ***Important:*** Currently the **Custom**, **User Repository** and **Microsoft AD LDAP** identity services are supported as **User Authentication Sources**.

   The Service Configuration details appear. The details include Token Endpoint, Profile Endpoint, Access Token Ttl (sec), and Enable Refresh Token.

4. Click the **Service Configuration** and configure the required fields:
   a. The **Access Point Token Ttl (sec)** sets the session time of the token. By default, the value is set to 1800 seconds. Modify the access token ttl value if required.
   b. From the **Enable Refresh Token** toggle button, select **Yes** if you want to enable refresh token.

5. Click the **Login configuration** and configure the required fields:
   a. From the **Supported Grant Type** list, select the required grant type from the drop-down menu. The following are the different grant types you can select:
      i. **Authorization Code**: An authorization code grant type is used if the client wants to request access to protected resources on behalf of another third-party user. When the client requests for the authorization, the authorization server redirects to the third party URL. The client enters the authorization code and gains access to the protected resources. In the Authorization code, you can configure the following:
         a. **Logo**: You can configure the logo that you want to appear on the authorization login page. You can drag a file or browse to upload a file. You can add a JPG, GIF, or PNG. The logo icon size should be 65 x 65 pixels.

b. **Header Label**: In the **Header Label** text box, enter the text you want to display below the logo on the login page.

c. **Button Label**: In the **Button Label** text box, enter the text you want to display for authorization. For example, Sign In.

> ***Important:*** For a custom identity service, you must configure the additional **Sign in Parameters** details along with the Logo, Header label, and Button label.

d. Follow the below step only for a custom entity service. If you have selected the Custom identity service in the **User Authentication Source** list, the **Sign in Parameters** section appears. Configure the sign in parameters for the login screen.

Input Type	Service Input parameter	Display Name
Select the type of input field as **Text** or **Password**.	The **Service Input parameter** is assigned to the attribute `<<name>>` of the input element. Enter the **Service Input Parameter** as per your server requirement as per your server.	The **Display Name** is assigned to the attribute `<<placeholder>>` of the input element. Enter the **Display Name** that you want to display in Login page of a client app.

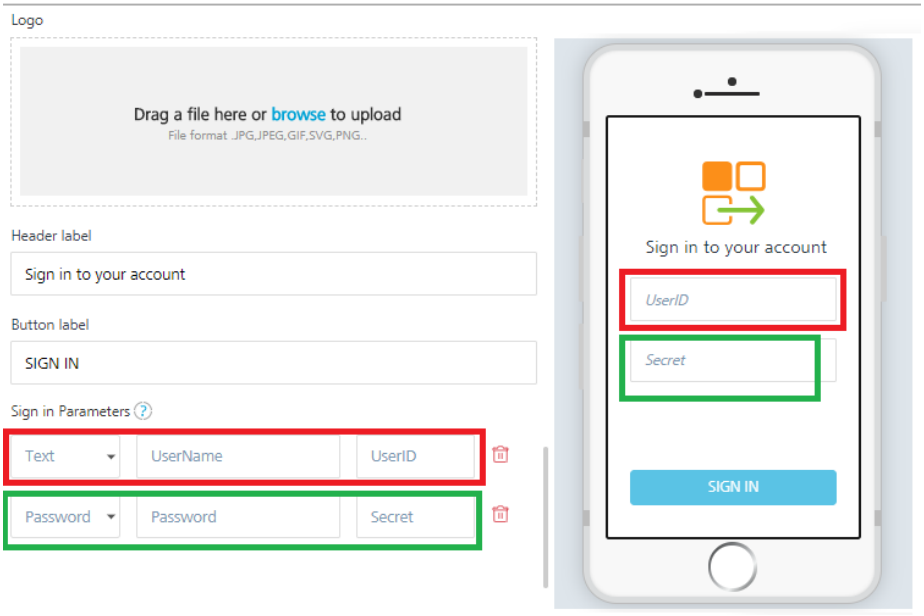
Input Type	Service Input parameter	Display Name

* For example, the following is a sample code for the **Text** input type.

```
<input class="form-control" type="text"
name="UserName" placeholder="userID">
```

For example, the following is a sample code for the **Password input type.

```
<input class="form-control" type="password"
name="password" placeholder="secret">
```



e. **Redirect URL on successful authentication**:

- **Any URL**: Select this to use any URL on successful authentication.
- **Allowed URL list**: Select this to use a specific set of URLs on successful authentication.
    - **URL**: In the URL text box, enter the allowed URLs.

ii. **Resource Owner Password**: On selecting this grant type, the client needs to provide the **Token Endpoint**. The client provides their access credentials to access the URL provided in the Token Endpoint. The back-end server validates these credentials. If the credentials are valid, it redirects back to the Console.

iii. **Client Credentials**: When the client credentials grant type is selected, the client can request an access token using only its client credentials when the client is requesting access to the protected resources under its control, or to those of another resource owner that has been previously set up with the authorization server.

6. Click the **Advanced** to provide additional configuration of your service definition:
    - Now you can enable or disable the integrity check for an identity service at the provider level. If the integrity is disabled at the provider level, then the provider is meant for server-to-server communication only. To disable the integrity check, In **Advanced**, select the **Restrict to Fabric Server to Server Authentication** check box. This setting blocks a traditional client app from using an identity service. It will only allow the identity service to be used from a Kony Fabric Server to authenticate and invoke services.
    - **Concurrent User Logins**: Select one of the following three options to configure concurrent user login sessions. For more information, refer to Concurrent User Logins.
        - **Allow concurrent user sessions (no restrictions)**: When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
        - **Allow only one active user session per app**: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app

user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked.

- **Allow only one active user session across all apps**: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service.

  > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

7. After entering the above details, click **SAVE** to save the service. The system displays the **Identity** page. The Kony Fabric OAuth Provider identity service is configured.
8. Navigate to the **Apps** pane from the left pane.
9. Create an app and link the above created OAuth Provider under the Identity tab.
10. Navigate to the **Publish** tab and publish the app. App key and secret which are displayed after publish will serve as the client id and secret for the provider.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

19.4.14.2 How to Use a Kony Fabric OAuth Provider

Once the OAuth provider is configured, and the application is published, the OAuth provider is ready for use. To use the OAuth Provider in the authorization code flow, follow these steps:

1. From your OAuth provider configuration page, copy the authorization endpoint and the token endpoint.
2. From the app publish page, copy the app key and the app secret.
   The app key and the app secret serve as the client ID and the client secret or OAuth 2.0.

3. Invoke the authorization request API in an internet browser with appropriate parameters.

   The Syntax for Authorization API is

   ```
   <Authorization endpoint>?client_id=<appkey>&scope=<space separated values of one or more of valid scopes >&response_type=code&redirect_uri=<valid redirect uri>&prompt=<value>
   ```

Parameter	Required	Description
client_id	Yes	The App Key string that you obtain from the Publish page.
response_type	Yes	If the value of response_type is **code**, the API launches a Basic flow. You should pass a POST query/request to the token endpoint to acquire the tokens.
scope	Yes	The scope parameter supports the following values: • openid • profile • email

Parameter	Required	Description
redirect_uri	Yes	It determines the location of the response that is sent. redirect_uri should be the HTTP endpoint on your server that receives the response from Kony OAuth Provider. In the **Redirect URL on successful authentication** field, you can either select **Allow any URL** or **Specify URL list**. In both cases, the Authorization API sends the authorization code to the redirect URI provided in the API. But for the Specify URL list option, the redirect_uri value must match one of the authorized redirect URIs as specified in the OAuth Provider. If this value does not match an authorized URI, a **redirect_uri_mismatch** error is thrown.
prompt	Optional	A space-delimited list of string values that specifies whether the authorization server prompts the user for re-authentication and consent. The prompt parameter supports the following values: • **consent**: The authorization server prompts you for consent before returning information to the client. • **login**: If you have already signed in, and passed login as an input to prompt, the authorization server prompts you to sign in again. ***Note:*** If you invoke the authorization API without passing the parameter prompt, the API takes **consent** as the value by default.

4. Enter login credentials and then click submit.
   A consent screen displays details of resources granted access.

5. Grant access as required. Once authorized, an authorization code is generated and the screen is redirected to a new URL with the code as the query parameter.
   For example, http://myredirect?code=XYZ

6. Make a POST call to the token endpoint with the code obtained in the previous step.
   For example, https://100000058.auth.konycloud.com/oidc/token/myprovider

```
 Request Method: POST


Request Body:
code=<CODE>&grant_type=code&scope=profile&client_
id=<appkey>&client_secret=<appsecret>&redirect_uri=<valid
redirect_uri>


Sample Response:

200 OK

{
    "access_token":"eyAi.....",
    "token_type":"Bearer",
    "expires_in":1799,
    "scope":"profile"
}
```

Or you can pass the `Request Body` as `Key-Value` pair. For example,

```
code:2510b2d0-c3b1-4426-bfa8-c4f721c94d8b
grant_type:authorization_code
scope:profile
client_id:fed6c7fc60454b62efaf9ca42d719fa2
```

```
client_secret:ed11ac1da0b54e06afeb88e10b9d3458
redirect_uri:https://manage.kony.com
```

> ***Important:*** For a custom identity service, you must pass the `X-Kony-RequestId` in the header (value for X-Kony-`RequestId` can be a random value.)

You can use the access token to invoke your Kony Fabric integration services.

7. To get the profile, pass the token obtained in the previous step:

   - Invoke a GET call on `https://<<accountnumber>>.auth.konycloud.com/oidc/profile` with the following contentheader:
     `Authorization : Bearer<space><token received in the previous step>`

     For example, `https://100000012.auth.konycloud.com/oidc/profile Authorization:Bearer eyAidHl…..H3mspKIPnAKzEbyc`

   - Or you can invoke a Curl call with the token received in the previous step:
     For example, `Curl -X GET " https://100000012.auth.konycloud.com/oidc/profile " -H "accept: application/json" -H "Authorization: Bearer eyAidHl….kVNGAz3jDd36MXuhWmaQ"`

8. To sign out of the OAuth Provider, invoke the following API in an internet browser.

   ```
   https://<<accountnumber>>.auth.konycloud.com/oidc/logout/<provider_name>?target_url=<redirect_url>
   ```

   For example,
   `https://100002634.auth.konycloud.com/oidc/logout/OAuthProvider11`

> ***Note:*** If you do not pass the target URL parameter, the API displays a logout success message. Otherwise, it redirects to the specified URL.

> ***Note:*** For more information on how you can integrate Kony OAuth Provider, User Repository, and OAuth 2.0 Identity services to create a basic login form, refer to a Base Camp article: Exploring Kony OAuth Provider.

19.4.15 Social Identity Providers

Kony Fabric provides a set of preconfigured OAuth 2.0 services under Social Identity. With Social identity services, a user can access resources through Kony Fabric applications. The Social Identity provides various popular services that include Google, Instagram, Microsoft, BOX, Facebook, LinkedIn, Amazon, and Yahoo.

19.4.15.1 How to Configure a Social Identity Service

To configure Google identity service, follow these steps:

1. Under the Identity service designer page, type a name for the service in the Enter Service Name text box.
2. From the **Type of Identity list**, select **Google** under the **Social Identity**.

   > ***Note:*** Fields marked with an asterisk are mandatory.

3. In the **Authorize Endpoint** text box, the default URL (https://accounts.google.com/o/oauth2/auth) is displayed. You can modify these details.
4. In the **Token Endpoint** text box, the default URL (https://www.googleapis.com/oauth2/v3/token) is displayed. You can modify these details.
5. In the **Callback URL** text box, the default URL is displayed. You cannot modify these details (https://<accountID>.auth.konycloud.com/OAuth2/Callback).

> ***Important:*** Configure your Google app to accept requests from authentication service by typing the auth.konycloud.com in the App Domain text field.

6. In the **Scope** text box, the default list of permissions is provided. The list can be modified. The list can contain more than one permission. The list must contain valid permissions separated with a space.
7. In the **Client Id** text box, enter the app key of the app instance that you created on Google.
8. In the **Client Secret** text box, enter the app secret key of the app instance that you created on Google.
9. After entering the above details, click on the **TEST LOGIN** button to verify the credentials.

   If you have not logged in to your the social identity service (for example: Gmail), the Console redirects you to the back-end identity provider's log-in page. Enter your credentials as required.

   The test results are displayed in the **Identity Response** dialog.
10. After entering the details in the **Client Secret** field, click **Save** to save the service. The system displays the Identity page. The Google identity provider is configured.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

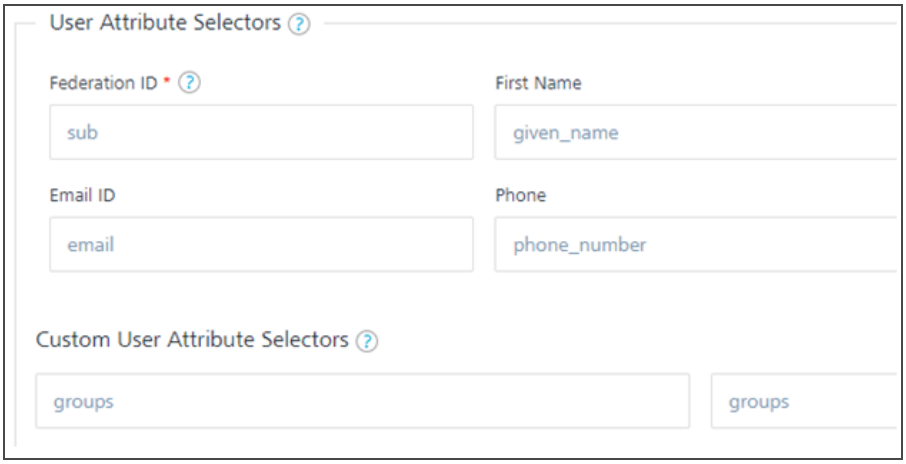
19.4.16 Groups Support for Identity Services

You can configure custom parameters to get information about groups from identity services.

Based on the groups' information in the result response of identity services, you can use the response from the identity services to handle your logic/code on the client side. You can also target functionality to a specific group.

19.4.16.1 Groups Support in Identity Services

The following identity service types support the configuration of custom properties to get information about groups.

Identity Service Type	Parameters that you need to configure in the Kony Fabric Console
Okta	Follow the give steps to configure custom parameters for an Okta Identity Service: 1. Go to the okta identity service configuration. 2. Add groups in the scope field • **Callback URL**:<URL> • **Scope**: openid profile email phone groups 3. Expand the **Advanced** section to view the **User attribute selectors**. You need to add a custom profile attribute selector with value as `groups`.  These steps will make sure groups are returned as part of the login response and as part of **getUserAttributes** API response. They can also be consumed using the identity scope parameters in integration service.

<table>
<tr><th>Identity Service Type</th><th>Parameters that you need to configure in the Kony Fabric Console</th></tr>
<tr><td>Lightweight Directory Access Protocol (LDAP)</td><td>Configure the following custom parameter in LDAP Identity Service type:
In Custom User Attribute Selectors section has Attribute Name and Attribute Selectors.
Attribute Name = groups
Attribute Selector = <code>memberof</code></td></tr>
<tr><td>Custom</td><td>For custom identity, Kony identity returns attributes from the backend login response. Therefor, if you want to receive the information about groups in the response for a custom identity service, the backend login response should include the groups attributes. You can refer to the following example for more information.

Sample Backend Login Response with groups information:

<pre>{
“user_attributes”:{
“first_name”:”John”,
“last_name”:”Doe”,
“groups”:[
“Everyone”,
“Admins”
]
},
“security_attributes”:{
“session_token”:”<backend_token>”,
…
…
},

}</pre></td></tr>
</table>

> ***Note:*** For more details on how to use Groups in apps, refer to Using Groups in an App.

19.5 How to Use an Existing Identity Service

Kony Fabric allows you to use an existing Identity service. You can add more than one services at a time from existing services.

1. Under the **Identity** page, click **USE EXISTING**.

   The **Existing Services** dialog that appears with a list of existing services. The service is added (linked) to your app and is available in the **Identity** page of your app.

   > ***Note:*** Existing Services contain a list of services created within the same parent account.

2. Select the check box for the desired services. If you want to add or clone more than one service, select the required check boxes from the existing services.

3. Click **ADD** button to reuse (link) an existing service. If any changes made to this service, the changes will affect all the apps using this service.

4. After a service is added successfully, click **CLOSE** to close the process dialog.

19.6 How to Unlink or Delete Multiple Existing Identity Services

Kony Fabric allows you to unlink or delete one or more existing identity services from the **Identity** list page.

To unlink or delete multiple existing identity services, follow these steps:

1. Go to the **Identity** tab. The page lists the existing services (if any). There is a check box provided for each service. By default, the check boxes are cleared for each service in the **Identity** list page.

2. Select one or more check boxes for services. The quick access bar for the selected services appears with actions such as **Unlink** and **Delete**.

    - **Unlink**: Allows you remove the service from the Identity tab of an app. When a service is unlinked, it is disassociated from a particular app.

      > ***Note:*** If you want to use an unlinked service, select the service from the **Existing Identity Service** dialog.

    - **Delete**: Allows you to delete a service.

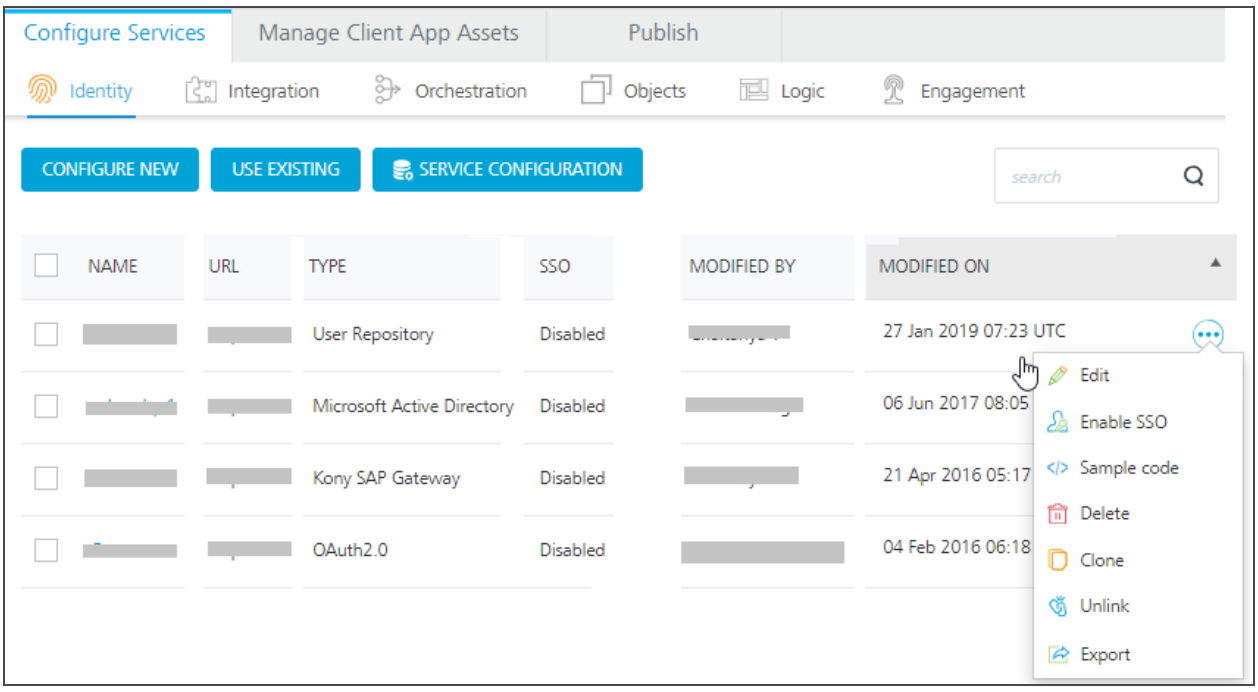
      > ***Note:*** If a service is a part of a published app, you can delete that service only after you unlink the service from all the published app.

    - **Clear All**: Allow you to clear the selection.

3. Click the desired (Unlink or Delete) button.

19.7 Context based Options

To perform various actions on an existing service, click the contextual menu of the required service.



The contextual menu contains the following options:

- **Edit**: Allows you to edit a service. After you edit a service, you have to republish all the apps that are using the service to apply the changes.

  > ***Note:*** To know more about publishing an app, refer to Publish an app.

- **Enable SSO**: Allows you to enable SSO between the mobile app and other apps that use the Identity service. You must republish the app for the new setting to take effect. If a Kony Fabric app uses multiple identity services, you must enable SSO for all the identity services linked to the Kony Fabric app. For more information about SSO for applications, see Application SSO.

- **Sample Code**: A dynamic code is generated based on the configuration of a service. You can use this code in your SDK.

- **Delete**: Allows you to delete a service.

  > ***Note:*** If a service is part of a published app, you can delete that service only after you unlink the service from all the published apps.

- **Clone**: Allows you to duplicate an existing identity service.

  > ***Note:*** From Kony Fabric V8 SP3 onwards, when you click **Clone**, the system generated new name appears for the cloned identity service, in the list. The new name remains in the edit mode until you click anywhere else on the screen. If you want, you can rename it. Changes made to a cloned identity service will not impact the original service.

- **Unlink**: Allows you remove the service from the **Identity** tab of an app. When a service is unlinked, it is disassociated from a particular app.

- **Export**: Allows you to export the identity service into your local system. The exported Kony Fabric Service Package is a file .zip file. You can import the service package to your app in API Management > Identity.

19.8 How to Configure Identity Session Timeout and HTTP Message Body Integrity

Kony Fabric supports configuring session timeout (idle timeout and fixed timeout) for an app identity session. In Apps, Kony Fabric supports configuration of **Identity Session Idle Timeout** that applies across all of your apps, as well as support for Enable HTTP Message Body Integrity Checking for an Application. The following procedures describe how to configure session timeout and client app security.

> ***Important:*** HTTP Integrity does not support Scheduler job.

19.8.1 How to Configure App Session Settings

You can configure either an idle timeout or fixed timeout for apps in the **Applications > Identity** page.

- **Idle Timeout**: Specifies the number of minutes that a session can remain idle before Kony Fabric automatically terminates the app.
  - **Identity Session Idle Timeout**: When an app session on a device remains idle for a certain period of time, the app session expires automatically. The user will need to log into

the app again.

- **Maximum Session Duration**: An apps log-in session is active until the maximum session duration time is met.

- **Fixed Timeout**: Specifies the session's idle timeout (HH:SS) of an app. When the timeout is reached, the session expires automatically, and the user will need to log into the app again.

To configure an Identity Session Idle Timeout, do the following:

1. Click on an app in the Apps, and then in the **Identity** tab, click the **SERVICE CONFIGURATION** button.
2. In Identity Session Timeout, do either of the following:
   - Click the **Idle Timeout**.
     - Enter the hours/seconds (HH:SS) in the **Identity Session Idle Timeout**.
     - Enter the hours/seconds (HH:SS) in the **Maximum Session Duration**.

   Or

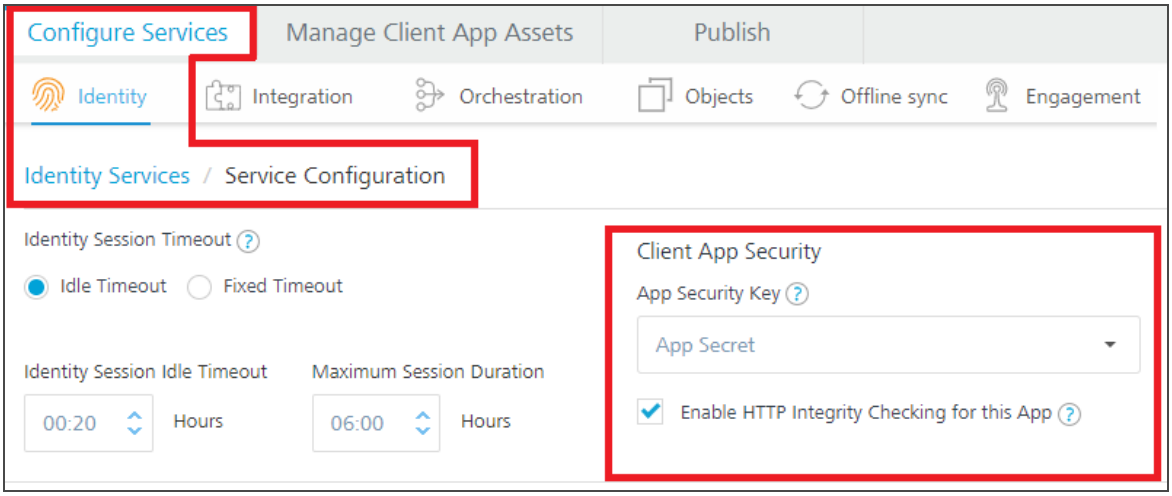
   - Click the **Fixed Timeout**, and enter the hours/seconds (HH:SS) in the **Fixed Identity Session Duration**.
3. Click **Save**.
4. Publish or republish the app to reflect the changes.

19.8.2 How to Enable HTTP Message Body Integrity

The Client App Security feature helps to secure data exchanged between a client app and a server app. Enterprise class applications may need to ensure that network traffic being exchanged between the server and client app is not tampered with. This feature detects and reports network traffic tampering on the data exchanged between the server and client app.

To enable HTTP Message body integrity for an Application, follow these steps:

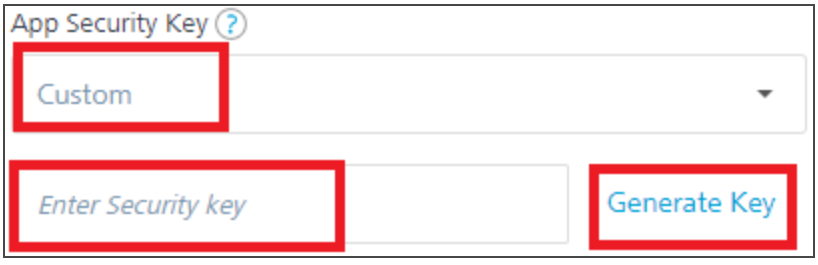
1. Click on an app in the Apps, and then in the **Identity** tab, click the **SERVICE CONFIGURATION** button to display the **Client App Security** section.



The App Security Key is used for HTTP Message body integrity checking and other client security features managed by the client app SDK.

2. In the **App Security Key**, you can select the default app security key or generate a custom security key. To select an app secret key, follow these steps:
    a. From the **App Security Key**, select **App Secret**. The App Secret is selected by default.

    a. If you want to generate a custom key, select **Custom** from the **App Security Key** list.



    b. Enter the custom security key and click OK. Otherwise you can generate a custom

security key by clicking the **Generate Key**. A security key is generated.

> ***Note:*** For more information, refer Usage of Custom App Security Key.

3. Select the **Enable HTTP Integrity Checking for this App** check box.

   The HTTP Message body integrity Checking signs the outbound HTTP requests from a client application and verifies in inbound HTTP response signature to further enhance the security between the client app and backend services.

4. Click **SAVE**.

> ***Note:*** More details to configure CORS and Identity Session Timeout in APIs

19.9 How to Enable Multi-Factor Authentication

Kony Fabric provides the ability to use Multi-factor authentication (MFA) to allow an additional layer of security for all user profiles associated with a Kony Fabric account. MFA requires each user to provide a secondary email address or phone number, which is associated with their profile. When a user logs in, a validation code is sent to the email or phone number, which the user must provide in addition to their other login credentials.

Each user can enable MFA on their profile individually, or an admin can enable MFA across all user profiles associated with the account. If enabled by the admin, each user associated with that account will need to activate MFA in their profile to continue to have access to the account.

> ***Important:*** MFA option is available only for users who have Admin and Owner roles in Fabric. MFA is applicable for AWS Cloud only.

19.9.1 Enabling Mutli-Factor Authentication for all Users

To enable multi-factor authentication for all users, do the following:

1. From the left pane, select **Settings**.
2. At the top of the Settings page, select **User Authentication**, and then select **Multi-Factor Authentication**.
3. Next to **Do you require multi-factor authentication**, select **Yes**.

19.9.2 Disabling Mutli-Factor Authentication for all Users

To disable multi-factor authentication for all users, do the following:

1. From the left pane, select **Settings**.
2. At the top of the Settings page, select **User Authentication**, and then select **Multi-Factor Authentication**.
3. Next to **Do you require multi-factor authentication**, select **No**.

19.9.3 Configuring a User Profile for Multi-factor Authentication

To enable MFA for your user profile, do the following:

1. Select your user profile from the top right corner of the Console.
2. Select the **Security** tab.
3. Under Virtual MFA applications, review the list of virtual multi-factor authenticator apps and verify that your secondary device has an appropriate virtual MFA installed.
4. Click **Activate MFA**.
5. In Activate MFA, under **Phone Number**, type in your phone number.
6. Under **Secondary Email**, type your secondary email address, and then click **Send Link**.
7. Under **Enter Validation Code**, type the validation code you received in your email, and then click **Verify**.

8. Click **Next**.
9. If your secondary device can scan QR Codes, scan the QR Code displayed, enter the two authentication codes, and then click **Activate**.
10. If your device cannot scan QR Codes, select **Manual Configuration**. On your secondary device, enter the secret key that is provided, and then click **Activate**.

19.9.4 Disabling Multi-factor Authentication on a User Profile

Users can choose to deactivate MFA at any time in their user profile settings.

> ***Important:*** If the administrator has enabled MFA across all user profiles associated with an account, then the user will lose access to the account unless they re-enable MFA.

To disable MFA for your user profile, do the following:

1. From your user profile, select **Settings**.
2. Select the **MFA** tab.
3. Click **Deactivate MFA**.
4. Click **Deactivate**.

> ***Note:*** After MFA has been deactivated, you will be prompted to log out and then log back in to your account.

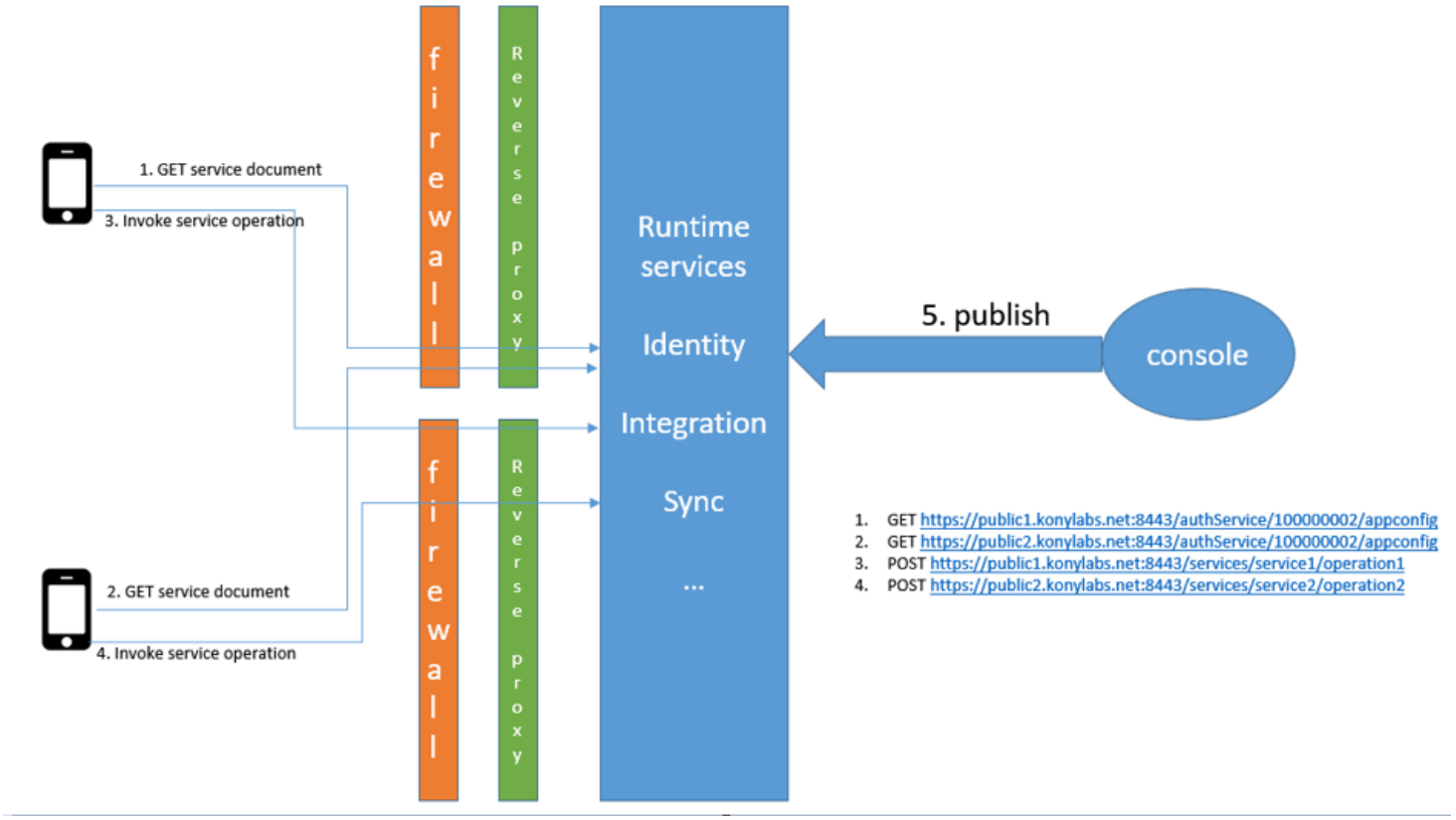
19.10 Support to MAP Public URLs - Reverse Proxy (on-premises)

In case of on-premises, if your Kony Fabric is installed behind a reverse proxy and registered using a private URL or IP, you will need to expose identity and services end-points as public URLs. A reverse proxy is configured for exposing public URLs and then routing them to private Kony Fabric URLs. When you create an Identity or Integration service, which is installed behind a reverse proxy, on Console, the service endpoints are exposed via intranet or private URLs during design time. You can access the service, however, via public URLs at runtime.

In some cases, there can be multiple reverse proxies depending on the region. For example, apps in **Region1** may access Kony Fabric via a reverse proxy in **Region1**, while the apps running in **Region2** may access Kony Fabric via a reverse proxy in **Region2**. In addition, the service endpoint URLs can be different when accessed from **Region1** vs **Region2**.

To get the desired results from private and public URLs, you must modify the mapping selection from public URLs to private URLs in the identity server.

19.10.1 Sample Deployment Topology of Multiple Tenant URL Support in Identity Server



API Commands to map Public URLs

The following is a sample code to map Identity public URLs (`CUSTOM_TENANT_URLS`).

```
 POST
https://mfprivate.konylabs.net:8443/authService/api/v1/setup/tenants/
100000002/properties
With headers
X-Kony-Authorization: <auth token of admin>
Content-Type: application/json
Request payload:
{
```

```
    "name": "CUSTOM_TENANT_URLS",
    "value": "
{\"https://public1.konylabs.net:8443/authService/100000002\":
\"https://public1.konylabs.net:8443/authService/100000002\",
\"https://public2.konylabs.net:8443/authService/100000002\":
\"https://public2.konylabs.net:8443/authService/100000002\"}"
}
```

The following is a sample code to map service URLs (`CUSTOM_TENANT_SVC_URLS`).

```
 {
"name": "CUSTOM_TENANT_SVC_URLS",
"value": "
{\"https://public1.konylabs.net:8443/authService/100000002\":

{\"https://mfprivate.konylabs.net:8443/services\":
\"https://public1.konylabs.net:8443/services\",
\"https://mfprivate.konylabs.net:8443/kpns\":
\"https://public1.konylabs.net:8443/kpns\",
\"https://mfprivate.konylabs.net:8443/syncservice\":
\"https://public1.konylabs.net:8443/ syncservice\" }
, \"https://public2.konylabs.net:8443/authService/100000002\":

{\"https://mfprivate.konylabs.net:8443/services\":
\"https://public2.konylabs.net:8443/services\",
\"https://mfprivate.konylabs.net:8443/kpns\":
\"https://public2.konylabs.net:8443/kpns\",
\"https://mfprivate.konylabs.net:8443/syncservice\":
\"https://public2.konylabs.net:8443/ syncservice\" }
}}"
}
```

The following table details sample private and public URLs.

URL	Type of URL
`https://mfprivate.konylabs.net:8443`	Private URL
`https://public1.konylabs.net:8443`	Public URL1
`https://public2.konylabs.net:8443`	Public URL 2

Important: You must restart the Identity server after this configuration change.

19.11 Single Sign-On

Single Sign-on (SSO) is a session and user authentication process. It allows you to access multiple applications by logging in only once with one set of login credentials.

The SSO feature supports applications using the same identity service.

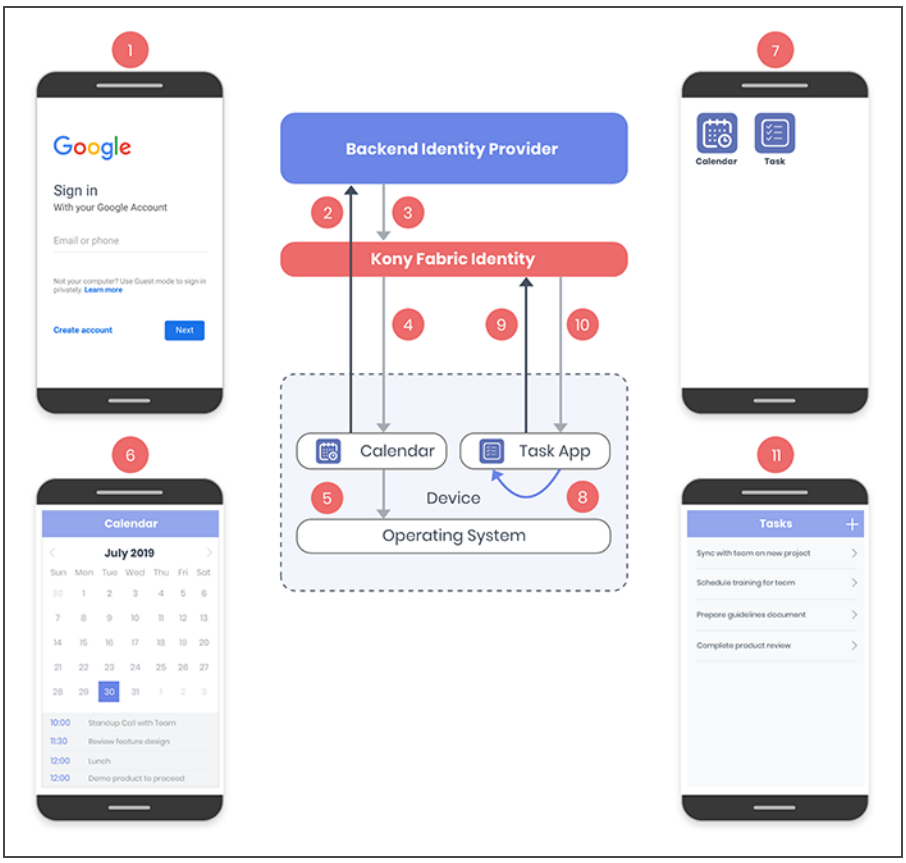
For example: **Application1** and **Application2** use a **Google Identity Provider**. When you sign in to **Application1** by using the credentials for the Google Identity Provider, you need not sign in to **Application2**. You will be automatically signed in to **Application2**.

Note:

- Single Sign-On is supported for Android, iOS, and Web platforms.
- When an Identity Service is mapped using the Data Panel in Kony Visualizer, the SSO feature does not function.

19.11.1 Use Case

To understand the functionality of the SSO feature, consider two applications namely Calendar app and Task app using the same identity provider. The following flow diagram illustrates how SSO feature works on user's device at runtime:



The workflow of these applications is as follows:

1. The user first starts the Calendar app, and then types the user credentials to log on to the app.
2. The Calendar app sends the credentials to the Backend Identity Provider for authentication.
3. On receiving the credentials, the Backend Identity Provider returns an authentication token to Kony Fabric Identity.
4. Kony Fabric Identity then sends an SSO token to the Calendar app.
5. The Calendar app stores the SSO token on the device.
6. On successful authentication, the Calendar app loads the user's calendar.
7. The user then starts the Task app.

8. The Task app fetches the SSO token from the device and sends it to Kony Fabric Identity for validation.
9. After the SSO token is validated, Kony Fabric Identity authenticates the user and sends the SSO token to the Task app.
10. Finally, the Task app is launched without the user having to log on.

19.11.2 SSO Configuration

To configure SSO in your applications, perform the following tasks:

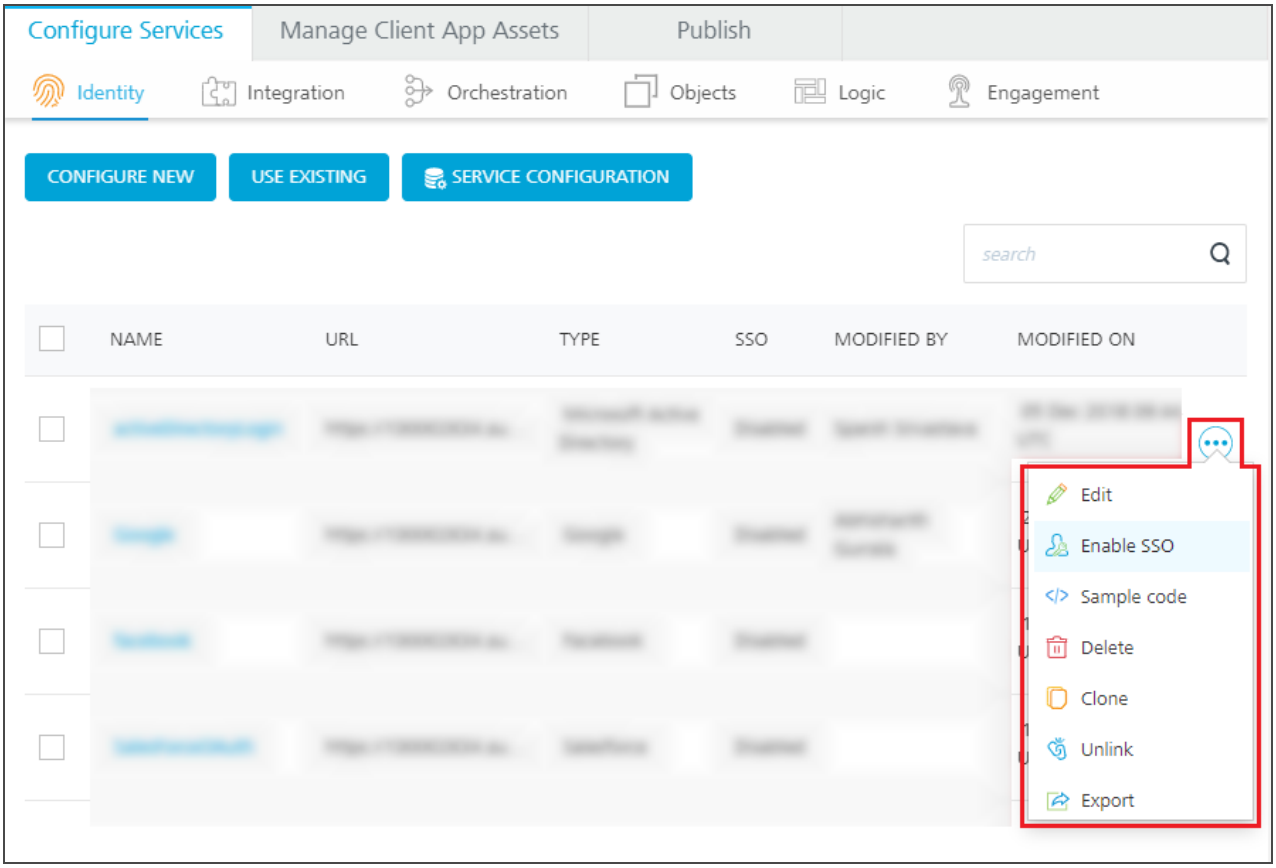
- Enable SSO in Kony Fabric Console
- Configure Client Applications on Kony Visualizer
- Configure SSO Permissions for Native Platforms

19.11.2.1 Enabling SSO in Kony Fabric Console

To enable SSO in Kony Fabric Console, follow these steps:

1. Sign in to Kony Fabric Console.
2. Go to the **Apps** tab and open the app for which the Single Sign-On feature is to be enabled.

3. In the Identity section of the services, click the contextual menu of the required service.



4. A list of options appears. Select the **Enable SSO** option from the list.

5. After the SSO feature is enabled, republish the app for the SSO feature to come into effect.

19.11.2.2 Configuring Client Applications on Kony Visualizer

The client applications are built on Kony Visualizer and they interact with the Kony Fabric server using the Kony Fabric SDK.

> ***Important:*** Ensure that the Kony Fabric application, which you have configured earlier is associated with your Kony Visualizer project.

To enable SSO on the client application for **login call**, add the following code in Kony Visualizer:

```
//Sample code to authenticate to Kony Fabric client
var serviceName = "identity_service_name";
var identitySvc = KNYMobileFabric.getIdentityService(serviceName);

var options = {};
var loginOptions = {};

options["userid"] = "userid";
options["password"] = "password";
loginOptions["isSSOEnabled"] = true;

option["loginOptions"] = loginOptions;

identitySvc.login(options, function(response) {
    kony.print("Login Success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Login Failure: " + JSON.stringify(error));
});
```

19.11.2.3 Configuring SSO permissions for Native Platforms

The SSO tokens are stored in shared space/keychains of the devices. You must configure application settings to ensure that these tokens are shared among all the SSO enabled apps so that the SSO is implemented and other applications can access these tokens.

Permissions in Android Devices

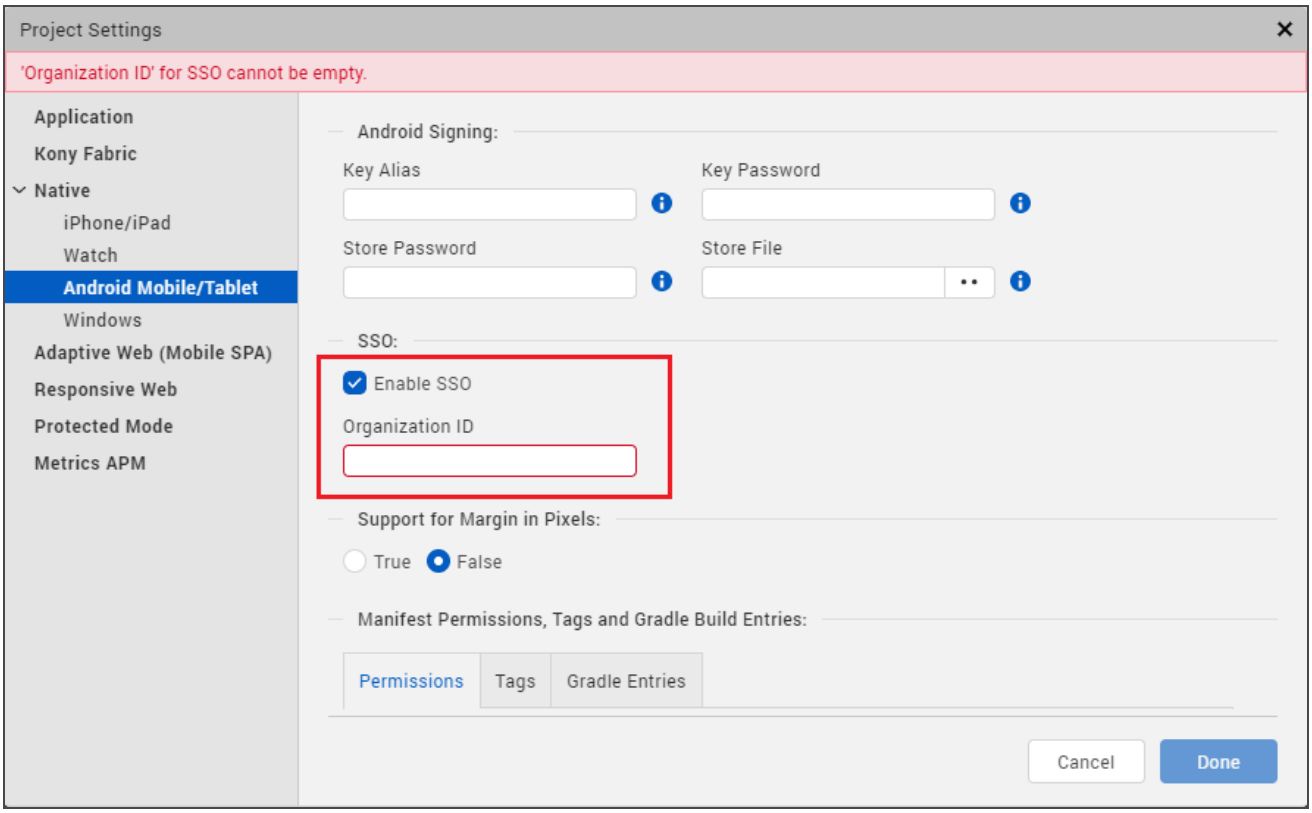
Permissions in iOS Devices

Permissions in Android Devices

For the SSO feature to work in the Android devices, follow these steps.

1. Go to **Project Settings** > **Native** > **Android Mobile/Tablet**.
2. In the **SSO** section,
    i. Select the **Enable SSO** check box.
    ii. In the **Organization ID** box, type your organization ID.

> ***Note:*** The Organization ID must be unique for every organization so that the SSO token is shared amongst a particular organization only.

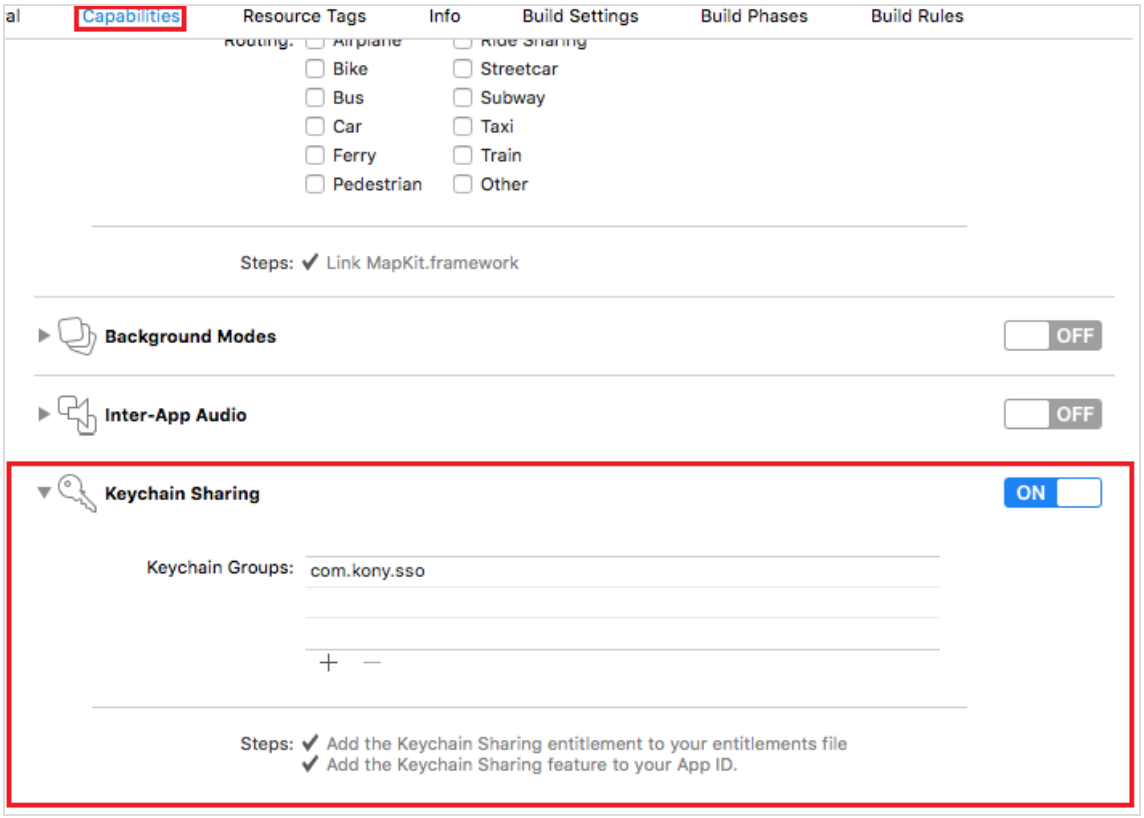


3. Click **Done** to save the changes.
4. Repeat the steps for each app that needs SSO capability to be enabled.

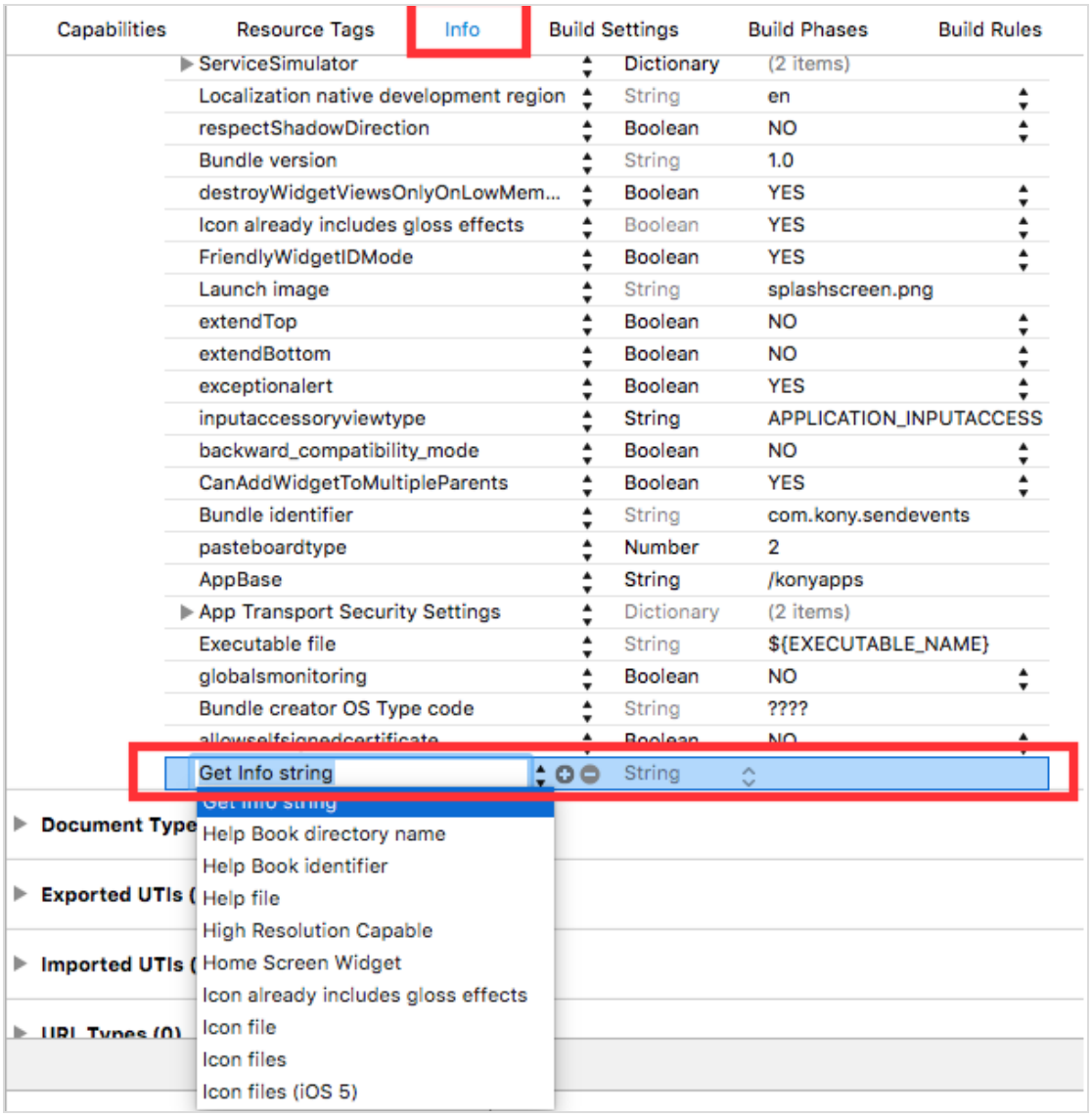
Permissions in iOS Devices

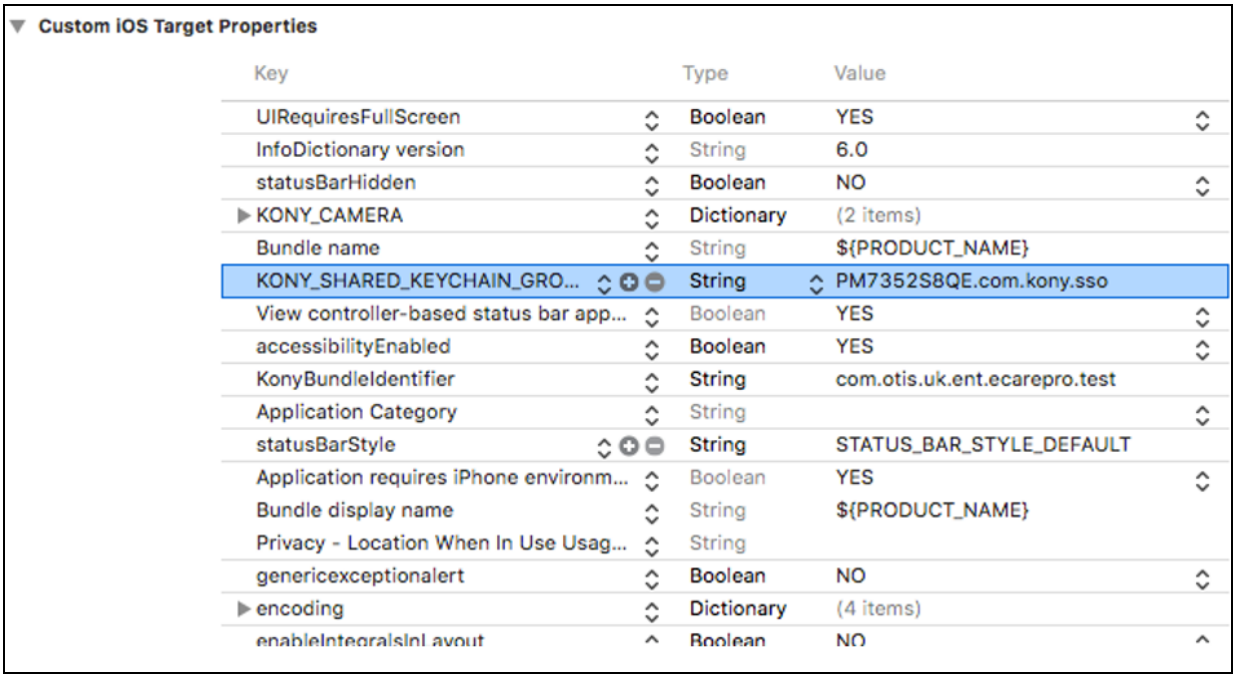
To configure permissions for iOS devices, follow these steps:

1. Open the project in Xcode.
2. In the **Capabilities** section, enable the **Keychain Sharing** option and enter a name for the keychain group.

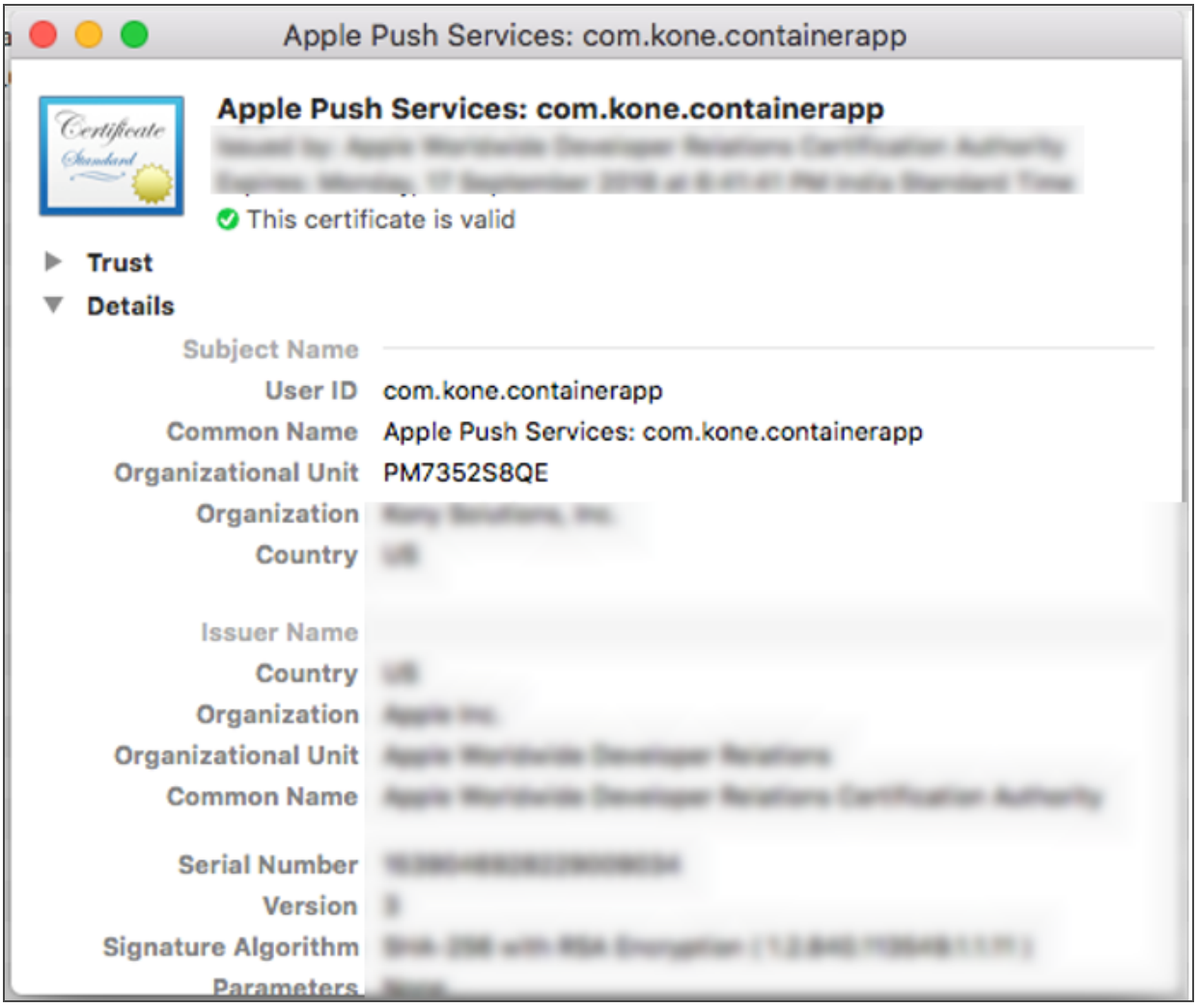


3. Go to **Info** tab in Xcode project. Create a new property in the **Custom iOS Target Properties** section called `KONY_SHARED_KEYCHAIN_GROUP` of the string data type. Set the name of the keychain group created in the previous step as value for the new property. For example, `KONY_SHARED_KEYCHAIN_GROUP = PM7352S8QE.com.kony.sso`





Here, `PM7352S8QE` is an Organizational Unit value in the certificate, which can be obtained from the developer certificate.



4. Repeat the steps for each app that needs SSO capability to be enabled.

> ***Important:*** For all apps that intend to share the same SSO group, the key name string must be the same. The key name string must be different for different SSO groups.

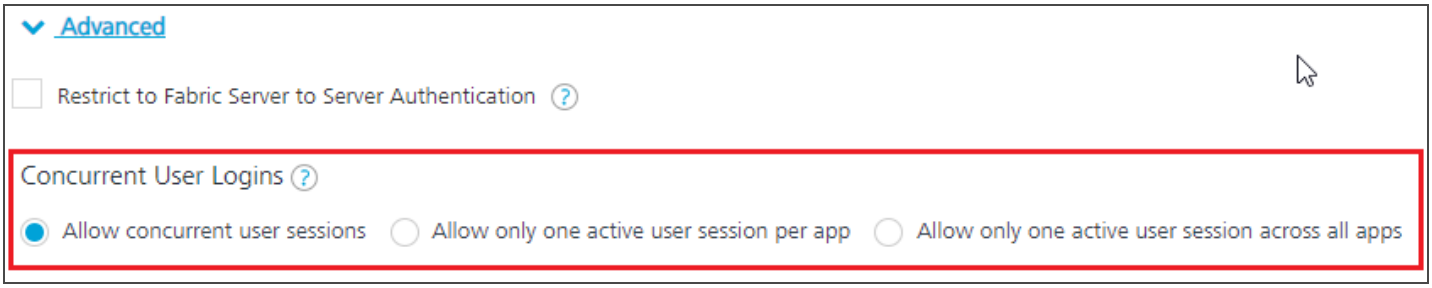
19.11.3 Additional Information

- For Android, if you upgrade your Visualizer from V8 SP3 or earlier versions to V9, delete the previous SSO configurations and reconfigure the SSO as described in the SSO Configuration section.
- For Visualizer V8 SP3 and earlier versions, SSO will not work for the first time due to parity issues. It will work in the subsequent executions effectively.

19.12 Concurrent User Sessions

When an enterprise uses an identity service in the Fabric app, it would want to have control over how to handle concurrent logins. For example, a banking enterprise may not want the same user to log in concurrently from different channels of the app.

Kony Fabric provides a series of options to control concurrent user sessions. You can access the **Concurrent User Logins** functionality in the **Advanced** section of an identity service definition page.



Use Case One: For a banking enterprise, you (app developer) are building a retail banking app and have linked an identity service (for example, Microsoft Active Directory) with a Fabric app and built client app for different channels (Android, iOS, and Web). If the app user can log in to Android and decides to login into Web app by default user session would be working in both the channels. However, for better security you may want to restrict the user session to be active only in one channel then you can use the options available in the **Concurrent User Logins** section. By using the options in this section you can have the user session in Android invalidated once the user logs into Web session.

Use Case Two: For the same banking enterprise in next stage you plan to build a consumer lending application. You create a new Fabric application and link the same identity service used in retail banking app and build the client application for multiple channels. In this case for better security if you want to restrict the app user to login to only one of the user applications (for example, if the user logs into a retail banking application and then wanted to check his loans using consumer lending application you want to invalidate the retail banking application session), you can use the options available in the **Concurrent User Logins** section.

In the **Concurrent User Logins** section, you can select any one of the following three options to configure user sessions:

- Allow concurrent user sessions (no restrictions): When this option is selected, an app user with unique credentials is allowed to have multiple apps from different instances.
- Allow only one active user session per app: Logging into simultaneous instances of **the same app** is not supported. When this option is selected, an app user can log in to only one instance of client apps linked to a specific Fabric app which has the identity service linked. If the same user tries to log in to another instance of the same app, the previous sessions of the user belonging to the same app and identity service combination are invalidated. **Use Case One** can be achieved using this option.
  User Session uniqueness is determined by a combination of Fabric apps, an identity service, and a user.

  > ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session per app.

- Allow only one active user session across all apps: Logging to simultaneous instances of **the same app or across apps** is not supported. When this option is selected, a unique app user can log in to only one instance of client apps linked to all Fabric apps using the identity service. If the same user tries to log in to another instance of the same app or different app, the previous sessions of the user are invalidated. **Use Case Two** described can be achieved using this option.
  User Session Uniqueness is a combination of Fabric apps and a user.

> ***Important:*** Apps enabled for SSO will not work if the option is selected, Allow only one active user session across all apps.

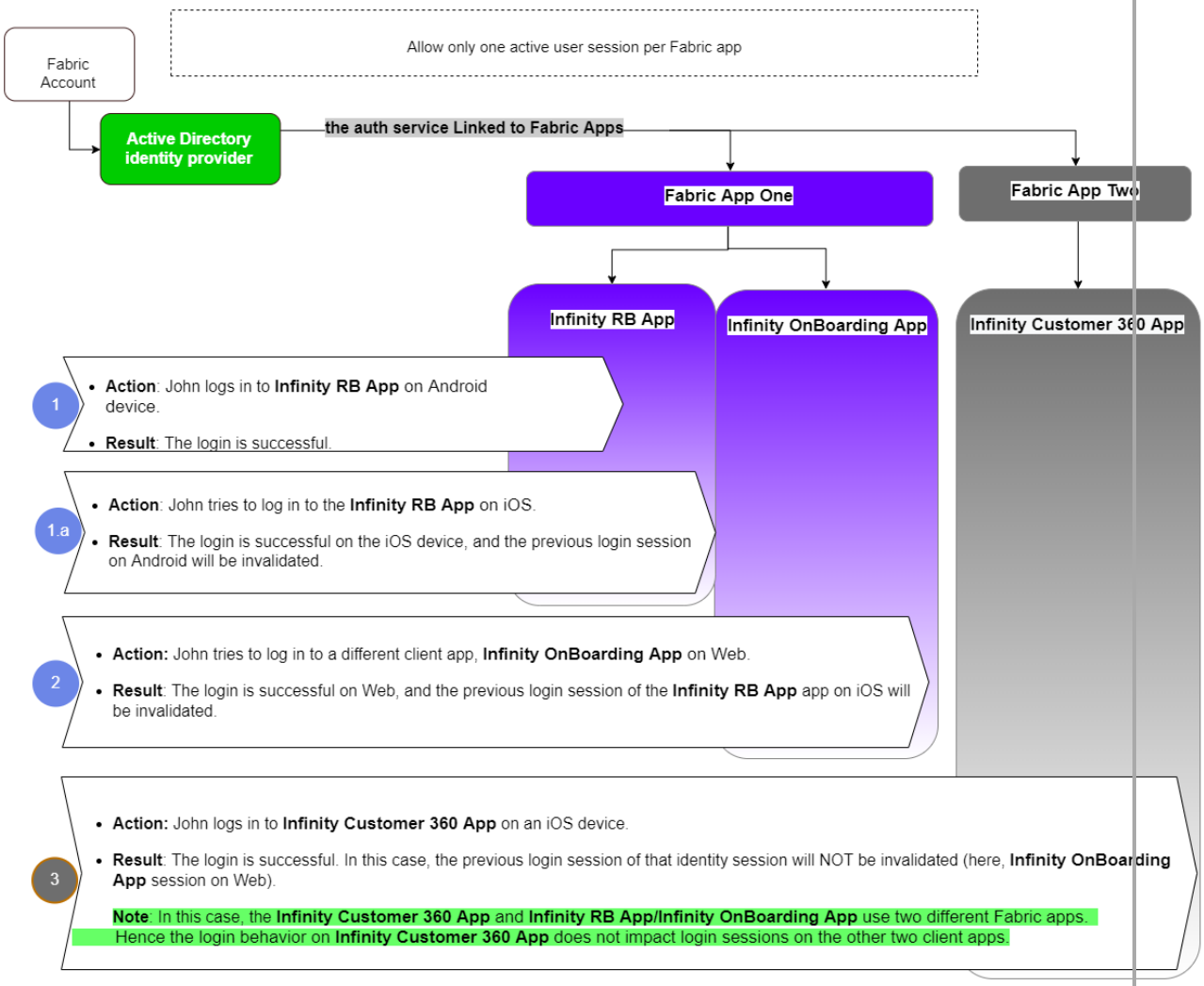
The following tables provide detailed information on Concurrent User Logins.

Option 1	**Allow concurrent user sessions (no restrictions).** This is the default option.
For example, An app user using the unique identity service is allowed to log in to multiple apps simultaneously in all instances, such as iOS, Android, and Web browser.	

Option 2	Allow only one active user session per app

For example, The `Infinity RB`, `Infinity OnBoarding`, and `Infinity Customer 360` apps are linked to an identity service (for example, Microsoft Active Directory). A user logs in to the `Infinity RB` app on an Android device successfully. When the user tries to log in to another instance of the **same app**, the login is successful. However, the previous session of that identity service is ended (here, `Infinity RB app` session on Android).

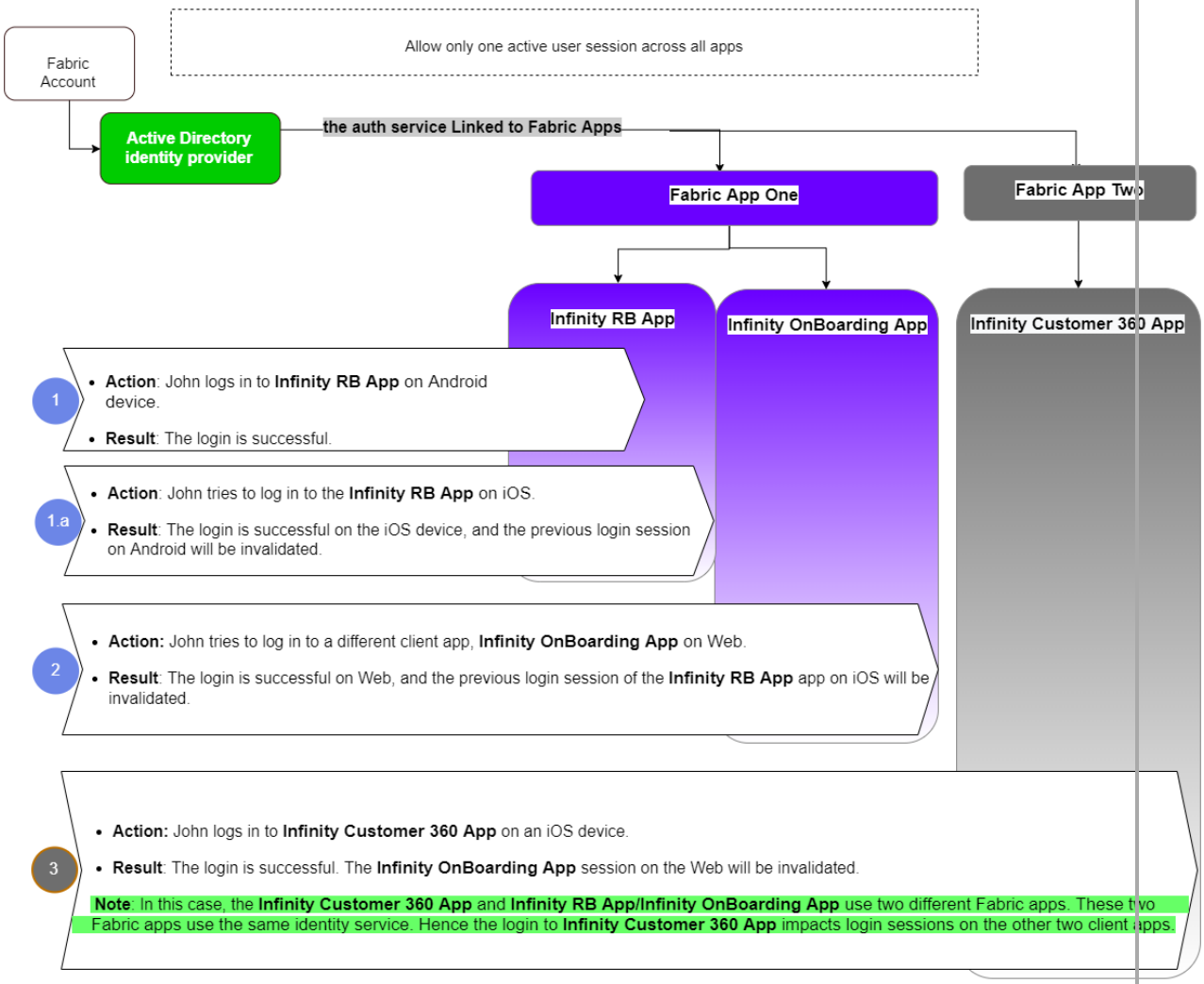
> ***Note:*** In this case, the `Infinity Customer 360 App` and `Infinity RB App/Infinity OnBoarding App` use two different Fabric apps. Hence the login behavior on `Infinity Customer 360 App` does not impact login sessions on the other two client apps.



Option 3	Allow only one active user session across all apps.

For example, The `Infinity RB`, `Infinity OnBoarding`, and `Infinity Customer 360` apps are linked to an identity service (for example, Microsoft Active Directory). A user logs in to the `Infinity RB` app on an Android device successfully. When the user tries to log in to another instance of the **same app or a different app**, the login is successful. However, the previous active sessions of the apps are ended (here, `Infinity RB` app session).

> ***Note:*** In this case, the `Infinity Customer 360 App` and `Infinity RB App/Infinity OnBoarding App` use two different Fabric apps. These two Fabric apps use the same identity service. Hence the login to `Infinity Customer 360 App` impacts login sessions on the other two client apps.



19.12.1 How to Configure Concurrent User Active Sessions for Apps

1. Log in to Kony Fabric Console.
2. In the **Apps**, click **ADD NEW** to create an app.
3. In the **Configure Services** tab > **Identity** service tab > click **CONFIGURE NEW**.
4. In the service designer page of the identity service, configure the required details.
5. Click the **Advanced** section > **Concurrent User Logins** section, select one of the following options:
   - Allow concurrent user sessions (no restrictions).
   - Allow only one active user session per app
   - Allow only one active user session across all apps.
6. Click **SAVE** to save the service details.

20. Legacy Services - SAP JCo Connector and Scraper Connector

Kony Fabric supports legacy services like SAP JCo and Scraper connectors. Legacy services include services that are originally created by using Kony Studio V 6.5 or lower, or services that are manually created using external tools. Apps whose services are created manually or outside Kony Studio are called Consolidated Services Definition (CSD) apps.

You cannot create new services using legacy connectors such as Scraper or SAP JCo directly in Kony Fabric. If you migrate apps with the legacy services into Kony Fabric, you can enable Kony Fabric apps with the services.

> ***Note:*** If your application is built in Kony Studio versions before th 6.0, upgrade your project to current version format by using Kony Studio 6.5, and then migrate the app. For more details, refer to Migrate a Project from an Earlier Version of Kony Studio or Visualizer.

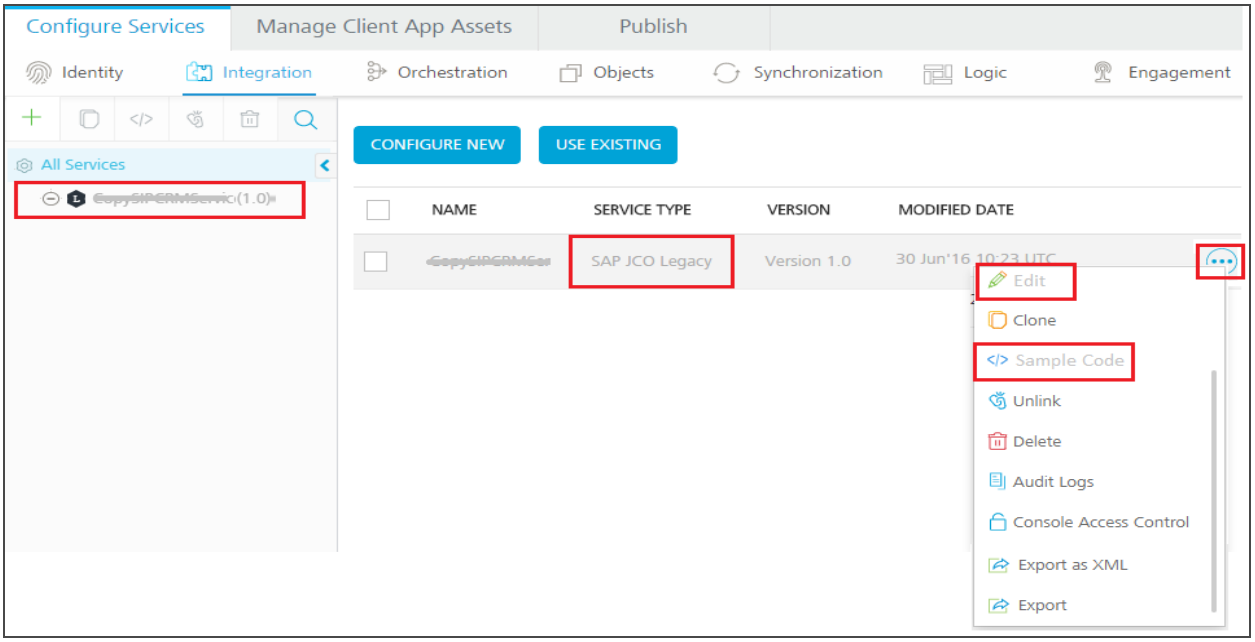
20.1 Limitations

Read the following limitations for legacy services before migrating your apps to Kony Fabric.

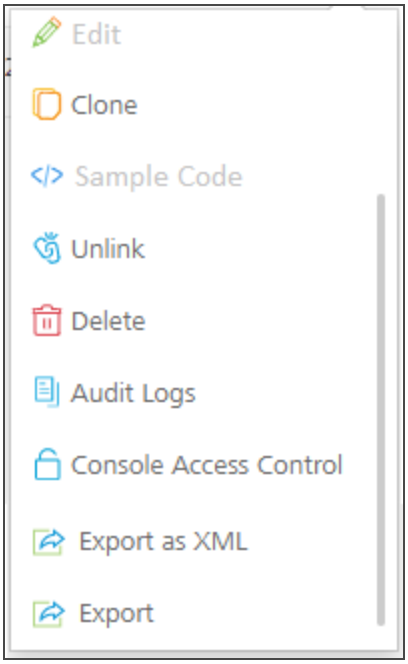
20.1.1 Limitations - SAP JCo Connector

- Due to limitations of the SAP JCo connector, Kony Fabric does not support middleware and services together to connect to same or different SAP systems on WebLogic, JBoss, and WebSphere. SAP JCo connector can be enabled either on middleware or services.

- When an app that contains an SAP JCo service is migrated, the **Service Type** field is filled in **SAP JCO Legacy** in the integration tab of the Kony Fabric Console, shown below:



- In the Kony Fabric Console, you can publish SAP JCo services like the other services supported in the Kony ecosystem.
- You cannot edit SAP JCo services in the Kony Fabric Console. If you click the **Settings cogwheel** button for the service, the following app menu appears.

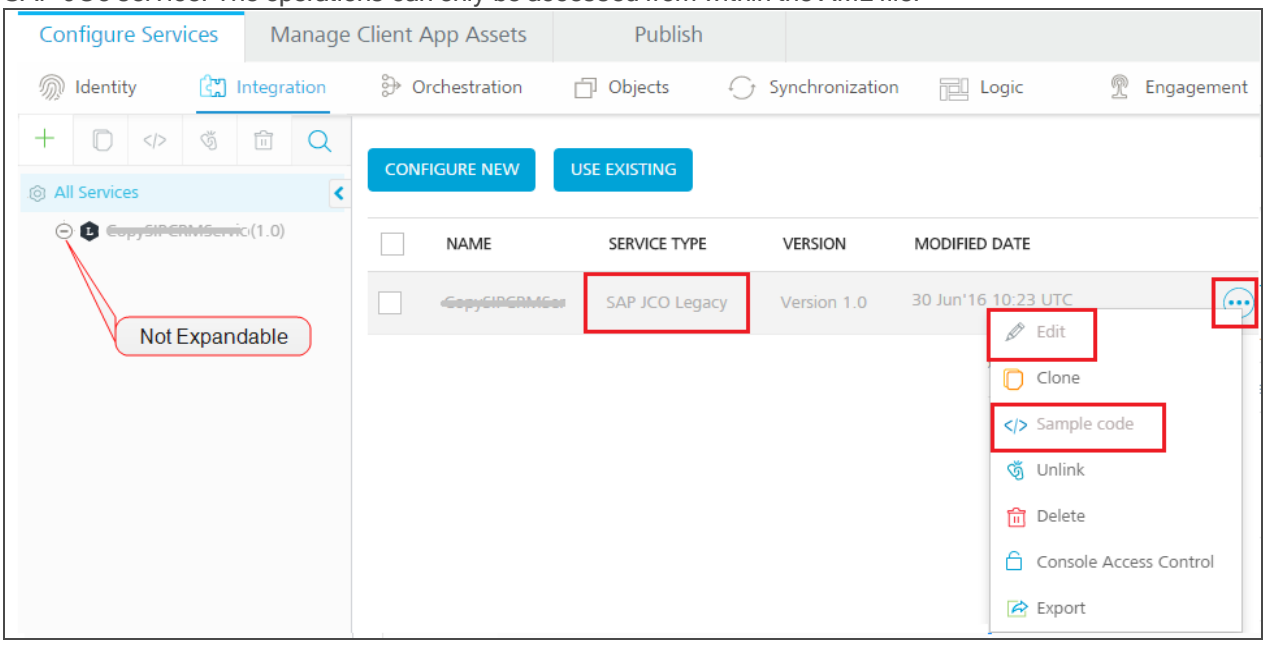


Note the following supported actions:

- **Edit** and **Sample code** are not supported.
- **Clone**, **Unlink**, **Delete**, **Console Access Control**, **Export** are supported. For details, refer to How to Use Actions on an Existing Integration Service

- You cannot view operations of an SAP JCo service in the Kony Fabric Console.
- An SAP JCo service is not clickable in Kony Fabric Console. You can only access operations for an SAP JCo service via the service definition XML directly. The **Configure Operation** button support is not available in the console.
- You cannot create SAP JCo services in Kony Fabric Console.

- For SAP JCo services, when you expand the tree view on the left pane in the integration service designer, no operations will be shown even if operations exist within the service definition XML of an SAP JCo service. The operations can only be accessed from within the XML file.

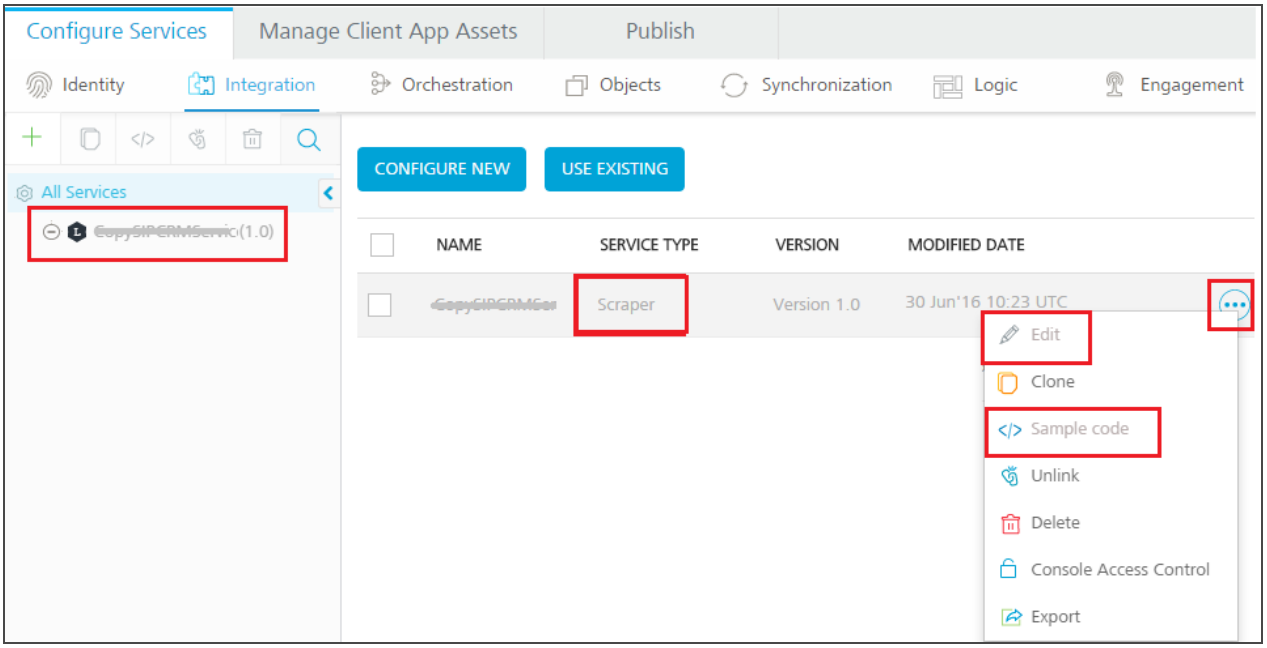


- While creating an orchestration, object or synchronization service in MobileFabric 7.x and later, legacy services are not available for linking.
- When an app that contains a Synchronization service referenced by legacy services as data source is migrated, sync scopes can only be viewed in Kony Fabric. You cannot edit the sync scopes.
- When an app that contains an Orchestration service referenced by legacy services is migrated, you can only view operations of the Orchestration service in Kony Fabric. You cannot edit the Orchestration service.

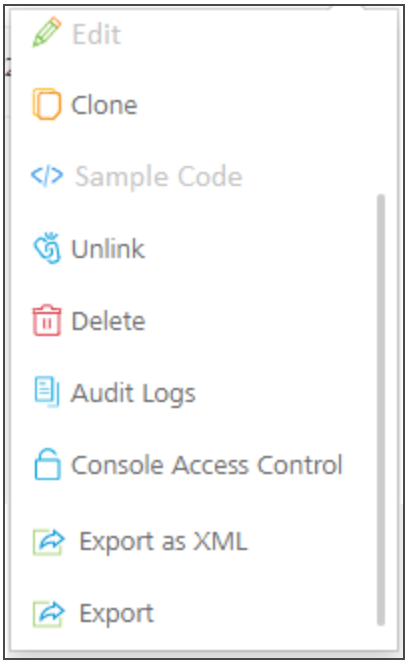
20.1.2 Limitations - Scraper Connector

- For each scraper service, there should be a corresponding dsl file.

- When an app that contains a scraper service is migrated, then the **Service Type** field is filled in **Scraper** in the integration tab of Kony Fabric Console, shown below:



- In Kony Fabric Console, you can publish Scraper services like the other services supported in the Kony ecosystem.
- You cannot edit scraper services in Kony Fabric Console. If you click the **Settings cogwheel** button for the service, the following app menu appears.

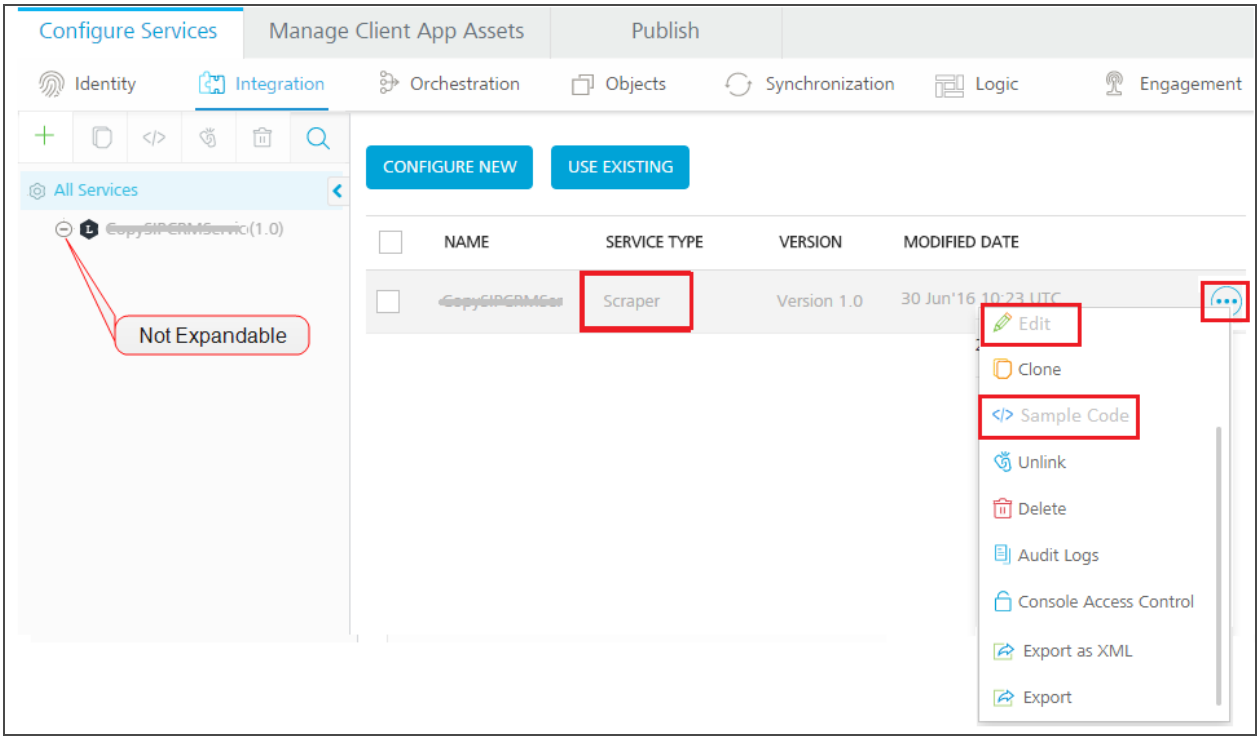


Note the following supported actions:

- **Edit** and **Sample code** are not supported.
- **Clone**, **Unlink**, **Delete**, **Audit Logs**, **Console Access Control**, **Export as XML**, and **Export** are supported. For details, refer to How to Use Actions on an Existing Integration Service.

- You cannot view operations of a scraper service in Kony Fabric Console.
- Scraper service is not clickable in Kony Fabric Console. You can only access operations for a scraper service via the service definition XML directly. The **Configure Operation** button support is not available in the console.
- You cannot create scraper services in Kony Fabric Console. To create scraper services, use a version of Kony Studio 6.5 or older.

- For scraper services, in the integration service designer, the tree view on the left pane when expanded, it will not show any operations. The operations can only be accessed from within the XML file.



- Ensure that DSL files and the corresponding Java services have the same name. Otherwise, the migration fails.
- While creating an orchestration, object, or synchronization service in MobileFabric 7.x and later, legacy services are not available for linking.
- When an app that contains a synchronization service referenced by legacy services as a data source is migrated, sync scopes can only be viewed in Kony Fabric. You cannot edit the sync scopes.
- When an app that contains an Orchestration service referenced by legacy services is migrated, you can only view operations of the Orchestration service in Kony Fabric. You cannot edit the Orchestration service.

> ***Note:*** Only one property file is allowed. The property file name must match the appID in the scraper servicedef file.

20.1.2.1 Scraper services with a custom classname

As custom classnames are no longer supported for scraper services, you must modify the custom classname to `com.kony.scrapper.gc.ScrapperJavaService`.

20.2 How to Enable SAP JCo Configurations on Standalone JBoss Server

Middleware.war and Services.war are not deployed on JBoss if SAP is enabled. To enable SAP JCo on standalone for JBoss, follow these steps:

1. Place the SAP JCo DLL file in the **System32** folder for Windows. For Linux, place the SAP JCo SO file in the user's **lib** folder.
2. Create a global module for SAP JCo jar inside `<jboss_home>\modules\system\layers\base\com`, and map it in the `standalone.xml`.
    a. Create the global module folder - for example, `sap\main`.
    b. Add the `module.xml` and SAP JCo jar files in the folder that you created.

    The following is sample content of the module.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="com.sap">
        <properties>
                        <property name="jboss.api"
value="unsupported"/>
        </properties>
        <resources>
                        <resource-root path="sapjco3.jar"/>
        </resources>
```

```
</module>

-       Add below mapping in standalone.xml global-modules tag.
<module name="com.sap" slot="main"/>
```

3. Restart JBoss server.

20.3 Migrating Apps to Kony Fabric

The following sections help you migrate apps with legacy services into Kony Fabric.

- Migrate an app from an earlier version of Kony Studio (6.5 or lower versions) to Kony Fabric
- Migrate a Consolidated Service Definition (CSD) to Kony Fabric

20.4 Migrate a Consolidated Service Definition to Kony Fabric

You can import apps into Kony Visualizer that are originally created manually or with external tools. Such apps are called as **Consolidated Services Definition (CSD)** apps. A CSD app can contain a combination of non-legacy services (for example, XML, JSON, SOAP, and Java) and legacy services (SAP JCo connector or Scraper connector).

A CSD app comprises multiple artifacts such as service definition, sync configuration, and jars. You cannot directly import a CSD app directly into Visualizer as the folder structure of a CSD app is different from the folder structure of an IDE app. To migrate a CSD app to Visualizer, copy all files based on the defined format within the existing app folder. The following sections provide more details.

20.4.1 Prerequisites

A CSD app created manually or through external tools must contain the required files and folders that are listed in the following section.

20.4.2 CSD App - Files and Folders

A CSD app contains the following folders and files.

- **dsl** - The dsl folder contains one `.properties` file and `.dsl` files. The dsl folder is only for scraper services.
  The name of the properties file should match the app name.
- **files** - The files folder contains .properties for SAP JCo/Siebel.
- **lib** - The lib folder contains jars required by an app. The lib folder is only for scraper services.
- **wsdl** - The wsdl folder contains the following additional files for SOAP services:
  - **mapping.json**: contains mapping file for wsdl source to operation names using that wsdl source (URL/file).

```
 // Sample entry in the mapping file in JSON format:

{
    "partner.wsdl": [
        "convertLead",
        "create"
    ],
    "http://wsf.cdyne.com/WeatherWS/Weather.asmx?wsdl": [
        "GetCityForecastByZIP",
        "GetCityWeatherByZIPtest"
    ]
}
```

  - **.wsdl**: contains wsdl files in the mapping entries mentioned above. The mapping entries refer to json entries present in `mapping.json` file.
- **servicedef.xml**: Servicedef.xml includes all services in the app.
- **syncconfig.xml**: The syncconfig.xml file contains sync definition.

20.4.3 CSD App - Folder Structure

When an app that contains SAP JCo services is migrated through Visualizer, the file name mentioned as a value for the `sapserverfile` config-param must be copied to the `files` folder.

> ***Important:*** The app name and the app_ID in the service definition must match.

The following is a sample folder structure of a CSD app.

```
\---ServicesApp
    |   servicedef.xml
    |
    +---dsl
    |       exp.dsl
    |       ServicesApp.properties
    |       test9314.dsl
    |       yahooFinance.dsl
    |
    +---files
    |       sap_KONYAWSCRM.properties
    |       sap_Non_SSO.properties
    |       sap_Non_SSO_AutoCommit.properties
    |       sap_Non_SSO_SkyTech.properties
    |       sap_Non_SSO_SkyTech2.properties
    |       sap_SapServer.properties
    |       sap_SAP_SSO.properties
    |
    +---lib
    |       exelonServices.jar
    |
    \---wsdl
            mapping.json
            partner.wsdl
```

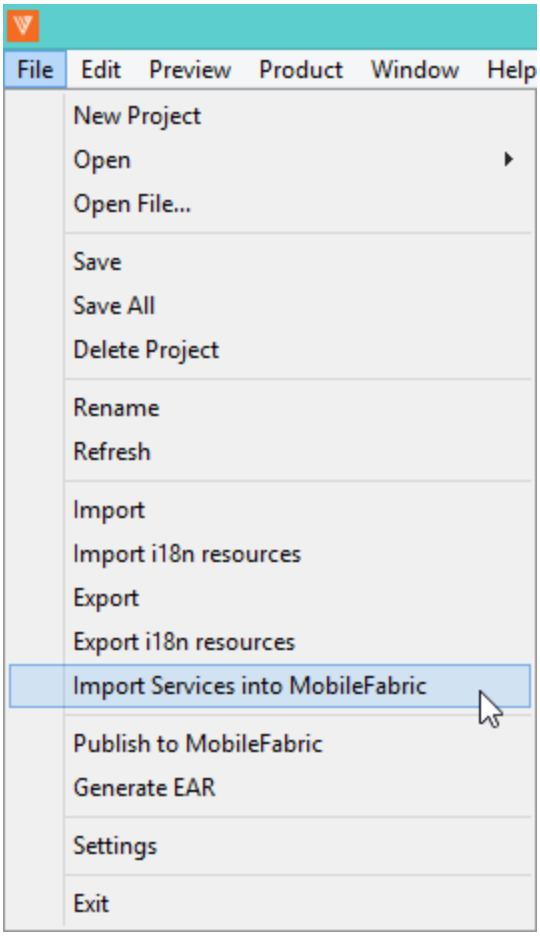
20.4.4 Migrating and Importing a CSD App

To migrate CSD app created to Visualizer 7.x, follow these steps:

1. Copy the folders and files from a CSD app. Paste them to your root of the workspace folder that you created for Visualizer under the app name folder.

   For example: `C:\Users\kh1423\Workspace_Folder\Sample_App_Folder\`.

2. In the Visualizer, go to **File** > **Import Services into Kony Fabric**.



   Visualizer imports your app into Kony Fabric.

You can launch Kony Fabric to view newly imported services in the app.

> ***Important:*** Apps with the same name are not allowed while importing to Kony Fabric.

21. Integration

21.1 Overview

An Integration Service is an application component that represents the application interaction with an external system or data source. A service definition comprises the meta-data or the configurations required to exchange data with the external system or data source. For example, the configurations can be service type, Endpoint URL, service ID, type (HTTP/HTTPS), request parameters, response parameters, preprocessors and postprocessors, and authentication credentials if required.

For more hands-on approach on how to implement Integration Services, import and preview the **News and Weather** app using Kony Visualizer.

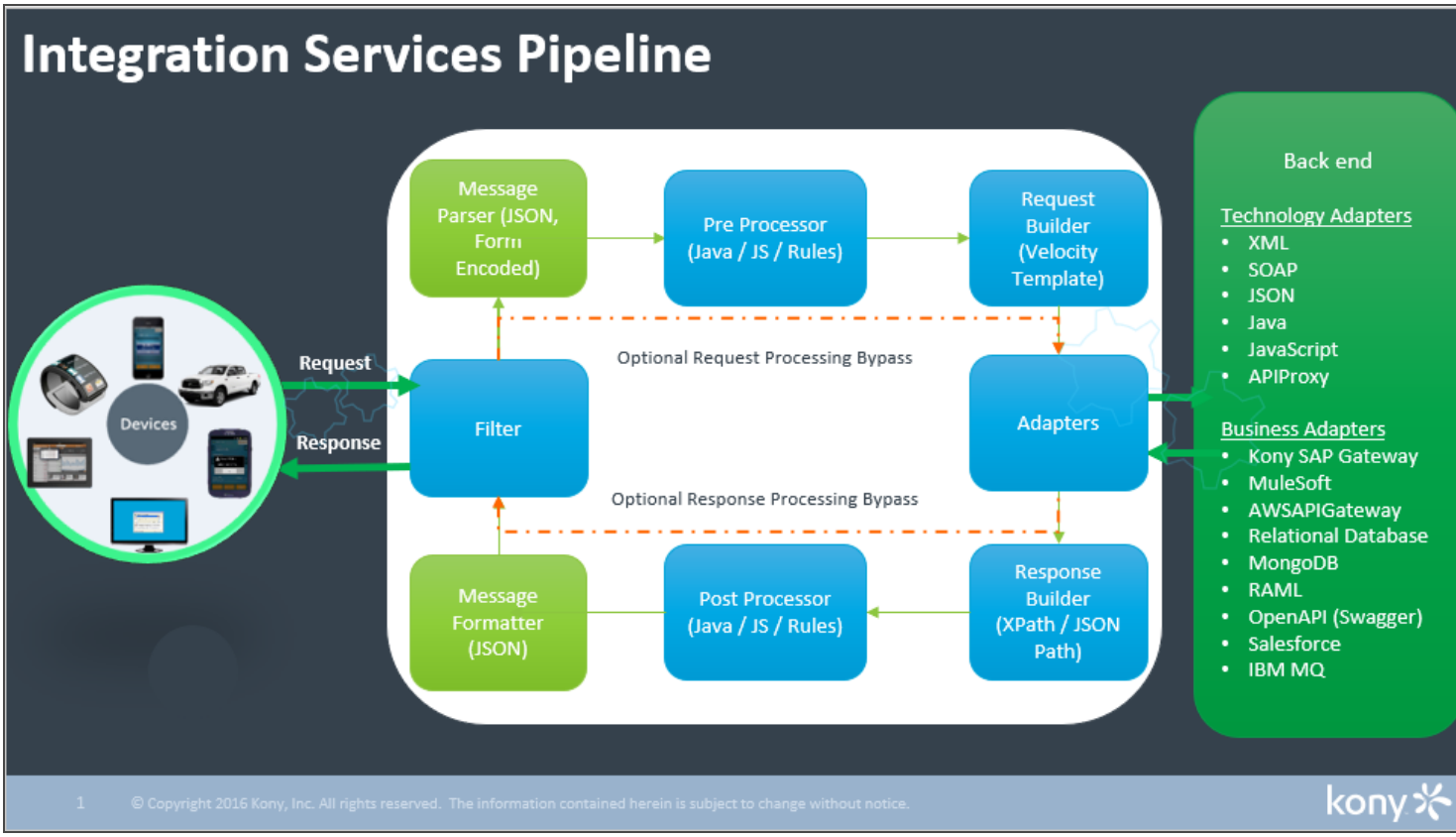
DOWNLOAD THE APP

21.2 Use Cases

If you want to develop an app which enables users to browse through news headlines with different categories such as *Top Stories*, *World*, *Science*, *Technology*, *Sports*, weather conditions and so on. In this scenario, you need to use Integration services of Kony Fabric to fetch data from 3rd party news and weather APIs, which populates the same data in the front end of the app.

21.3 Workflow of Integration Services

The following workflow describes the various stages of Integration services:


Integration Services Pipeline
Devices
Request
Response
Message Parser (JSON, Form Encoded)
Pre Processor (Java / JS / Rules)
Request Builder (Velocity Template)
Optional Request Processing Bypass
Filter
Adapters
Optional Response Processing Bypass
Message Formatter (JSON)
Post Processor (Java / JS / Rules)
Response Builder (XPath / JSON Path)
Back end
Technology Adapters
XML
SOAP
JSON
Java
JavaScript
APIProxy
Business Adapters
Kony SAP Gateway
MuleSoft
AWSAPIGateway
Relational Database
MongoDB
RAML
OpenAPI (Swagger)
Salesforce
IBM MQ
1
© Copyright 2016 Kony, Inc. All rights reserved. The information contained herein is subject to change without notice.
kony

Service Request flow

a. **Request**
A client sends a **request** to Kony Fabric.

b. **Filter**

c. **Message Parser (JSON, Form Encoded)**

d. **Pre-Processor**
After Kony Fabric receives the call from the device, Kony Fabric executes custom Java/JS code based on the configured **Pre-Processor** before making any outbound calls.
The custom Java/JS code is typically used to decide on what service to call. The custom code can also perform data validation on the request input.

e. **Request Builder (Velocity Template)**
For adapters (such as XML, JSON), user can format the request sent to an enterprise backend by defining a velocity template.

f. **Adapters**
An adapter is the component that communicates to the backend. it takes the formatted request from the request builder and sends it to the enterprise backend. In addition to the out of the box adapters, we can import adapters from Kony Marketplace.

> ***Note:*** You can select the **Enable pass-through** check box in the **Request Input** tab for Kony Fabric to forward the headers and body of clients request to the back end as is.

Service Response flow

Service Request flow

a. **Adapters**
An adapter receives the response from the enterprise backend for further processing in Kony Fabric.

b. **Response Builder**
Kony Fabric extracts data from the backend payload using **XPath** or **JSON Path** in the form of response parameters.

c. **Post-Processor**
After extracting the data, Fabric executes the custom Java/JS code based on the configured **Post-Processor**.
The custom Java/JS code is typically used to process the data before returning the data to the client. The custom code can also determine whether Fabric needs to make additional service calls.

d. **Message Formatter (JSON)**

e. **Filter**

f. **Response**
Kony Fabric sends the transformed **response** (in JSON) to the client.

> ***Note:*** You can select the **Enable pass-through** check box in the **Response Output** tab for Kony Fabric to forward the response from the back end as is to clients.

21.4 Supported Endpoint Adapters

Kony Integration supports back-end connectivity to Web services such as XML, JSON, JavaScript, Database, Salesforce and so on. If external data sources do not expose the services to these well-known interfaces, you can build a Java service with a custom code.

Out of the box Kony Fabric Integration Services support connectivity to the following different endpoint adapters.

Endpoint Type	Description	Endpoint Configuration
XML	An XML Adapter communicates with an external data source using an XML endpoint over the HTTP protocol.	• Configure XML Endpoint Adapter • Create Operations for XML ◦ Configure Request Operation for XML ◦ Configure Response Operation for XML
SOAP	Simple Object Access Protocol (SOAP) is a messaging protocol that uses WSDL to describe the functionality of a SOAP based web service.	• Configure SOAP Endpoint Adapter • Create Operations for SOAP ◦ Configure Request Operation for SOAP ◦ Configure Response Operation for SOAP
JSON	JavaScript Object Notation (JSON) communicates with an external data source over the HTTP protocol, and returns a response in JSON format.	• Configure JSON Endpoint Adapter • Create Operations for JSON ◦ Configure Request Operation for JSON ◦ Configure Response Operation for JSON

Endpoint Type	Description	Endpoint Configuration

Java	Java Adapter interacts with the software application that does not support restful APIs using a custom Java code.	• Configure Java Endpoint Adapter
		• Create Operations for Java
		  ◦ Configure Request Operation for Java
		  ◦ Configure Response Operation for Java

JavaScript	JavaScript integrates plain JavaScript services to applications in Kony Fabric.	• Configure JavaScript Endpoint Adapter
		• Create Operations for JavaScript
		  ◦ Configure Request Operation for JavaScript
		  ◦ Configure Response Operation for JavaScript

API Proxy	API Proxy forwards the request and response without intermediate transformation (without affecting the actual request and response).	• Configure API Proxy Endpoint Adapter • Create Operations for API Proxy

Mock Data	The Kony Mock Data adapter capability helps you to continue to develop apps when the back-end services that an app connects to are not ready to be leveraged.	• Configure Mock Data Endpoint Adapter
		• Create Operations for Mock Data
		  ◦ Configure Request Operation for Mock Data
		  ◦ Configure Response Operation for Mock Data

Endpoint Type	Description	Endpoint Configuration

Kony SAP Gateway	Kony SAP Gateway communicates with external SAP services over supported HTTP methods.	• Configure Kony SAP Gateway Endpoint Adapter
		• Create Operations for Kony SAP Gateway
		  ◦ Configure Request Operation for Kony SAP Gateway
		  ◦ Configure Response Operation for Kony SAP Gateway

MuleSoft	MuleSoft (Anypoint Platform™) is a platform that helps app developers to design custom APIs and deploy to a Mule Enterprise Service Bus runtime (ESB). With MuleSoft integration service in Kony Fabric, developers can interact with more than 50 types of adapters.	• Configure MuleSoft Endpoint Adapter
		• Create Operations for MuleSoft
		  ◦ Configure Request Operation for MuleSoft
		  ◦ Configure Response Operation for MuleSoft

AWS API Gateway	AWS API Gateway connects to the services configured and deployed under API Gateway Service in Amazon Web Services.	• Configure AWS API Gateway Endpoint Adapter
		• Create Operations for AWS API Gateway
		  ◦ Configure Request Operation for AWS API Gateway
		  ◦ Configure Response Operation for AWS API Gateway

Endpoint Type	Description	Endpoint Configuration

Database	With Kony Fabric database adapter, you can connect to your own database as an endpoint. After you configure the database adapter in Kony Fabric Console, you can perform create, read, update, and delete (CRUD and Binary CRUD) operations on data in the tables and invoke stored procedures, functions, and views.	• Configure Database Endpoint Adapter
		• Create CRUD Operations for Database Adapter
		• Configure CRUD Operations for a Database Adapter
		  ◦ Create a Database Record with Create Operation
		  ◦ Query a Database and Display Information with Read Operation
		  ◦ Update a Database Record with Update Operation
		  ◦ Delete a Database Record with Delete Operation

MongoDB	With Kony Fabric MongoDB database adapter, you can connect to your own MongoDB database as an endpoint. After you configure the MongoDB adapter, you can perform create, read, update, and delete (CRUD) operations on MongoDB collections and documents.	• Configure MongoDB Adapter • Create and Configure CRUD Operations for a MongoDB Adapter

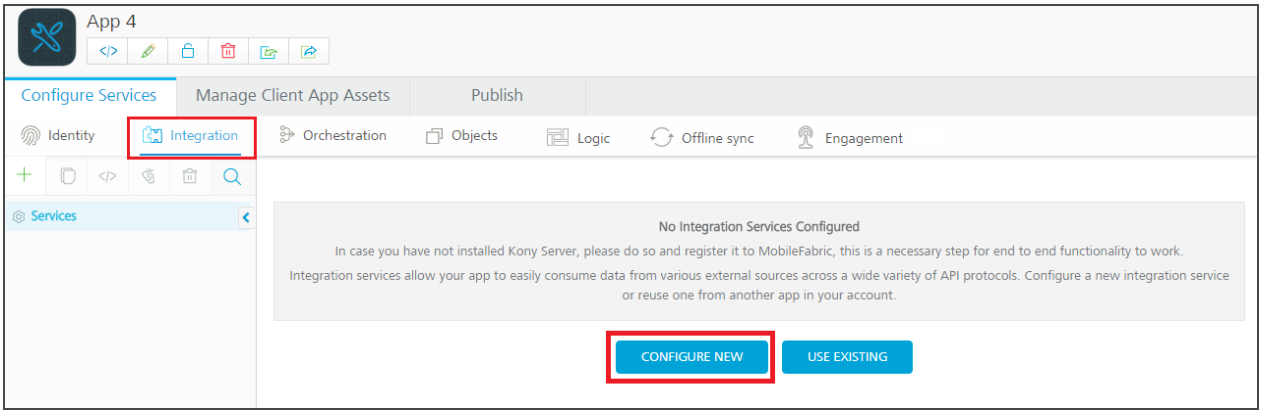
Endpoint Type	Description	Endpoint Configuration
RAML	RESTful API Modeling Language (RAML) is a YAML-based language for describing RESTful APIs.	• Configure RAML Endpoint Adapter • Create Operations for RAML ◦ Configure Request Operation for RAML ◦ Configure Response Operation for RAML
Salesforce	Salesforce uses a protocol-specific adapter to connect to an external system and access its data.	• Configure Salesforce Endpoint Adapter • Create Operations for Salesforce ◦ Configure Request Operation for Salesforce ◦ Configure Response Operation for Salesforce
IBM MQ	IBM MQ server 9.0.0 version is a messaging middleware that simplifies and accelerates the integration of diverse applications and business data across multiple platforms. Kony uses the IBM MQ service to secure the message delivery and reduce the risk of data loss.	• Configure IBM MQ Endpoint Adapter • Create Operations for IBM MQ ◦ Configure Request Operation for IBM MQ ◦ Configure Response Operation for IBM MQ

Endpoint Type	Description	Endpoint Configuration
SAP JCo	Kony Fabric allows you to access and use the external SAP services using custom SAP Connector. You must load the required Business Application Programming Interface (BAPI) functions to define an SAP service. The BAPI files contain the SAP methods and functions.	• Configure SAP JCo Endpoint Adapter • Create Operations for SAP JCo ◦ Configure Request Operation for SAP JCo ◦ Configure Response Operation for SAP JCo
Open API (Swagger)	Swagger is a powerful open API specification framework backed by a large ecosystem of tools that helps you design, build, document, and consume your RESTful APIs.	• Configure Open API (Swagger) Endpoint Adapter • Create Operations for Open API (Swagger) ◦ Configure Request Operation for Open API (Swagger) ◦ Configure Response Operation for Open API (Swagger)

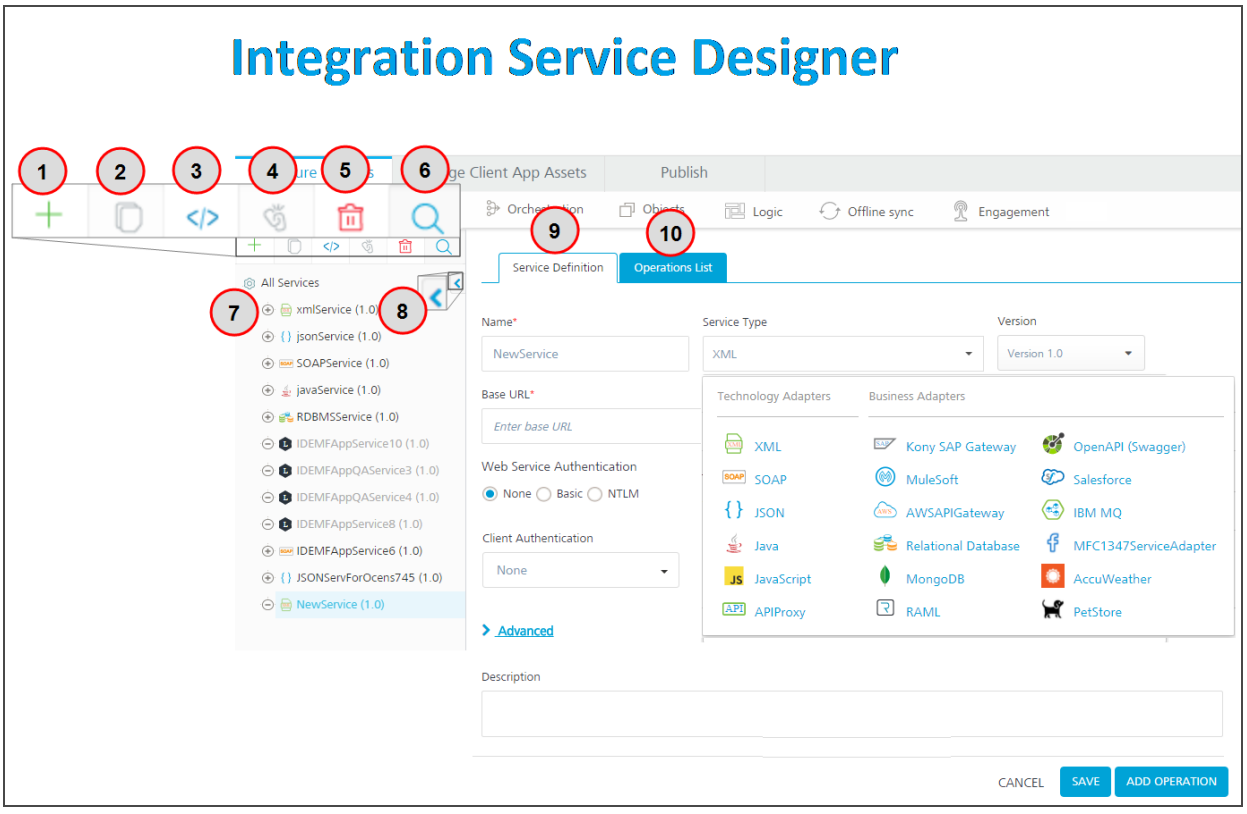
21.5 Configure Integration Service

Menu path for integration service designer:

1. After you create an application, in the **Configure Services** tab, click the **Integration** service tab. The **Integration** page appears and lists the existing integration services (if any).



2. Click **CONFIGURE NEW** to create an integration service. The following details are displayed in the integration service designer.



The Integration page displays the following sections for an endpoint adapter:

Number	Section	Description
1	**Add**	Allows you to add the following: • Add New Service • Use Existing • Add New Operation
2	**Clone**	Allows you to duplicate an existing service. Changes made to a cloned service will not impact the original service.
3	**Sample Code**	A dynamic code is generated based on the configuration of a service. You can use this code in your SDK.
4	**Unlink**	Allows you remove the service from the Integration services section of an app. When a service is unlinked, it is disassociated from a particular app.
5	**Delete button**	Allows you to delete a service.
6	**Search button**	Allows you to search services and operations in the **Services** section.

Number	Section	Description
7	**Services section**	Configured services for an app. You can click the expand or collapse button of a service to show or hide operations in the services.
8	**Show / Hide button**	Allows you the show or hide the tree. When you hide the tree section, the right pane is used for service definition or operations details.
9	**Service Definition tab**	Allows you to configure service defection for an integration service.
10	**Operations List tab**	Allows you to configure operations for an integration service. The **Operations List** tab appears only after you save the details in the service definition tab.

3. Select the Integration service from the **Service Type** list, and configure the service.

   For more information, refer supported Integration Endpoint Adapters.

> ***Note:*** For more information on existing integration services, refer Manage Existing Integration Services.

21.5.1 XML Adapter

An XML Adapter communicates with an external data source using an XML endpoint over the HTTP protocol.

21.5.1.1 Configure XML End-point Adapter

To configure a XML service in Integration service definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **XML**.

   > ***Note:*** XML is selected, by default.

3. Provide the following details to create a XML service.

Fields	Description
Version	Specify the version number for the service.
Base URL	Type the URL.

Fields	Description
Web Service Authentication	Select one of the following modes: • **None**: Select this option if you do not want to provide any authentication for the service. • **Basic**: Provide User ID and Password if the external Web service requires a form or basic authentication. • **NTLM**: Your service follows the NT LAN Manager authentication process. You are required to provide the User ID, Password, NTLM Host, and NTLM Domain.
Identity Service for Backend Token	Select the Identity service associated with your app if this service needs backend token like access_ token from that Identity service to access the backend server.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Specify JAR	To specify a JAR associated to the service, select one from the **Select Existing JAR** list or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. You can download the uploaded jars to your local system.

Field	Description
API Throttling	• If you want to use **API throttling** in Kony Fabric Console to limit the number of request calls within a minute, do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer Override API Throttling Configuration.
URL Provider Class	Enter the qualified name of the URL Provider Class. For more information, refer URL Provider Support for XML, JSON, SOAP, and API Proxy.

> ***Note:*** All options in the Advanced section are optional.

5. Enter the **Description** for the service.
6. Click **SAVE** to save your service definition.

21.5.1.2 Create Operations for XML

The **Operations List** tab appears only after the service definition is saved.

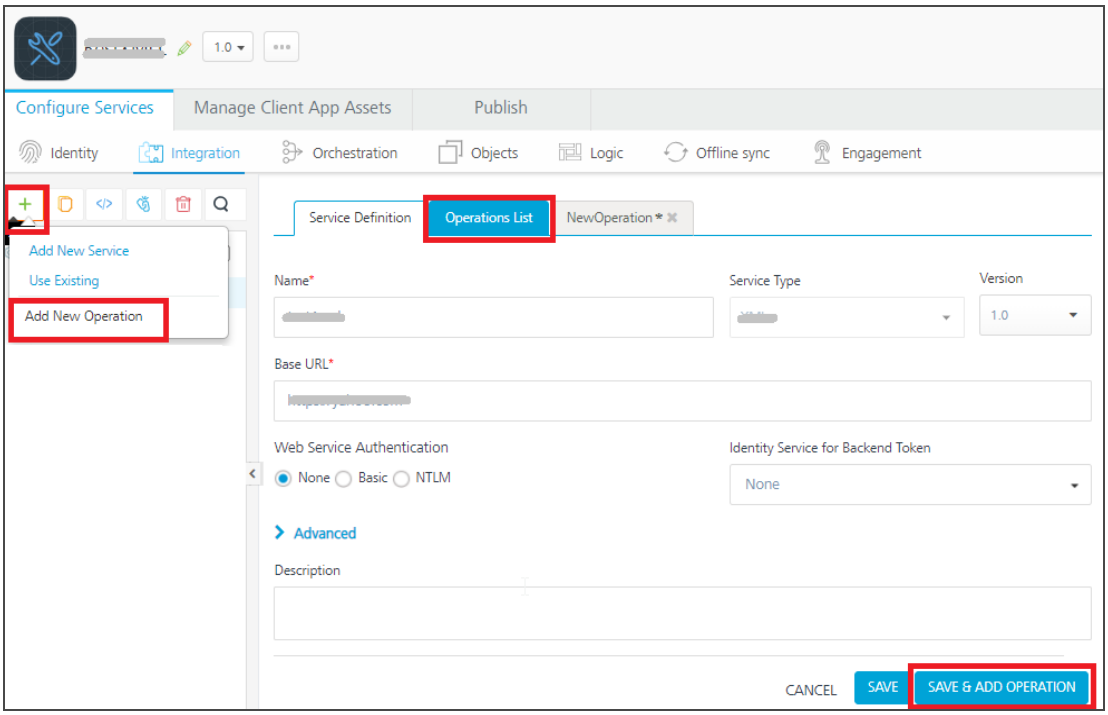
Note: Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



Note: To use an existing integration service, refer to How to Use an Existing Integration Service.

2. Type the following fields to create a new operation:

Field	Description
Name	Type a new name for the operation in the **Operation Name** box. ***Important:*** While configuring an integration service with basic auth mode, ensure that some reserved IDs are not used as input (or) header IDs. Key words such as userID and password are reserved by middleware when a user selects basic auth mode.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default.

Field	Description
Target URL	You can select which HTTP method to invoke on the back-end service from integration server. The **Target URL** field is pre-populated with the URL. You can add the suffix, if required. `http://baseurl.com/suffix` For Example, to the base URL, you can add suffix such as `/latest` or `/sports` to get latest news or sports news: • `http://feeds.foxnews.com/foxnews/latest` • `http://feeds.foxnews.com/foxnews/sports`
Target HTTP Method	You can select which HTTP method to invoke on the back-end service from integration server. Select the required method for the operation from the **Target HTTP Method** field.

3. For additional configuration of request (or) response operations, provide the following details in the **Advanced** section.

Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.

Stub Backend Response	Stub Backend Response allows you enable a stub back-end service. To enable Stub Backend Response, refer How to Enable Stub Back-end Response. For more details on Stub back-end response, refer to How to Develop Apps based on a Stubbed Service.
Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Pass-through Cookies	Pass-through Cookies allows you send cookies present in the incoming client request to the backend target request. For detailed information, refer Pass-through Cookies.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the **Advanced** section for operations are optional.

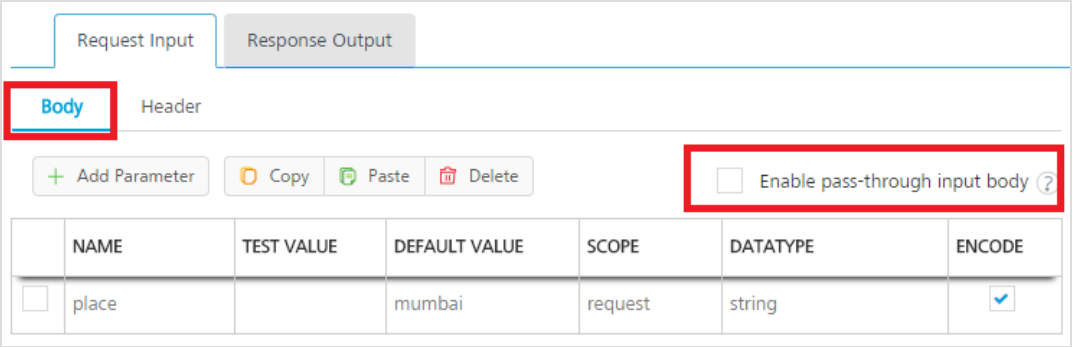
4. Enter the **Description** for the operation.

21.5.1.3 Configure Request Operation for XML

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

You can perform the following actions in **Request Input** tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .
4. Under the **Body** tab, perform the following actions:
   a. To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to [How to Enable Pass-through Proxy for Operations](#).



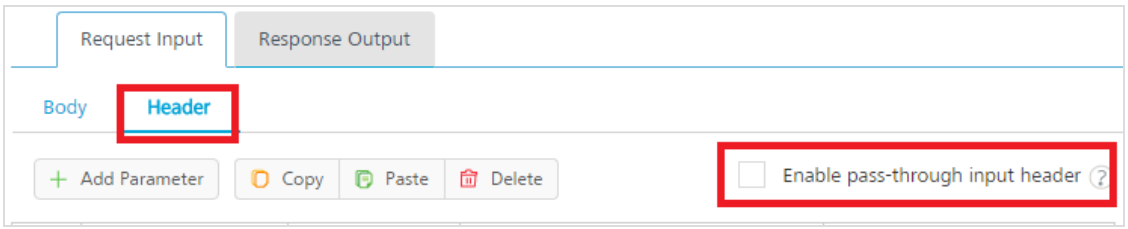
   b. To configure parameters in the clients body, do the following:

Field	Description
Name	Enter the name for the request input parameter.

Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.
TEST VALUE	Enter a value. A test value is used for testing the service.

Field	Description
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select request or session. This field is set to **Request**, by default.
DATA TYPE	The default data type for the selected column is loaded under the DATA TYPE field.
Encode	Select the checkbox to enable an input parameter to be encoded. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere the HTML URL encoding standards.
Description	Enter the description for the Request Input parameter.

5. Click the **Header** tab to provide the following custom headers for an operation.



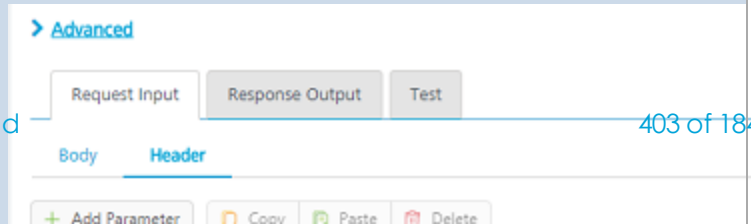
You must provide the custom HTTP headers based on the operation. For example, post or get.

Perform the following actions to provide the custom header

a. To forward headers of the client's request to backend as it is, select the **Enable pass-through input header** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

b. To configure parameters in the client's header, do the following:

Field	Description
Name	Provide custom HTTP headers required by the external source.

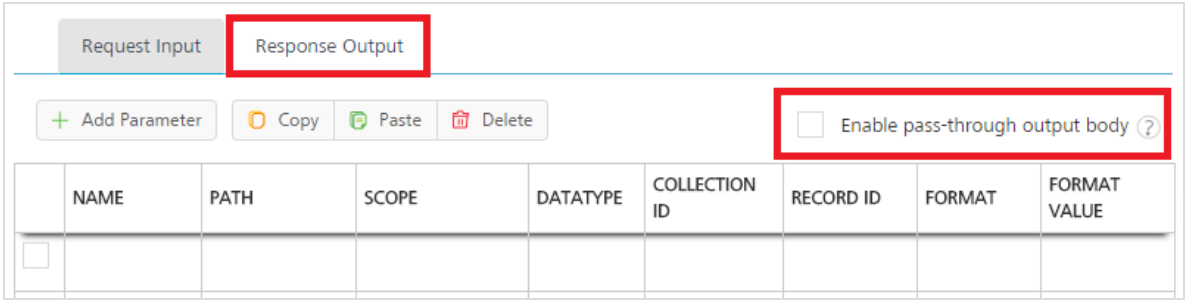
Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select Request or Session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default. ***Note:*** If the header value is scoped as a **Request** (or) **Session** and the same header is accessed under the **Expression** header value, then the expression must be represented as $request.header (or) $session.header. **Example**: If a header 1 value is a request and header 2 value is an expression, then the value of the expression must be $Request.header1. 

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Change the syntax, if required. The default value will be used if the test value is empty.
Description	Enter the description for the header parameter.

6. You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages are displayed in the service execution for better debugging. You can refer to Test a Service Operation for the steps to test a service.

21.5.1.4 Configure Response Operation for XML

Click **Response Output** tab to configure the fields of the table for displaying the data.



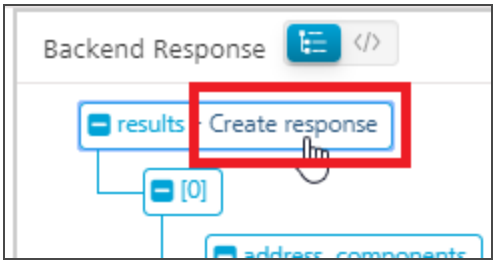
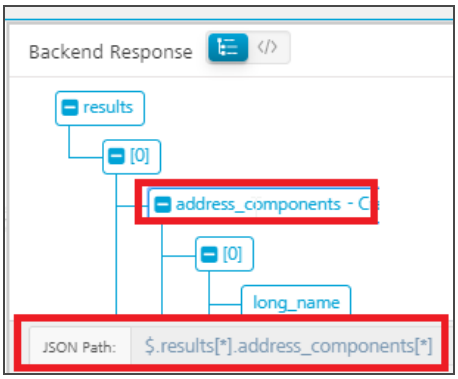
Note: If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

1. To forward the response from the backend to the client as is, select the **Enable pass-through output body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

2. You can configure XPath expressions for extracting the required elements from the back-end response of the service call. So that the extracted output can be sent to the client app. You can create an XPath **automatically** or **manually**.

   > ***Note:*** Auto generation of XPath support is available from Kony Fabric V8 SP3 onwards.

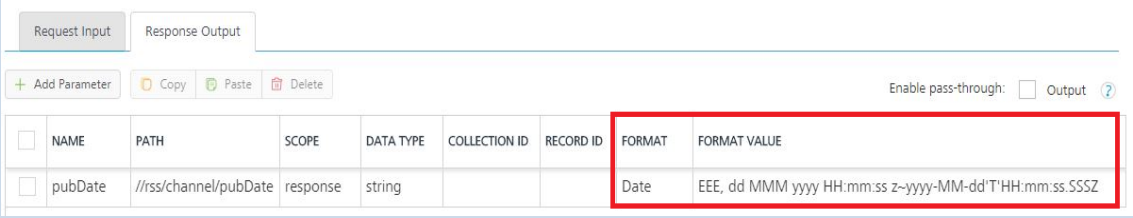
   The following table details XPath generation:

To create XPath **automatically (SP4)**	To create XPath **manually**
1. After you click **Save and Fetch Response**, the **Tree** view with the back-end response appears by default in the **Test > Backend Response** pane. 2. Click or hover your mouse cursor over the node for which you want to create XPath. The **Create response** button appears next to that node. 3. Click the **Create response** button.  A new row is created automatically along with the XPath for the selected node in the Response Output tab. ***Note:*** The **Response Output** tab appears if you have clicked the **Create response** button from Request Input tab.	1. Click **Add Parameter** to add new row. 2. Click the **Tree** button in the **Backend Response** tab. This displays the backend response in a tree structure format. 3. Click the node for which you want to create XPath. The XPath for that node is displayed at the bottom of the Tree structure.  4. Enter that XPath in the row that you have created.

3. To configure parameters in the response, enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format and format value.

Important: If the back-end for an XML service provides the date in a specific format and you want send the date in a different format to a device, you can configure the data format and FormatValue (syntax : `inputDateFormat~outputDateFormat`) in the response tab.

For example, if a back-end sends the date as `Thu, 07 Sep 2017 07:03:00 GMT` and you want convert it to `2017-09-07T07:03:00.000+0000`, then set the format value as `EEE, dd MMM yyyy HH:mm:ss z~yyyy-MM-dd'T'HH:mm:ss.SSSZ`.



For more details on the syntax of the date formats, refer https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

Note: When you enable Pass-through proxy flags, you will notice that you cannot configure request input, headers, and response out parameters for this operation.

4. To validate the operation details, click **Save** and Test. For more details, refer to Test a Service Operation.
5. Click **Save Operation** to save the changes.

   To use an existing integration service, refer to How to Use an Existing Integration Service.

Note: You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

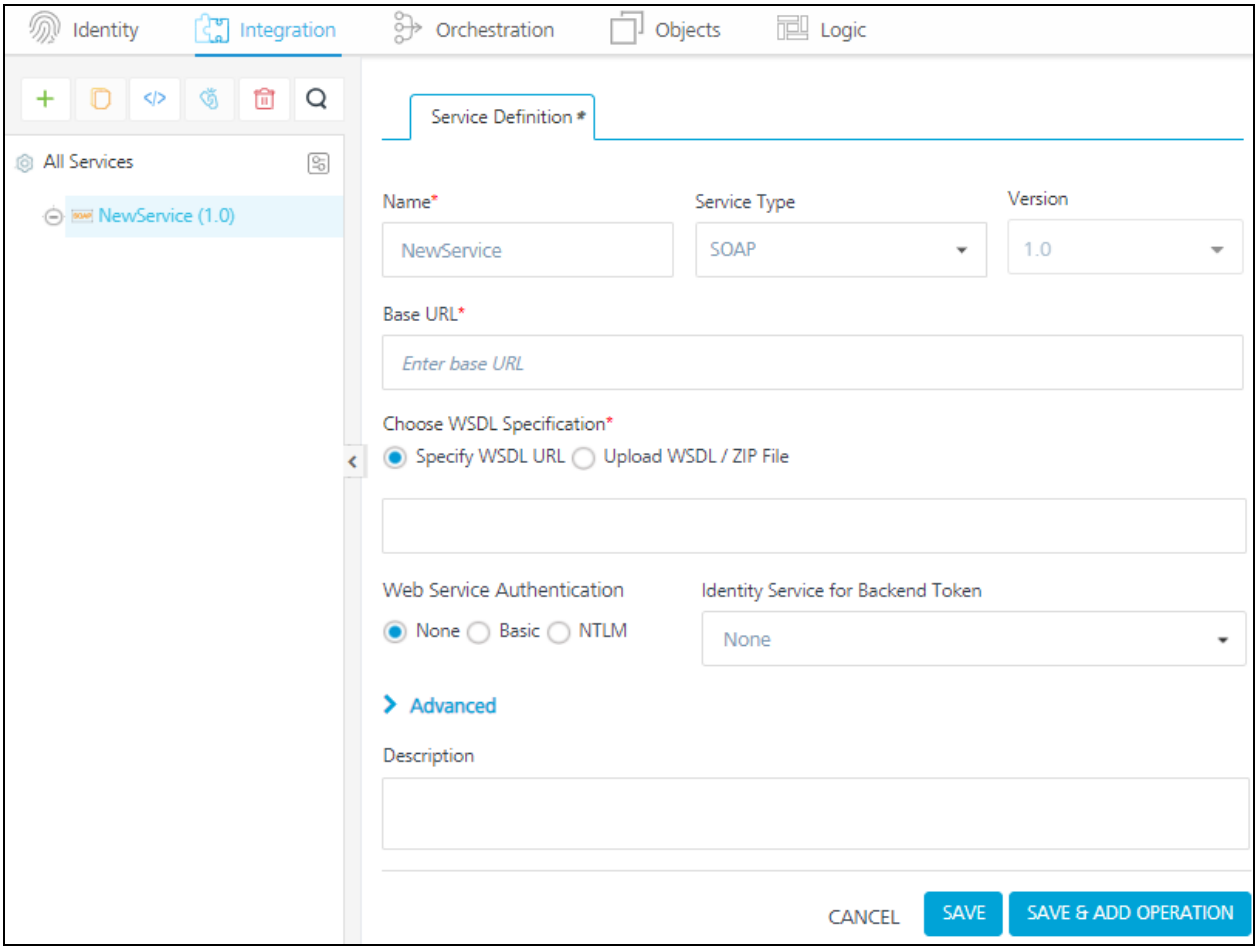
21.5.2 SOAP

To create the service definitions for an external data source providing SOAP interface, use the SOAP Service. You need to have a WSDL URL or file to create the service definition. For example, we will use a WSDL URL of a weather application that has a `GetCityWeatherByZip` service and will provide appropriate configurations for the Service Definition.

21.5.2.1 Configure SOAP End-point Adapter

To configure a SOAP service in Integration service definition tab, follow these steps:

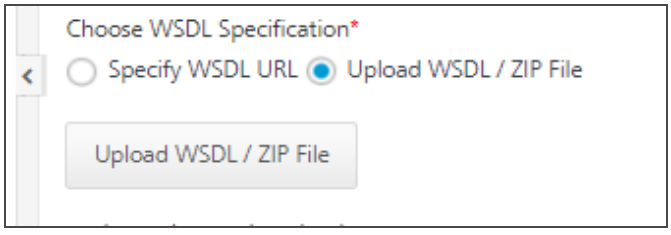
1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **SOAP**.



> ***Note:*** XML is selected, by default.

3. Provide the following details to create a SOAP service.

Fields	Description
Version	Specify the version number for the service.

Fields	Description
Base URL	Type the URL.
Choose WSDL Specification	Select the option to specify the WSDL (Web Service Definition Language) URL or upload the WSDL / ZIP file. • Click **Specify WSDL URL** and type the WSDL URL. ***Important:*** In case, if you are unable to read metadata from an integration service (for example, SOAP) that is protected by an SSL certificate, enable the integration service by following one of the ways: - Download the WSDL file from the https WSDL URL and upload the WSDL file on Kony Fabric console. - Importing the SSL into your cacerts in your Kony Fabric install location. For more details, refer to FAQs. • Click **Upload WSDL / ZIP File** and follow these steps to upload your WSDL / ZIP file. The ZIP file that refers to local XSD files. ◦ Click **Upload WSDL / ZIP File** button. ◦ Navigate to the WSDL or ZIP file from your local system and click **Open**.  The selected WSDL or ZIP file gets uploaded.

Fields	Description
Web Service Authentication	Select one of the following modes: • **None**: Select this option if you do not want to provide any authentication for the service. • **Basic**: Provide User ID and Password if the external Web service requires a form or basic authentication. • **NTLM**: Your service follows the NT LAN Manager authentication process. You are required to provide the User ID, Password, NTLM Host, and NTLM Domain.
Identity Service for Backend Token	Select the Identity service associated with your app if this service needs backend token like access_token from that Identity service to access the backend server.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Custom code	To specify a JAR associated to the service, select one from the **Select Existing JAR** drop-down menu or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. You can download the uploaded jars to your local system.

Field	Description
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration.
URL Provider Class	Enter the qualified name of the URL Provider Class. For more information, refer URL Provider Support for XML, JSON, SOAP, and API Proxy.

> ***Note:*** All options in the Advanced section are optional.

5. Enter the **Description** for the service.
6. Click **SAVE** to save your service definition.

21.5.2.2 Create Operations for SOAP

The **Operation List** tab appears when you click **Add Operation** in the **Service Definition** page.

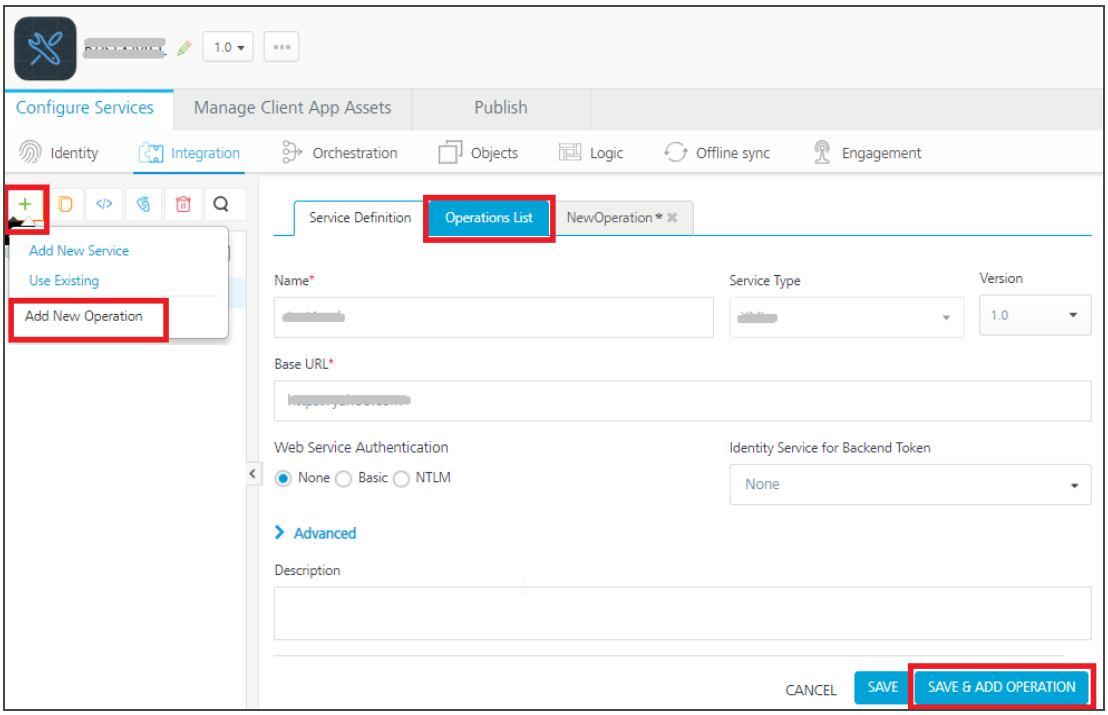
Note: Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



Note: To use an existing integration service, refer to How to Use an Existing Integration Service.

2. In **Operations List** tab, click **Operation Name** list box and select one or more **Operations**.

3. Click **Add Operation**. The selected operations appears under **Configured Operations** list.

> ***Important:*** While configuring an integration service with basic auth mode, ensure that some reserved IDs are not used as input/header IDs. Key words such as userid, pwd and password are reserved by middleware when a user selects basic auth mode.

4. Click the operation name under **Configured Operations**. The operation details page appears.
5. Type the following fields to create an operation

Field	Description
Name	The operation name appears in the Name field. You can modify the name, if required.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default.

Field	Description
Target URL	The **Target URL** field is pre-populated with the URL. You can add the suffix, if required. `http://baseurl.com/suffix` For Example, to the base URL, you can add suffix such as `/latest` or `/sports` to get latest news or sports news: • `http://feeds.foxnews.com/foxnews/latest` • `http://feeds.foxnews.com/foxnews/sports`

6. For additional configuration of request (or) response operations, provide the following details in the**Advanced** section.

Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Pass-through Cookies	**Pass-through Cookies** allows you send cookies present in the incoming client request to the backend target request. For detailed information, refer Pass-through Cookies.

Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Stub Backend Response	**Stub Backend Response** allows you enable a stub back-end service. To enable Stub Backend Response, refer How to Enable Stub Back-end Response. For more details on Stub back-end response, refer to How to Develop Apps based on a Stubbed Service.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the **Advanced** section for operations are optional.

7. Enter the **Description** for the operation.

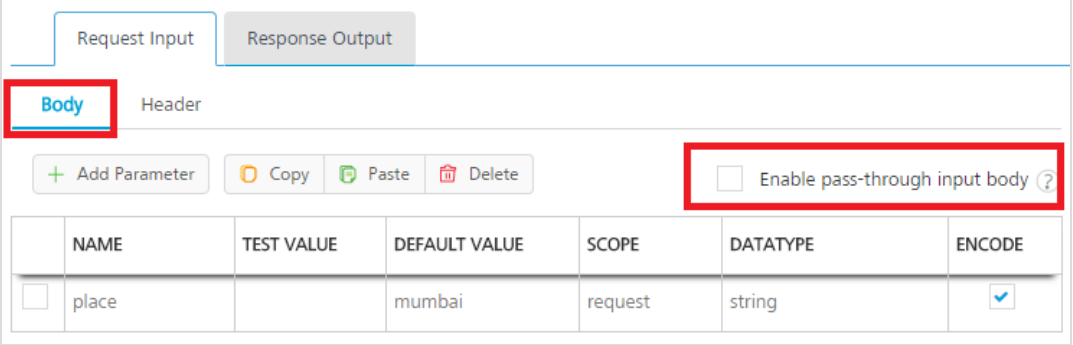
21.5.2.3 Configure Request Operation for SOAP

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

You can perform the following actions in **Request Input** tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .

4. Under the **Body** tab, perform the following actions:
    a. To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.



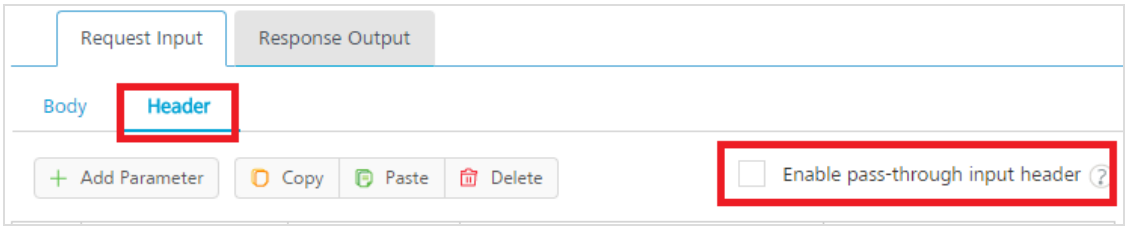
    b. To configure parameters in the clients body, do the following:

Field	Description
Name	Enter the name for the request input parameter.

Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.
TEST VALUE	Enter a value. A test value is used for testing the service.

Field	Description
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select request or session. This field is set to **Request**, by default.
Encode	Select the checkbox to enable an input parameter to be encoded. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere the HTML URL encoding standards.
Description	Enter the description for the request.

5. Click the **Header** tab to provide the following customer headers, perform the following actions:



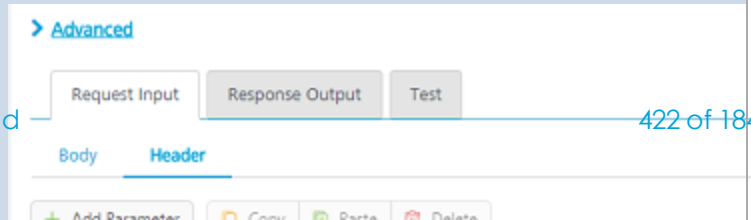
You must provide the custom HTTP headers based on the operation. For example, post or get.

a. To forward headers of the client's request to backend as it is, select the **Enable pass-through input header** check box. For more details on API Proxy service, refer to How to

Enable Pass-through Proxy for Operations.

b. To configure parameters in the client's header, do the following:

Field	Description
Name	Provide custom HTTP headers required by the external source.

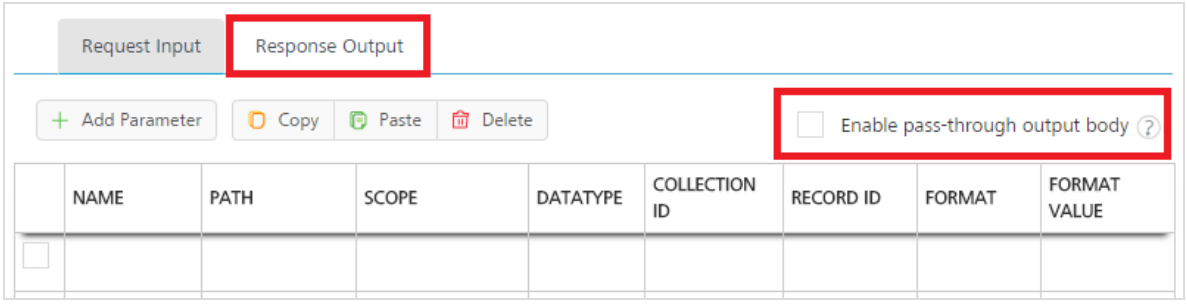
Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default. ***Note:*** If the header value is scoped as a **Request** (or) **Session** and the same header is accessed under the **Expression** header value, then the expression must be represented as $request.header (or) $session.header. **Example**: If a header 1 value is a request and header 2 value is an expression, then the value of the expression must be $Request.header1. 

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Change the syntax, if required. The default value will be used if the test value is empty.
Description	Enter the description for the header.

6. You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages are displayed in the service execution for better debugging. You can refer to Test a Service Operation for the steps to test a service.

21.5.2.4 Configure Response Operation for SOAP

Click the **Response Output** tab to configure the fields of the table for displaying the data.



Note: If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

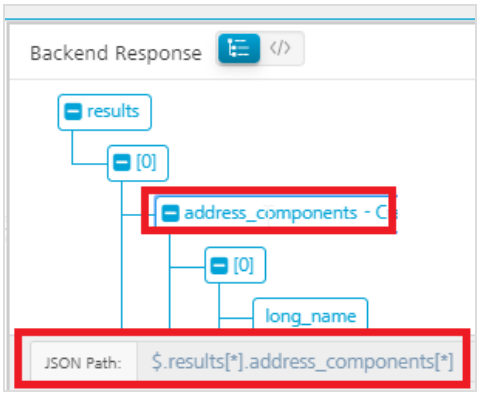
1. To forward the response from the backend to the client as it is, select the **Enable pass-through output body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.
2. You can create a desired output based on the back-end response by using XPath. So that the extracted output can be sent to the client app. You can create an XPath **manually**.

   The following steps details XPath generation manually:

   1. Click **Add Parameter** to add new row.
   2. Click the **Tree** button in the **Backend Response** tab.

      This displays the backend response in a tree structure format.
   3. Click the node for which you want to create XPath.

      The XPath for that node is displayed at the bottom of the Tree structure.

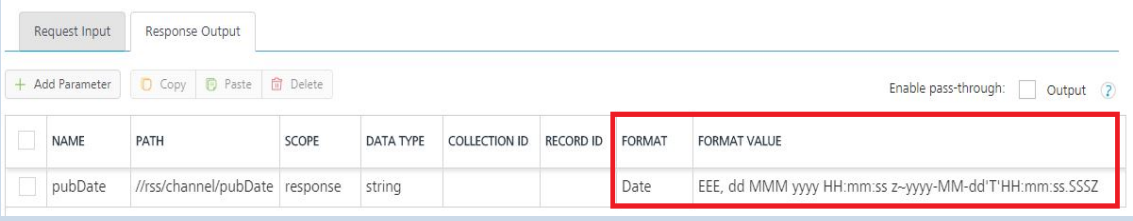
   4. Enter that XPath in the row that you have created.

      For more details, refer XPath in Kony Fabric.
3. To configure parameters in the response, enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format and format value.

ID	Path
State	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /State
City	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /City
Temperature	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /
Humidity	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /RelativeHumidity
Wind	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /Wind

Important: If the back-end for a SOAP service provides the date in a specific format and you want send the date in a different format to a device, you can configure the data format and FormatValue (syntax : `inputDateFormat~outputDateFormat`) in the response tab.

For example, if a back-end sends the date as `Thu, 07 Sep 2017 07:03:00 GMT` and you want convert it to `2017-09-07T07:03:00.000+0000`, then set the format value as `EEE, dd MMM yyyy HH:mm:ss z~yyyy-MM-dd'T'HH:mm:ss.SSSZ`.



For more details on the syntax of the date formats, refer https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

> ***Note:*** When you enable Pass-through proxy flags, you will notice that you cannot configure request input, headers, and response out parameters for this operation.

7. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.
8. Click **SAVE OPERATION** to save the changes. To use an existing integration service, refer to How to Use an Existing Integration Service.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.3 JSON Adapter

A service that communicates with an external data source using JSON over the HTTP protocol, and returns a response in JSON format is known as a JSON Service. You can use the JSON services in any case where you would use an XML service. But, the response of a JSON service is in a JSON format.

Concepts about JSON Adapter

Notations

- JSON Object - {}
- JSON Array - []

Important Considerations

- JSON Array will consist of an array of JSON Objects or a blank array.
- JSON Object is a key value pair. The key is a String and value can be a String, number(int, float,double), JSON Object or JSON Array.

- JSON string will not contain attributes.
- JSON path does not provide Axes like Xpath.
- For JSON Path, if a JSON backend contains a collection, the **parameters set** should be same for all the records in the collection. For example, if the collection has parameters: `name` and `type`, these parameters should be repeated in the backend response, shown in the following sample code:

```
 // Sample code from a response:

{"GlossDef":{"para":"A meta-markup language, used to create
markup languages such as DocBook.","GlossSeeAlso":
[{"name":"test","type":"XML"},{"name":"tester","type":"HTML"}]}}
```

- For XPath, if a JSON backend contains a collection, the fetch `only a parameter from the record`, the XPath should be in `<record>/<collection>[*]/<parameter>`, which includes an `[*]`.

  For example, in the previous sample code, if you want to fetch `name` from the `GlossSeeAlso`, the sample XPath should be as follows:

```
 // Sample code to fetch only the name parameter from a record,
XPath:

/GlossDef/GlossSeeAlso[*]/name
```

Selecting Elements

Element	Description
elementname	Selects all child elements of the named Element.
//	Selects elements in the document from the current element that match the selection no matter where they are.

Example

Path Expression	Result
bookstore (or) /bookstore	Selects all the child elements of the bookstore element
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the JSON string
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element

Predicates

Predicates are used to find a specific element or an element that contains a specific value. Predicates are always embedded in square brackets.

Path Expression	Result
bookstore/book [0]	Selects the first book element that is the child of the bookstore element
bookstore/book [last()]	Selects the last book element that is the child of the bookstore element
bookstore/book [last()-1]	Selects the last but one book element that is the child of the bookstore element
bookstore/book [position()<3]	Selects the first two book elements that are children of the bookstore element

Path Expression	Result
bookstore/book [price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
bookstore/book [price>35.00] /title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

Note: If a key name contains a value with a dot (`.`), the path for the same should be in double quotes (`[""]`). For example, if the key in response is `<element1>.<element2>`, the path must be as `["<element1>.<element2>"]`

Operators

Operator	Description	Example	Result
>	Greater than	price>9.80	• true if price is 9.90 • false if price is 9.80
>=	Greater than or equal to	price>=9.80	• true if price is 9.90 • false if price is 9.70
<	Less than	price<9.80	• true if price is 9.00 • false if price is 9.80

Operator	Description	Example	Result
!=	Not equal	price!=9.80	• true if price is 9.90 • false if price is 9.80
=	Equal	price=9.80	• true if price is 9.80 • false if price is 9.90

21.5.3.1 Configure JSON End-point Adapter

To configure a JSON service in Integration service definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **JSON**.

> ***Note:*** XML is selected, by default.

3. Provide the following details to create a JSON service.

Fields	Description
Path Expression	You can choose between JSON Path or XPath expressions to process and transform the data from the backend response. ***Note:*** JSON Path is the recommended option for a JSON service. (Requires Kony Fabric runtime version 8 with Service Pack 3 or higher) ***Note:*** To walk-through creating automated response parameters for a JSON service by using the Tree view with Kony Fabric, take a look at our hands-on tutorial for UI to view and create a Service Response.
Version	Specify the version number for the service.
Base URL	Type the URL.

Fields	Description
Web Service Authentication	Select one of the following modes: • **None**: Select this option if you do not want to provide any authentication for the service. • **Basic**: Provide User ID and Password if the external Web service requires a form or basic authentication. • **NTLM**: Your service follows the NT LAN Manager authentication process. You are required to provide the User ID, Password, NTLM Host, and NTLM Domain.
Identity Service for Backend Token	Select the Identity service associated with your app if this service needs backend token like access_ token from that Identity service to access the backend server.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Specify JAR	To specify a JAR associated to the service, select one from the **Select Existing JAR** drop-down menu or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. You can download the uploaded jars to your local system.
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration.
URL Provider Class	Enter the qualified name of the URL Provider Class. For more information, refer URL Provider Support for XML, JSON, SOAP, and API Proxy.

> ***Note:*** All options in the Advanced section are optional.

5. Enter the **Description** for the service.
6. Click **SAVE** to save your service definition.

21.5.3.2 Create Operations for JSON

The **Operation List** tab appears when you click **Add Operation** in the **Service Definition** page.

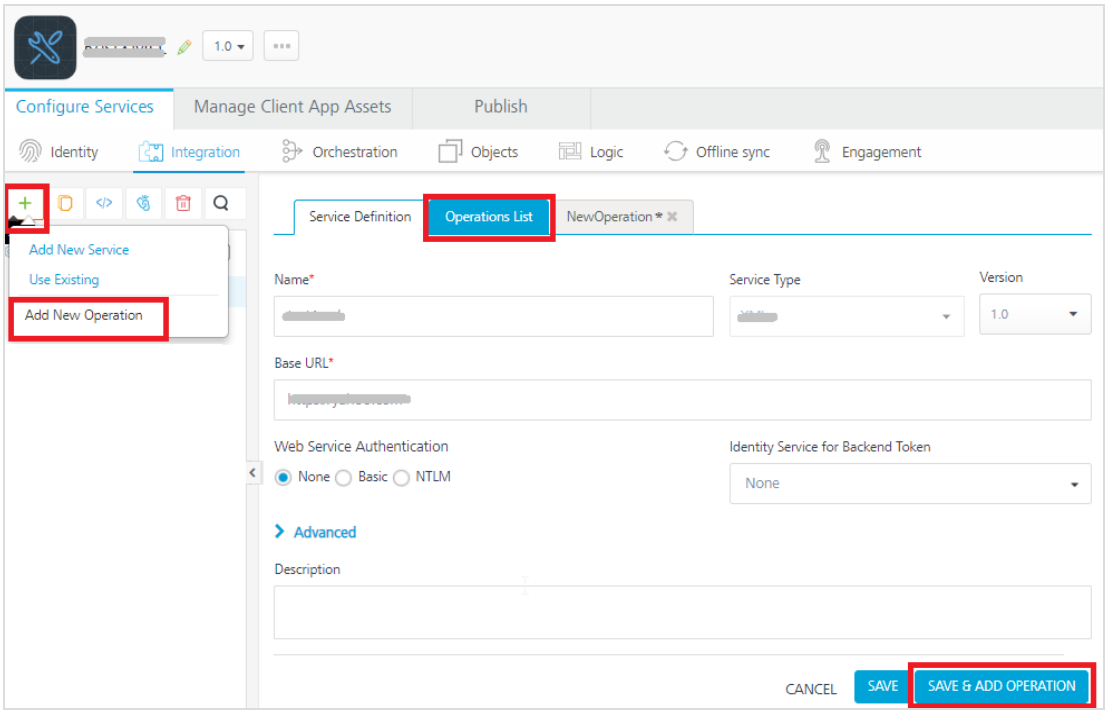
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

2. Click **Add Operation**. The selected operations appears under **Configured Operations** list.
3. Provide the following details to create an operation.

Field	Description
Name	The operation name appears in the Name field. You can modify the name, if required.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default.

Field	Description
Target URL	The **Target URL** field is pre-populated with the URL. You can add the suffix, if required. `http://baseurl.com/suffix` For Example, to the base URL, you can add suffix such as `/latest` or `/sports` to get latest news or sports news: • `http://feeds.foxnews.com/foxnews/latest` • `http://feeds.foxnews.com/foxnews/sports`

4. For additional configuration of request (or) response operations, provide the following details in the **Advanced** section.

Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Pass-through Cookies	Pass-through Cookies allows you send cookies present in the incoming client request to the backend target request. For detailed information, refer Pass-through Cookies.

Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Stub Backend Response	Stub Backend Response allows you enable a stub back-end service. To enable Stub Backend Response, refer How to Enable Stub Back-end Response. For more details on Stub back-end response, refer to How to Develop Apps based on a Stubbed Service.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the **Advanced** section for operations are optional.

5. Enter the **Description** for the operation.

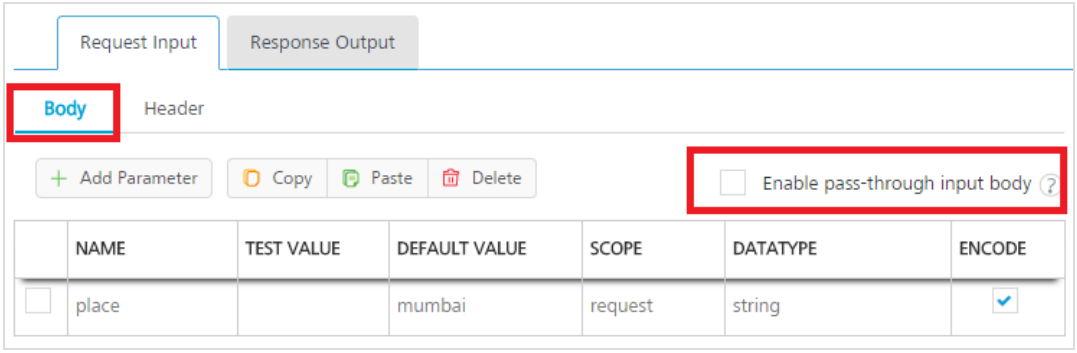
21.5.3.3 Configure Request Operation for JSON

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

You can perform the following actions in **Request Input** tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .

4. Under the **Body** tab, perform the following actions:
    a. To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.



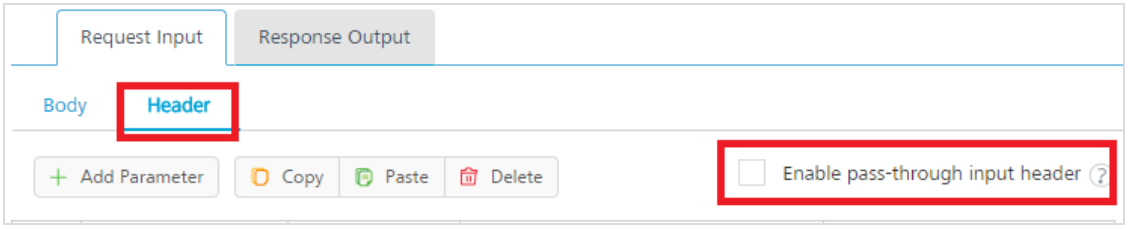
    b. To configure parameters in the clients body, do the following:

Field	Description
Name	Enter the name for the request input parameter.

Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.
TEST VALUE	Enter a value. A test value is used for testing the service.

Field	Description
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select request or session. This field is set to **Request**, by default.
Encode	Select the checkbox to enable an input parameter to be encoded. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere the HTML URL encoding standards.

5. Click the **Header** tab to provide the following custom headers for an operation.



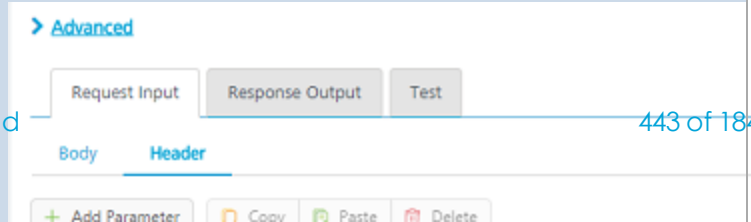
You must provide the custom HTTP headers based on the operation. For example, post or get.

Perform the following actions to provide the custom header:

a. To forward headers of the client's request to backend as it is, select the **Enable pass-through input header** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

b. To configure parameters in the client's header, do the following:

Field	Description
Name	Provide custom HTTP headers required by the external source.

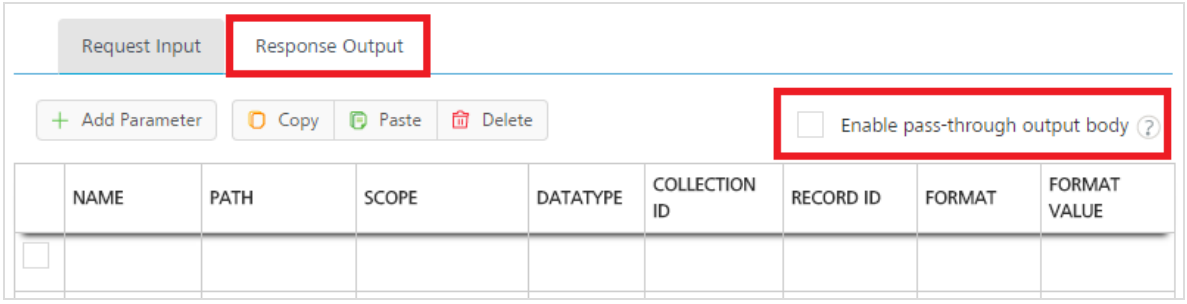
Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select Request or Session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default. ***Note:*** If the header value is scoped as a **Request** (or) **Session** and the same header is accessed under the **Expression** header value, then the expression must be represented as $request.header (or) $session.header. **Example**: If a header 1 value is a request and header 2 value is an expression, then the value of the expression must be $Request.header1. 

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Change the syntax, if required. The default value will be used if the test value is empty.
Description	Enter a proper description.

6. You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages are displayed in the service execution for better debugging. You can refer to Test a Service Operation for the steps to test a service.

21.5.3.4 Configure Response Operation for JSON

Click **Response Output** tab to configure the fields of the table for displaying the data.



Note: If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

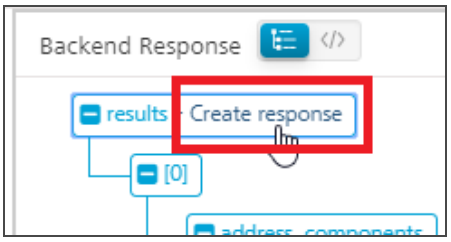
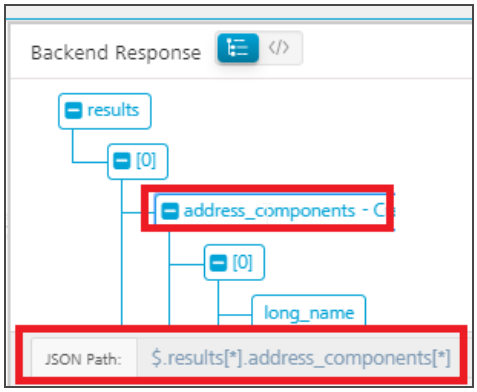
1. To forward the response from the backend to the client as it is, select the **Enable pass-through output body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

2. You can configure XPath or JSON path expressions for extracting the required elements from

the back-end response of the service call. So that the extracted output can be sent to the client app. Based on the path expression selected in the service definition, you can create an XPath or JSON path **manually**. For JSON services only, JSON Path can be auto-generated.

> ***Note:*** Auto generation of XPath support is available from Kony Fabric V8 SP3 onwards.

The following table details XPath/JSON generation:

To create JSON Path **automatically** (for V8 SP4) in case if you have selected JSON Path in the **Path Expression** field, follow these steps	To create XPath/JSON Path **manually**
1. After you click **Save and Fetch Response**, the **Tree** view with the back-end response appears by default in the **Test > Backend Response** pane. 2. Click or hover your mouse cursor over the node for which you want to create JSON Path. The **Create response** button appears next to that node. 3. Click the **Create response** button.  A new row is created automatically along with the JSON Path for the selected node in the Response Output tab. ***Note:*** The **Response Output** tab appears if you have clicked the **Create response** button from the Request Input tab. ***Important:*** If you have selected the **Path Expression** as **JSON Path**, in the **Response Output** tab, the **Add All to Response Output** cogwheel is	1. Click **Add Parameter** to add new row. 2. Click the **Tree** button in the **Backend Response** tab. This displays the backend response in a tree structure format. 3. Click the node for which you want to create XPath/JSON Path. The XPath/JSON Path for that node is displayed at the bottom of the Tree structure.  4. Enter that XPath/JSON Path in the row that you have created.

3. To configure parameters in the response, enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format and format value.

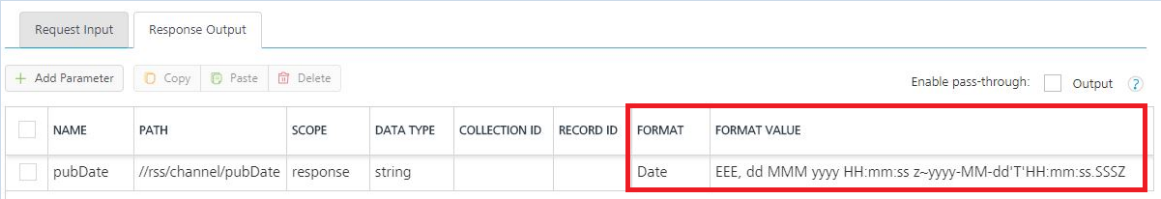
To create JSON Path **automatically** (for V8 SP4)	To create XPath/JSON Path **manually**
in case if you have selected JSON Path in the **Path Expression** field, follow these steps	Enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format and format value.

ID	Path
city	//current_observation/display_location/city
latitude	//current_observation/display_location/latitude
longitude	//current_observation/display_location/longitude
temperature	//current_observation/temp_c
relative_humidity	//current_observation/relative_humidity
windspeed	//current_observation/wind_string
icon	//current_observation/icon
icon_url	//current_observation/icon_url
forecast_url	//current_observation/forecast_url

Important: If the back-end for an XML service provides the date in a specific format and you want send the date in a different format to a device, you can configure the data format and FormatValue (syntax : `inputDateFormat~outputDateFormat`) in the response tab.

For example, if a back-end sends the date as `Thu, 07 Sep 2017 07:03:00 GMT` and

you want convert it to `2017-09-07T07:03:00.000+0000`, then set the format value as `EEE, dd MMM yyyy HH:mm:ss z~yyyy-MM-dd'T'HH:mm:ss.SSSZ`.



For more details on the syntax of the date formats, refer https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

Note: When you enable Pass-through proxy flags, you will notice that you cannot configure request input, headers, and response out parameters for this operation.

4. To validate the operation details, click **Save** and Test. For more details, refer to Test a Service Operation.
5. Click **Save Operation** to save the changes.

   To use an existing integration service, refer to How to Use an Existing Integration Service.

Note: You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.4 Java Adapter

With Java service, you can interact with your software application that does not support restful APIs. A service that uses a custom Java adapter is a Java service. The Java adapter is a custom Java class and you can create a Java adapter either by implementing `com.konylabs.middleware.common.JavaService` interface or `com.konylabs.middleware.common.JavaService2` interface. Kony recommends to use JavaService2 as you can get an access to `DataControllerRequest` and `DataControllerResponse` objects.

You must load the required JAR files to define a Java service. The JAR files contain the Java classes. The Java classes contain the Java methods. These methods have the logic defined that is required for a service. Java services are mostly used with Webconnector Services.

> ***Note:*** The `middleware-system.jar` helps you to develop a Java adapter. You can download the `middleware-system.jar` from Admin Console's download page.

> ***Note:*** You should not modify JVM timezone through custom code as modifying it can result in app server outage.

21.5.4.1 Data Conversion of a Java Adapter

The data structure of the Kony Result object `(com.konylabs.middleware.datamapping.Result)` shows that all the data is converted, but the complete data in the Kony Result object in not exposed as expected. You get a part of the data in Kony Fabric because only a few unnamed records are converted into an object instead of an array.

The use of JSON Arrays is particularly important for the mapping of data in user interface segments that require arrays as data input.

The following code details sample original JSON data, data converted to Kony Object, and in the result - data converted only limited unnamed records into an object in Kony Fabric (loss of data).

```
//Sample: Original JSON Data:

{"booking":[
{"amount":-254.6,"description":"Paiement carte BIM STORES PALMIER  le
21/02/2016 à 18:54","doc_id":17150820,"value_date":"2016-02-
19","temporary":false,"date":"2016-02-21"},{"amount":-
44.26,"description":"Paiement internet LYDEC ECOM  le 19/02/2016 à
13:04","doc_id":17146660,"value_date":"2016-02-
18","temporary":false,"date":"2016-02-19"},{"amount":-
37.26,"description":"Paiement internet LYDEC ECOM  le 19/02/2016 à
13:19","doc_id":17146835,"value_date":"2016-02-
18","temporary":false,"date":"2016-02-19"},{"amount":-
14.34,"description":"Paiement internet LYDEC ECOM  le 19/02/2016 à
13:11","doc_id":17146744,"value_date":"2016-02-
18","temporary":false,"date":"2016-02-19"}
]}


Sample: Data converted to KONY Object:
Dataset [id=booking, index=-1,
       records=[
               Record [index=-1, order=0, id=,
                       params=[
                               Param [name=amount, value=-254.6],
                               Param [name=description, value=Paiement carte BIM STORES PALM
21/02/2016 à 18:54],
                               Param [name=doc_id, value=17150820],
                               Param [name=value_date, value=2016-02-19],
                               Param [name=temporary, value=false],
                               Param [name=date, value=2016-02-21]
                       ], datasets=[], records=[]
                       ], Record [index=-1, order=0, id=,
                       params=[
                               Param [name=amount, value=-44.26],
                               Param [name=description, value=Paiement internet LYDEC ECOM
à 13:04],
```

```
                                Param [name=doc_id, value=17146660],
                                Param [name=value_date, value=2016-02-18],
                                Param [name=temporary, value=false],
                                Param [name=date, value=2016-02-19]
                            ], datasets=[], records=[]
                            ],
```

```
//Sample: Data converted only limited unnamed records into an object in Kony Fabric:
"booking": {
            "": {
                "date": "2016-02-21",
                "temporary": "false",
                "amount": "-254.6",
                "description": "Paiement carte BIM STORES PALMIER  le 21/02/2016 à 18:54",
                "doc_id": "17150820",
                "value_date": "2016-02-19"
            }
               }
```

21.5.4.2 Writing a Java Class

To write a Java class for a Java adapter, follow these steps:

1. Create a Java adapter either by implementing the `com.konylabs.middleware.common.JavaService` interface or `com.konylabs.middleware.common.JavaService2` interface.
2. When you implement `com.konylabs.middleware.common.JavaService`, you have to implement the following `invoke()` method with the signature:

```
 public Object invoke(String paramString, Object[]
paramArrayOfObject) throws Exception;
```

3. When you implement `com.konylabs.middleware.common.JavaService2`, you have to implement the following `invoke()` method with the signature:

```
 public Object invoke(String methodID, Object[] objectArray,
DataControllerRequest request, DataControllerResponse response)
throws Exception;
```

21.5.4.3 Middleware API to get Output parameters in a Java Service

A middleware API is provided to get the output params/records/datasets configured as part of a service definition of a Java service.

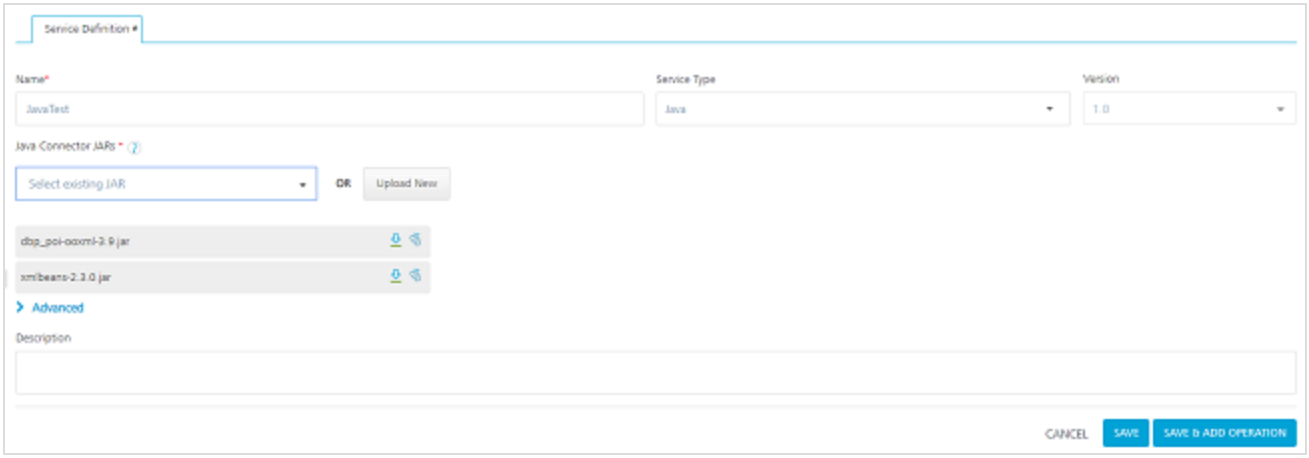
The following is a sample code:

```
ServiceOutputWrapper serviceOutputWrapper =  request.-
getServicesManager().getOperationData()
.getServiceOutputWrapper();
```

> ***Note:*** Request is an instance of data controller request.

21.5.4.4 Configure Java Endpoint Adapter

To configure Java service in the Integration Service Definition tab, follow these steps:



1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **Java**.
3. From the **Java Connector JARs** list, select the required JAR file that contains the classes that implement the JavaService interface for this Integration Service from existing JARs in the account or click **UploadNew** to select the JARs from your local machine.

   > ***Note:*** You can select multiple JAR files from the **Java Connector JARs** list if required.

   > ***Important:*** To upload an updated JAR file, upload the new file, which must have the same name as the old JAR file. The new JAR file overrides the existing file.dsd

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. This step allows you to further filter the data sent to the back end. You can select multiple JAR files if required. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device. You can download the uploaded jars to your local system.
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration.

> ***Note:*** All options in the Advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.
6. Click **Save** to save your service definition.

21.5.4.5 Create Operations for Java

The **Operation List** tab appears when you click **Add Operation** in the **Service Definition** page.

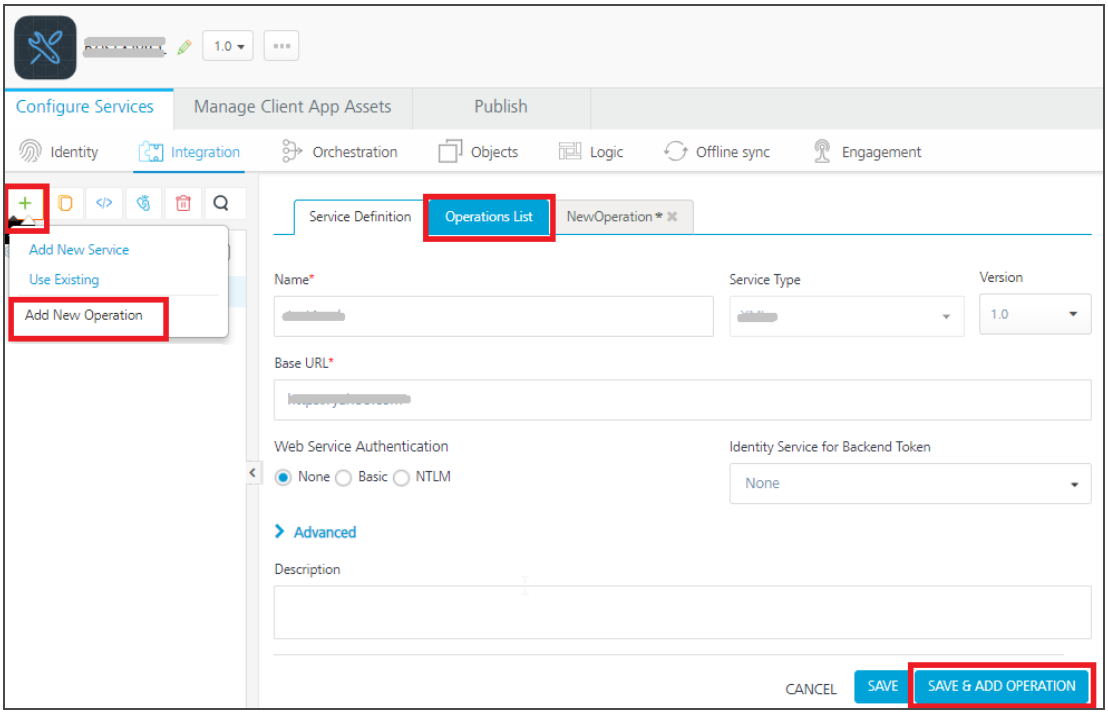
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

2. In the **Operation Modal** tab, follow these steps:

   This tab contains the request input, response output, and advanced sections. The input values are data types, scope, and format types. By default, the system will display the **Request Input** tab.

   > ***Note:*** You can add an entry by clicking the **Add** button if entries for the input and the output tabs do not exist.
   >
   > You can also delete an entry. Select the check box for an entry, and then click **Delete** .

3. To configure an operation, provide the following details:

Field	Description
Name	It is prepopulated with the operation name. You can change the name if required.
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

4. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation - Preprocessor and Postprocessor (for Java and JavaScript)	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Front End API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

21.5.4.6 Configure Request Operation for Java

Integration services accept only `form-url-encoded` inputs for all the input parameters provided in the service input parameters (request input).

To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

You can perform the following actions in Request Input tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .
4. To configure the request input tab, provide the following details:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select Request or Session. It is set to **Request** by default. • **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. • **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric. • **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services.

Field	Description
Data Type	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.
Description	Provide a suitable description.

5. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

21.5.4.7 Configure Response Operation for Java

1. Click the **Response Output** tab, and enter the values for required fields such as name, scope, data type, collection ID, record ID, format and format value.

> ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

In Java service, the response (output) from a backend is not parsed based on the response values. The complete response from the backend is sent to the client device.

> ***Note:*** By default, the `opStatus` and `httpStatusCode` values for Java and JavaScript services are added as 0 and 200.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.
3. Click **SAVE OPERATION** to save the operation. The system updates the operation definition.

   If you click **Cancel**, the **Edit Service Parameters** window will close without saving any information.

> ***Note:*** To add more operations for your Java service, repeat Step 3 through Step 4.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.4.8 How to Edit an Existing Java Adapter

If you want to edit an existing Java service, you can edit details such as service name, JAR files, operation modal details. While editing a Java service, you can change the Java service type. A Java service must be available in the Integration home screen. To add an existing Integration service, refer to **Use an Existing Service**.

To edit an existing Java service, follow these steps:

1. In the **Integration** page, click one of your Java services.
2. Under **Operations** > **Configured Operations**, hover your cursor over the required service, click the **Settings** button, and then click **Edit**.

   The operation details are displayed in the **Edit Service Parameters** dialog.

3. Make the necessary changes in the **Service Definition** section, and click **Update**.
   For more details, refer to How to Configure Service Definition for Java Service.
4. Under the **Operation** section, hover your cursor over the required service, click the **Settings** button, and then click **Edit** to display the **Operation Modal** tab.
   To modify Java operations, refer to How to Configure and Edit Operation Modal.
5. Click **Done** to update your Java service. The Integration page is displayed.

21.5.5 JavaScript Adapter

With the JavaScript Adapter, you can integrate plain JavaScript services to applications in Kony Fabric.

You can upload custom JavaScript files to Kony Fabric. Using JavaScript adapters, you can easily create server-side code and make it available as operations. The JavaScript file must contain functions that a user wants to be made available as operations, along with other supporting functions. You can also add preprocessor and postprocessor to the operations defined. For more details on the structure for JavaScript function definitions, refer to the Limitations section.

For more details about the sample JavaScript code, refer to JavaScript Sample Code for Preprocessor and Postprocessor.

Based on the JDK version supported by Kony Fabric Installer, the JavaScript adapter uses the following JavaScript engines:

- If the Java version is 1.7 or 1.8, then the adapter uses Nashorn JavaScript engine
- If the Java version is below 1.7 , then the adapter uses Rhino JavaScript engine

21.5.5.1 Configure JavaScript Endpoint Adapter

To configure JavaScript service in the Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.

2. From the **Service Type** list, select **JavaScript**.
3. Provide the following details in the JavaScript service definition:

Field	Description
Specify JavaScript libraries	From the **Specify the Javascript libraries** section, select a .JS file, or click **Upload** to select the .JS file from your local machine. The console adds your JS file to the console. The system displays the added JS file's name under the **Specify the Javascript libraries** section. The system allows you to upload more than one JS file at **Specify the Javascript libraries**. Once uploaded, JavaScript files are available across Kony Fabric console. They cannot be deleted, only unlinked. You can unlink uploaded .JS files by clicking the **Unlink** icon. JavaScript file Validation for the JavaScript service: You can validate the associated JavaScript files for the JavaScript service before saving the integration service. To validate the JavaScript file, click the **Validate** icon available for that file. • While uploading a JavaScript file, and if the JavaScript file **has no errors**, the **Validation** icon turns into the **Successful Validation File** icon for that file. And the JavaScript file is uploaded to the Workspace. > ***Note:*** When you link the validated JavaScript file to another JavaScript service, the **Successful Validation File** icon is displayed for that file in the new JavaScript Service. • While uploading a JavaScript file, and if the JavaScript file **has errors**, the Console validates the file and displays the **Error** message dialog with a **Download** link to the validation results. When you click **Download**, a .txt file with the validation results will be downloaded to your local system. In this case you must upload a valid JavaScript file and ensure that the upload is successful. > ***Important:*** While uploading the JavaScript file, and if found errors, the Console does not allow you to upload the file to the Workspace. For the existing linked JavaScript files, the validation statuses icons are treated as follows: • **Validate** icon indicates that the linked JavaScript files are not validated • **Successful Validation File** icon indicates that the linked JavaScript files are validated, and no errors found

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-premises.

> ***Note:*** All the fields in the Advanced tab are optional.

5. In the **Description** field, provide a suitable description for the service.
6. Click **Save** to save your service definition.

21.5.5.2 Create Operations for JavaScript

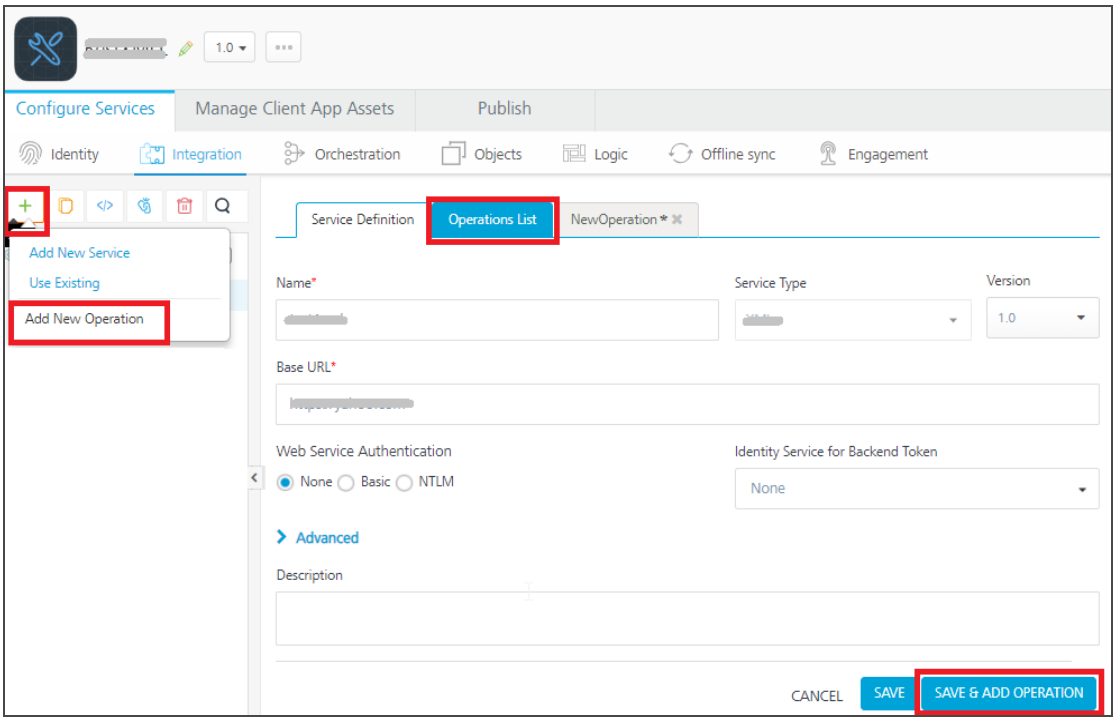
The **Operation List** tab appears when you click **Add Operation** in the **Service Definition** page.

> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

a. Under **Operations List** tab, in the **JS Library** drop-down list, select the required .JS file. For example, Sample.js. This will populate the Function drop-down list.

b. From the **Function** drop-down list, select the required functions. Each function equates to an operation.

c. Click **ADD OPERATION** to create operations with the selected functions.
The new operations are created and listed under the **Configured Operations**.

Operation names are auto-generated in the format : <Name-of-the-JS-file>_<function-name>. For example, `sample_addTwoNumbers`

2. To edit an operation, either click on the operation name or click **Edit** from the **Settings**.

3. In the **Operation Modal** tab, provide the following details to configure an operation:|

Field	Description
Name	It is prepopulated with the operation name. You can change the name if required.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.
Description	Provide a suitable description of your operation.

4. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Properties allows you to configure service call time out cache response. **Cache Response** - the duration in seconds within which the service response is fetched from the cache. Select the **Cache Response** check box, and provide the details in the text box.
Front End API	It allows you map your endpoint/back-end URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

21.5.5.3 Configure Request Operation for JavaScript

Integration services accept only `form-url-encoded` inputs for all the input parameters provided in the service input parameters (request input).

You can perform the following actions in Request Input tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .
4. To configure the request input tab, provide the following details:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Value	Select one of the following options. It is set to **Request** by default. • **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. • **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric. • **Identity**:Selecting this option allows you to send values from identity session as request inputs. Use `<Identity Provider><"Profile"/"Security"><Name of the Parameter>` notation to send identity session values. ◦ **<Identity Provider>** - Name of the identity provider from which the value must be extracted. ◦ **<Parameter>** is the key whose value must be passed along with the service request.

Field	Description
Data Type	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.
Description	Provide a suitable description.

5. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

21.5.5.4 Configure Response Operation for JavaScript

1. Click the **Response Output** tab, and enter the values for required fields such as name, scope, data type, collection ID, record ID, format and format value.

> ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

In JavaScript service, the response (output) from a back end is not parsed based on the response values. The complete response from the back end is sent to the client device.

> ***Note:*** By default, the `opStatus` and `httpStatusCode` values for Java and JavaScript services are added as 0 and 200.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.
3. Click **SAVE OPERATION** to save the operation. The system updates the operation definition.

   If you click **Cancel**, the **Operation Modal** tab closes without saving any information.

> ***Note:*** To add more operations for your JavaScript service, repeat Step 3 through Step 4.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.5.5 Limitations for JavaScript Engine - JavaScript Adapter

- The JavaScript Engine does not support some common JavaScript libraries and global JavaScript functions such as jQuery, setTimeout, setInterval or XMLHttpRequest. However, it provides an alternate mechanism to perform the same operation. You can invoke the required functionality using Java for such cases. For example, consider the XMLHttpRequest API. Since Nashorn does not support this API, you will need to use Java to perform the required operations. You can use URLConnection java class or HttpClient API to achieve the same goal.
- The JavaScript Engine only supports ECMAScript-262 Edition 5.1. It does not support any features of Edition 6 or any nonstandard features provided by other JavaScript implementations.
- The JavaScript Engine does not include a browser plug-in API.
- The JavaScript Engine does not include support for DOM/CSS or any related libraries (such as jQuery, Prototype, or Dojo)

- The JavaScript Engine does not include direct debugging support.
- The JavaScript Engine does not support event loop or a task queue.

21.5.5.6 Limitations for Supported Function Formats - JavaScript Adapter

- Supported formats of the JavaScript function definition as follows:

```
function abc() { … }
```

21.5.6 API Proxy Adapter

With Kony Fabric API Proxy (pass-through proxy) integration service, you can forward the request and response without intermediate transformation (without affecting the actual request and response.)

For example:

If an organization has an existing set of APIs already exposed and wants to secure and throttle the APIs, it can use the API Proxy adapter. By enabling API Proxy, the input (body and headers) of a client's input request is forwarded to the back end and the output response from the back end is forwarded to the device with no changes in the input request and output response.

If your APIs are documented using an Open API Specification file (Swagger file), you can create a new API Proxy service and import the OAS(Swagger) file to automatically create Kony Fabric Operations.

You can configure API Proxy integration service for the following endpoints:

- XML
- JSON

21.5.6.1 Configure API Proxy Endpoint Adapter

To configure API Proxy service in the Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **API Proxy**.
3. Provide the following details in the API Proxy service definition:

Field	Description
Base URL Configuration	• **Base URL** - Type the URL (provide the format and explain the URL parameters) • **Upload Open API file** - Click Upload File, the system allows you to upload an OAS (Swagger) file. Navigate to the swagger file from your local system, and click Open. The system uploads the selected swagger file. The operations for your API proxy service are created based on the resources defined in OAS (Swagger) file.
Web Service Authentication	Web Service Authentication Select one of the following modes: • **None**- Select this option if you do not want to provide any authentication for the service. • **Basic**- Provide User ID and Password if the external Web service requires a form or basic authentication. • **NTLM**- Your service follows the NT LAN Manager authentication process. You are required to provide the User ID, Password, NTLM Host, and NTLM Domain.
Identity Service for Backend Token	Select the Identity service associated with your app if this service needs backend token like access_token from that Identity service to access the backend server.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. This step allows you to further filter the data sent to the back end. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device.

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-premises.
URL Provider Class	Enter the qualified name of the URL Provider Class. For more information, refer URL Provider Support for XML, JSON, SOAP, and API Proxy.

> ***Note:*** All options in the Advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.
6. To enable the proxy, select the **Use proxy from settings** check box. By default, the check box is cleared. The Use proxy from settings check box dims when no proxy is configured under the **Settings > Proxy**.
7. Click **Save** to save your service definition.

21.5.6.2 Create Operations for API Proxy

The **Operations List** tab appears only after the service definition is saved.

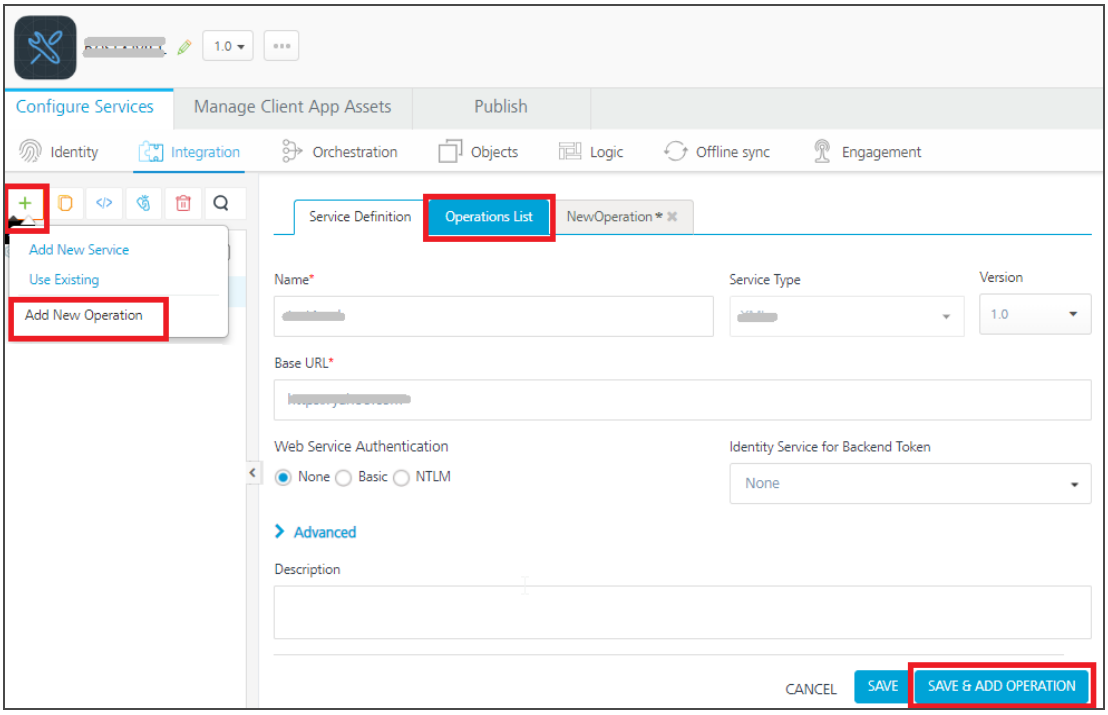
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

> ***Important:*** If you have imported an OAS (Swagger) file, the operations for your API proxy service are created automatically from that OAS (Swagger) file. The endpoint URL is also auto-generated based on the Swagger file. The auto-generated operations for a proxy service will not have request/response parameters.

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

1. To create an operation, provide the following details:

Field	Description
Name	Enter a unique name for your operation.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

Field	Description
Front End HTTP Method	Select a HTTP method that you want to invoke on the integration server. By default, the field is set to **Post** method. ***Note:*** The front-end HTTP methods are used for all non-SDK clients such as API Management users. Invoking a service from an SDK will continue to use the POST method for operations. ***Note:*** From SP3 onwards, the **Front End HTTP Method** is called as **Resource Method**. You can configure the **Resource Method** in the **Advanced> Front End API** section.
Target HTTP Method	Select a HTTP method that you want to invoke on the back-end service from integration server.
Operation Path	Modify the path if required. ***Note:*** If you provide incorrect Salesforce endpoint details, the **Object** list will contain only *_Login* object.
Base URL and Target URL	The **Target URL** field is prepopulated with the URL that you provided at the **Base URL** field. You can add the suffix, if required. For example, to the base URL, you can add suffix such as `/latest` or `/sports` to get latest news or sports news: • `http://feeds.foxnews.com/foxnews/latest` • `http://feeds.foxnews.com/foxnews/sports`

2. For additional configurations of request (or) response operations, provide the following details in

the **Advanced** section:

Field	Description
Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Additional configuration properties (timeout, cachable, unescape embedded xml in response, response encoding, number of connection retries allows you to configure service call time out cache response
Front End API	It allows you map your endpoint/back-end URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

3. Click **Save** to save the operation.

21.5.6.3 Limitations for API Proxy Adapter

- API Proxy service operations and operations for which pass-through (inputs, headers, and outputs) is enabled cannot be used in orchestration services.
  For example, when you create an integration service with an API Proxy type or the operations of XML, SOAP, or JSON endpoints with pass-through enabled, the system does not show the services and operations while configuring orchestration operations.
- Using the MBaaS Client SDK with API Proxy integration service is not supported.

21.5.6.4 How to Enable Pass-through Proxy for Operations

You can also configure the following pass-through proxy flags in operations for adapters such as XML, SOAP, and JSON:

- Under the **Request Input> Body** tab, select the **Enable pass-through input body** check box to forward the body of a client's request to the back end.
- Under the **Request Input>Header** tab, select the **Enable pass-through input header** check box to forward headers of a client's request to the backend.
- Under the **Response Output** tab, select the **Enable pass-through output body** check box to forward the response from the backend to a client.

21.5.7 Mock Data Adapter

The **Kony Mock Data adapter** capability helps you to continue to develop apps when the back-end services that an app connects to are not ready to be leveraged.

There are several instances in an app development life-cycle when back-end systems and app development happen in parallel and only the contract or interface for the app to communicate to a backend is finalized. In this scenario, you can create a response template to stub the response that is expected from the actual backend. The response template can have hard-coded values or use pre-built functions such as concat, firstName, lastName, gender, random, email, and phone, and options to randomize the output within the required criteria.

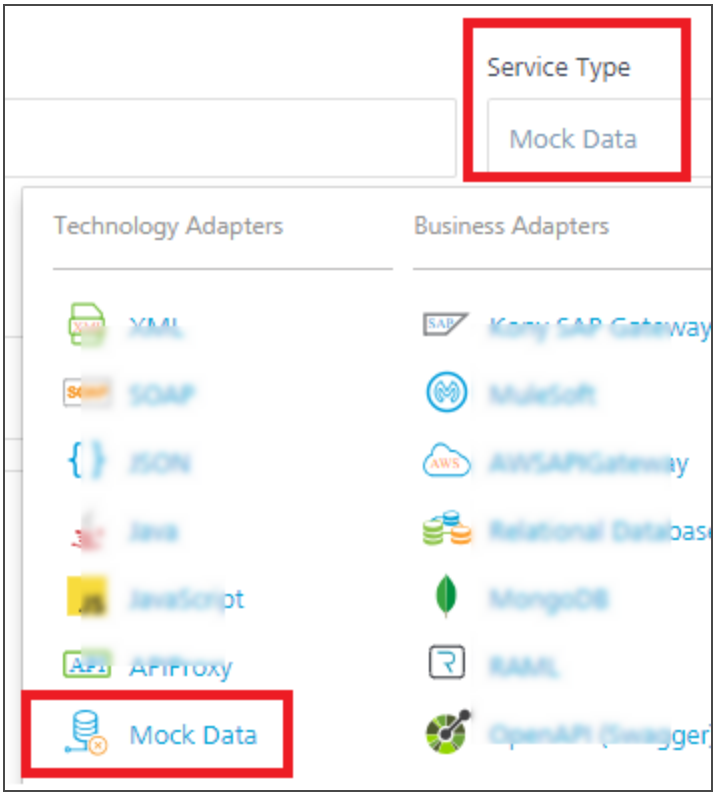
For example, in the scenario mentioned earlier, a Kony app developer can create the service based on a stub template. The stub response template can be set for each operation of a service. The app developers can continue to develop apps based on a sample back-end response from the stub template.

> ***Important:*** Services built with Mock Data adapter will always give you mock data and cannot be switch between live backend and mock data like Stub backend response feature.

21.5.7.1 Configure Mock Data Adapter

To configure your Mock Data Adapter, provide the following details:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **Mock Data**.



3. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. This step allows you to further filter the data sent to the back end. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device. You can download the uploaded jars to your local system.
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration.

> ***Note:*** All options in the Advanced section are optional.

4. In the **Description** field, provide a suitable description for the service.
5. Click **Save** to save your service definition.

21.5.7.2 Create Operations for Mock Data

The **Operations List** tab appears only after the service definition is saved.

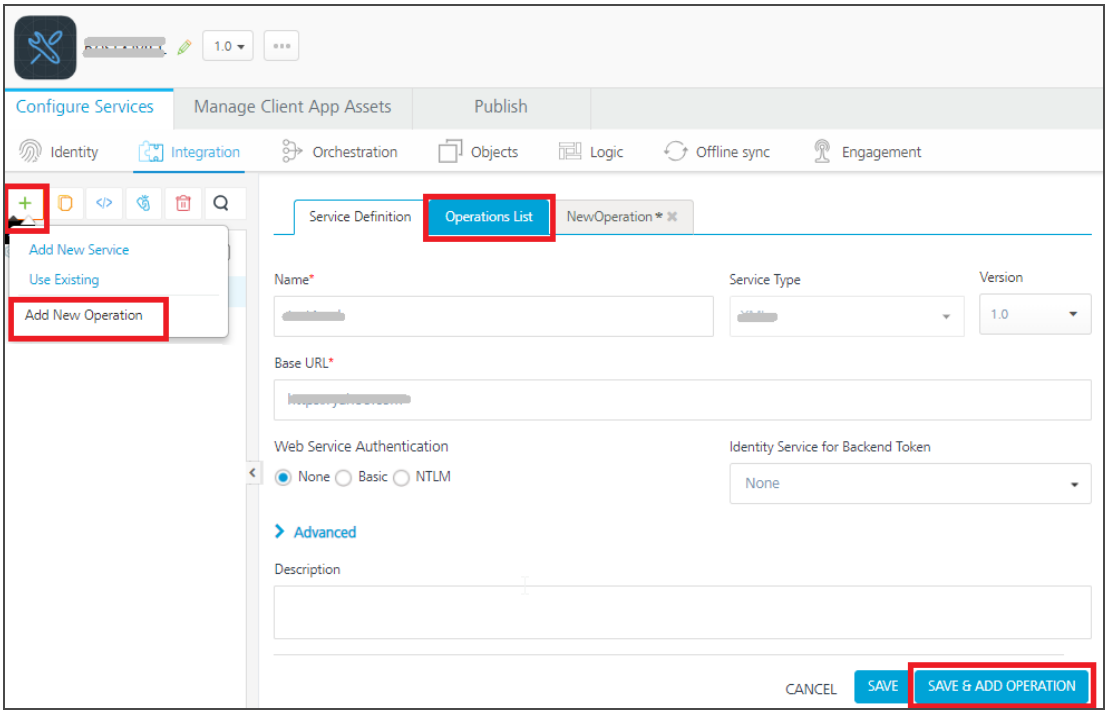
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

2. In the **Operation Modal** tab, follow these steps:

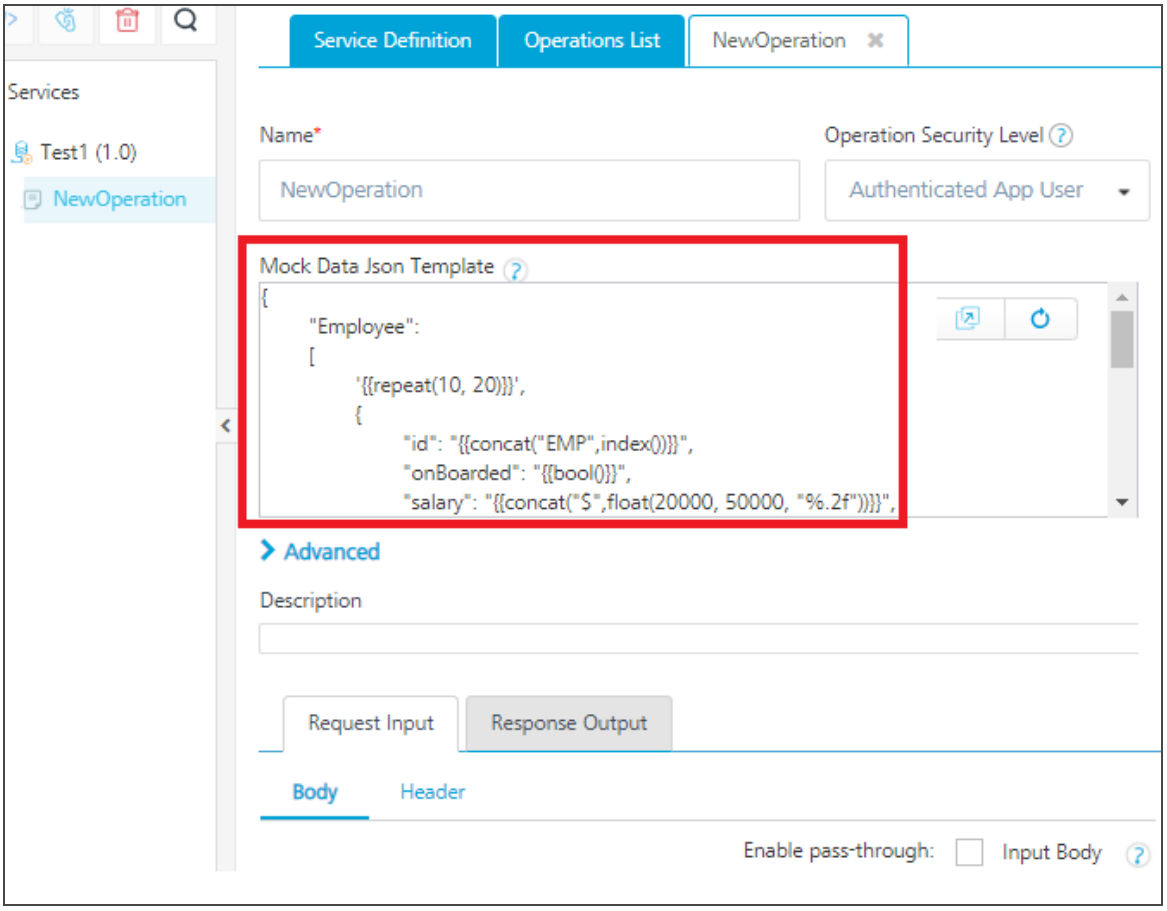
   This tab contains the request input, response output, and advanced sections. The input values are data types, scope, and format types. By default, the system will display the **Request Input** tab.

   > ***Note:*** You can add an entry by clicking the **Add** button if entries for the input and the output tabs do not exist.
   >
   > You can also delete an entry. Select the check box for an entry, and then click the **Delete** button.

3. To configure an operation, provide the following details:

Field	Description
Name	It is prepopulated with the operation name. You can change the name if required.
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

4. Configure your Mock data template in the **Mock Data JSON Template** text field. By default this field is enabled with a sample mock response template.



> ***Note:*** For more information on Pre-built functions supported in Mock Data template, refer Mock Data Template and Supported Pre-built functions in Mock Data

5. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation - Preprocessor and Postprocessor (for Java and JavaScript)	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Additional configuration properties (timeout, cachable, unescape embedded xml in response, response encoding, number of connection retries allows you to configure service call time out cache response
Front End API	It allows you map your endpoint/back-end URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

21.5.7.3 Configure Request Operation for Mock Data

Integration services accept only `form-url-encoded` inputs for all the input parameters provided in the service input parameters (request input).

1. In the **Request Input > Body** tab, do the following:
    a. To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

b. Click **Add Parameter** button to create new entries for the input.

> ***Note:*** - To make duplicate entries, select the check box for the entry, click Copy, and then click Paste.
>
> - To delete an entry, select the check box for an entry, and then click the Delete button.

c. Configure parameters in the client's body, do the following:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Value	Select Request or Session. It is set to **Request** by default. i. **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. ii. **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric. iii. **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services.
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.

Field	Description
Datatype	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.
Description	Provide a suitable description.

2. In the **Request Input > Header** tab, do the following:
    a. To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.
    b. Click **Add Parameter** button to create new entries for the input.

> ***Note:*** - To make duplicate entries, select the check box for the entry, click Copy, and then click Paste.
>
> - To delete an entry, select the check box for an entry, and then click the Delete button.

c. Configure parameters in the client's header, do the following:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.

Field	Description
Value	Select Request or Session. It is set to **Request** by default. By default, this field is set to **Request.** Five different options are available in Kony Fabric under **Request Input > Headers** > **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header**.** • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Constant**: Constant is used to configure the value that is picked and sent to back-end by the Integration Server during the run-time. • **Expression**: Select this option to configure the velocity template expressions for the header values. You cannot edit the default value for expression. • **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** If the header value is scoped as a **Request** (or) **Session** and the same header is accessed under the **Expression** header value, then the expression must be represented as $request.header (or) $session.header. **Example**: If a header 1 value is a request and header 2 value is an expression, then the value of the expression must be $Request.header1.

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
Datatype	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Description	Provide a suitable description.

3. Click **SAVE OPERATION** to save the operation. The system updates the operation definition.

21.5.7.4 Create Response Operation for Mock Data

To forward the response to the client as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.

1. Click the **Response Output** tab, and enter the values for required fields such as name, scope, data type, collection ID, record ID, format and format value.

> ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

2. Click **SAVE OPERATION** to save the operation. The system updates the operation definition.

   If you click **Cancel**, the **Edit Service Parameters** window will close without saving any information.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.7.5 How to Test a Mock Data Response from Admin Console

1. Publish your app to a runtime server.
2. Go to the runtime server in **Admin Console**.
3. Go to the **Integration Services** tab.
4. For the Mock Data service that you created, select the stubbed operation from the **Operations** list.
5. Click **Get Response**.

   > ***Note:*** The `X-Kony-Stub-Response header` as `true` in the back-end response indicates that the response is generated from the Mock Data template.

21.5.7.6 Advanced Parameters in Mock Data

How to Configure a Request Input and Request Header Parameters in Mock Data

You can access input parameters in a Stub template by using the `{{requestBody("<request_param_name>")}}` function. Additionally, you can access headers by using the `{{requestHeader("<header_name>")}}` function.

For example, you want to send the `testUser` request input parameter to the Stub template and the value of the parameter is defined in the Input Parameters section of Console. You can access the input parameter in the stub template as `"inputtest": "{{requestBody("testUser")}}"`.

The following sample Stub template has been configured with the testUser request input parameter.

```
[
  '{{repeat(30,40)}}',
  {
    "locationID": "{{index()}}",
    "company": "{{toUpperCase(company())}}",
    "phone": "+1 {{phone()}}",
    "address": "{{integer(100, 999)}} {{street()}}, {{city()}},
{{state()}}, {{integer(100, 10000)}}",
   "inputtest": "{{requestBody("testUser")}}",
    "latitude": "{{float(-90.000001, 90)}}",
    "longitude": "{{float(-180.000001, 180)}}",
    "office":"{{random("HR Head Office","Sales Head Office","Marketing
Head Office","Development Center")}}"
  }
]
```

Note: Kony functions for request input and header are as follows:

- To access any header with name, `requestHeader`: Syntax: `{{requestHeader ("<header_name>")}}`
- To access a request parameter, `requestBody`: Syntax: `{{requestBody("<request_param_name>")}}`

21.5.7.7 Mock Data Template and Pre-built Functions Supported in Mock Data Template

Mock Data Template

The back-end stubbed response specifies a Mock Data template for stubbing and returning dynamically-generated mock data.

The following table details a sample Mock Data response template and a back-end response that is generated based on the Mock Data template.

Sample Mock Data Template (JSON)	Sample Mock Data Response

<pre>[
'{{repeat(1,2)}}',
{
"locationID": "{{index()}}",
"company": "{{toUpperCase(company())}}",
"phone": "+1 {{phone()}}",
"address": "{{integer(100, 999)}} {{street()}}, {{city()}}, {{state()}}, {{integer(100, 10000)}}",
"latitude": "{{float(-90.000001, 90)}}",
"longitude": "{{float(-180.000001, 180)}}",
"office":"{{random("HR Head Office","Sales Head Office","Marketing Head Office","Development Center")}}"
}
]</pre>
	<pre>[{
	"locationID": "0",
	"company": "RODEOMAD",
	"phone": "+1 371-222-9269",
	"address": "671 Division Place, Grill, South Dakota, 9220",
	"latitude": "-0.29528046",
	"longitude": "159.72824",
	"office": "Marketing Head Office"
	},
	{
	"locationID": "1",
	"company": "ACME",
	"phone": "+1 311-324-8984",
	"address": "257 Adam Place, McCoy, South Carolina, 21245",
	"latitude": "-0.23528046",
	"longitude": "124.72824",
	"office": "Sales Head Office"
	}]</pre>

21.5.7.8 Pre-built Functions Supported in Mock Data Template

The following list of the **sample pre-built functions** are supported in the Mock Data response template:

repeat(<lower_value>, <upper_value>)

Description	**repeat** function repeats JSON or XML objects randomly based on the **value range** provided in the syntax. This is typically set at the beginning of a collection to repeat number of times as required. For example, it can be set to get a random number of transactions for an account for a set format by setting repeat function at the head of the collection as shown in the default template above.
Syntax	`'{{repeat (<lower_value>, <upper_value>)}}'`
Sample Mock Data template	`'{{repeat(30,40)}}'`

repeat(<number>)

Description	**repeat function repeats JSON or XML objects randomly based on the fixed number** provided in the syntax. This is typically set at the beginning of a collection to repeat number of times as required. For example, it can be set to get a random number of transactions for an account for a set format by setting repeat function at the head of the collection as shown in the default template above.
Syntax	`'{{repeat (<number>)}}'`
Sample Mock Data template	`'{{repeat(2)}}'`

integer(min,max)

Description	Generates a random integer in the specified range.
Syntax	`"{{integer(min,max)}}"`

Sample Mock Data template	`"productRating":"{{integer(0,5)}}"`
Sample Mock Data response	`"productRating": "4"`

float(min,max)

Description	Generates a random 32-bit floating point number in the specified range.
Syntax	`"{{float(min,max)}}"`
Sample Mock Data template	`"floatrange": "{{float(3,9)}}"`
Sample Mock Data response	`"floatrange": "6.718242"`

float(min,max,"%.2f")

Description	Generates a random 32-bit floating point number in the specified range of floating point numbers, with an option to round of the number of decimal places.
Syntax	`"{{float(min,max,"%.2f")}}"`
Sample Mock Data template	`"floatrange": "{{float(3,9,"%.2f")}}"`
Sample Mock Data response	`"floatrange": "8.87"`

double(min,max)

Description	Generates a random 64-bit double number in the specified range.
Syntax	`"{{double(min,max)}}"`

Sample Mock Data template	`"double": "{{double(2,8)}}"`
Sample Mock Data response	`"double": "7.654668228367652"`

long(min,max)

Description	Generates a random long number in the specified range.
Syntax	`"{{long(min,max)}}"`
Sample Mock Data template	`"network":"{{long(200,500)}}"`
Sample Mock Data response	`"network":"378"`

uuid()

Description	Generates a random GUID.
Syntax	`"{{uuid()}}"`
Sample Mock Data template	`"objects": "{{uuid()}}"`
Sample Mock Data response	`"objects": "fb81ad08-42e3-4b61-9a2c-636f95952766"`

hex()

Description	Generates a random 16 bytes hexadecimal string.
Syntax	`"{{hex()}}"`
Sample Mock Data template	`"color": "{{hex()}}"`

Sample Mock Data response	`"color": "b53ff7fa4b63b18cdf7729e85c39e660"`

hex(size)

Description	Generates a random hexadecimal string according to the specified size in bytes.
Syntax	`"{{hex(size)}}"`
Sample Mock Data template	`"color": "{{hex(2)}}"`
Sample Mock Data response	`"color": "1b41"`

objectId()

Description	Generates a hexadecimal string of size 12 bytes.
Syntax	`"{{objectId()}}"`
Sample Mock Data template	`"id": "{{objectId()}}"`
Sample Mock Data response	`"id": "1c91293a4c777000585e04e6"`

bool()

Description	Generates a random Boolean value, either True or False.
Syntax	`"{{bool()}}"`
Sample Mock Data template	`"stockAvailable": "{{bool()}}"`
Sample Mock Data response	`"stockAvailable": "true"`

bool(<probability>)

Description	Generates a random Boolean value, either True or False as per the given probability.
Syntax	`"{{bool(probability)}}"`
Sample Mock Data template	`"stockAvailable": "{{bool(0.9)}}"`
Sample Mock Data response	`"stockAvailable": "true"`

index()

Description	Generates an incrementing index integer for each record with a specific starting point.
Syntax	`"{{index()}}"`
Sample Mock Data template	`"locationID": "{{index()}}"`
Sample Mock Data response	`"locationID": "0"`

index("index-name")

Description	Generates an incrementing index integer for each record based on the name of the index.
Syntax	`"{{index("index-name")}}"`
Sample Mock Data template	`"index-number": "{{index("abc")}}"`
Sample Mock Data response	`"index-number": "42"`

index(<number>)

Description	Generates an incrementing index integer for each record with a specific starting point.
Syntax	`"{{index(78)}"`
Sample Mock Data template	`"index-number": "{{index(78)}}"`
Sample Mock Data response	`"index-number": "4248"`

index("index-name",<number>)

Description	Generates an incrementing index integer for each record based on both a specific starting point and name of the index.
Syntax	`"{{index("index-name",78)}"`
Sample Mock Data template	`"index-name-number": "{{index("abc",78)}}"`
Sample Mock Data response	`"index-name-number": "626"`

lorem(count,"words")

Description	Generates a random dummy text. User must specify the count of words required.
Syntax	`"{{lorem(count,"words")}}"`
Sample Mock Data template	`"productDescription": "{{lorem(5, "words")}}"`
Sample Mock Data response	`"productDescription": "lorem ipsum porta sit curabitur"`

lorem(count,"paragraphs")

Description	Generates a random dummy paragraph. User must specify the count of paragraphs required.
Syntax	`"{{lorem(count,"paragraphs")}}"`
Sample Mock Data template	`"about": "{{lorem(2, "paragraphs")}}"`

Sample Mock Data response	`"about": " Lorem ipsum eros amet accumsan non quisque ut molestie nullam sagittis tincidunt.Lorem ipsum quis aliquam nostra.`
	
	`Lorem ipsum litora tristique arcu habitant.Lorem ipsum nulla mauris inceptos fusce adipiscing tortor torquent."`

phone()

Description	Generates a random phone number. The phone number is preceded by `+` to indicate a country code. This allows to set phone numbers for different countries. For example, for USA/Canada, the phone number format is `+1 xxx xxx xxxx`
Syntax	`"{{phone()}}"`
Sample Mock Data template	`"phone": "+1 {{phone()}}"`
Sample Mock Data response	`"phone": "+1 371-222-9269"`

gender()

Description	Generates a random gender value, either male or female.

Syntax	`"{{gender()}}"`
Sample Mock Data template	`"gender": "{{gender()}}"`
Sample Mock Data response	`"gender": "female"`

date()

Description	Generates the current date.
Syntax	`"{{date()}}"`
Sample Mock Data template	`"date": "{{date()}}"`
Sample Mock Data response	`"date": "Tue, 11 Sep 2018 10:55:44 GMT"`

date("java-simple-date-format")

Description	Generates the current date in the specified date format.
Syntax	`"{{date("java-simple-date-format")}}"`
Sample Mock Data template	`"date_format": "{{date("dd-MM-yyyy HH:mm:ss")}}"`
Sample Mock Data response	`"date_format": "11-09-2018 12:00:00"`

date("begin-date","end-date","java-simple-date-format")

Description	Generates a random date in the specified range and specified format. Your date range input must be in this format: `dd-MM-yyyy HH:mm:ss`

Syntax	`"{{date("begin-date","end-date","java-simple-date-format"}}"`
Sample Mock Data template	`"{{date("01-01-2014 12:00:00", "01-01-2018 12:00:00", "yyyy-MM-dd'T'HH:mm:ss Z")}}"`
Sample Mock Data response	`"date_of_joining": "2015-12-22T22:00:33 +0000"`

date("begin-date","end-date")

Description	Generates a random date in the specified range of dates with default format. Your input must be in this format `EEE, d MMM yyyy HH:mm:ss z`
Syntax	`"{{date("begin-date","end-date")}"`
Sample Mock Data template	`"date_default_format": "{{date("01-01-2014 12:00:00", "01-01-2018 12:00:00", "EEE, d MMM yyyy-MM-dd'T'HH:mm:ss Z")}}"`
Sample Mock Data response	`"date_default_format": "Tue, 21 Jun 2016-06-21T19:26:18 +0000"`

timestamp()

Description	Generates the current timestamp (milliseconds, between the current time and midnight, January 1, 1970 UTC):
Syntax	`"{{timestamp()}}"`
Sample Mock Data template	`"time": "{{timestamp()}}"`
Sample Mock Data response	`"time": "1536662962710"`

timestamp("begin-date","end-date")

Description	Generates the current timestamp (milliseconds, between the current time and midnight, January 1, 1970 UTC) between two dates with default format. Your input must be in this format `EEE, d MMM yyyy HH:mm:ss z`
Syntax	`"{{timestamp("begin-date","end-date"}}"`

country()

Description	Generates a random country name.
Syntax	`"{{country()}}"`
Sample Mock Data template	`"country": "{{country()}}"`
Sample Mock Data response	`"country": "Montenegro"`

countryList()

Description	Generates a JSON mapping with all country codes and country name.
Syntax	`"{{countryList()}}"`

countryList("country_code_1", "country_code_2")

Description	Generates a JSON mapping with given country codes and country name.
Syntax	`"{{countryList("IN", "US", "UK")}}"`

city()

Description	Generates a random city.
Syntax	`"{{city()}}"`

Sample Mock Data template	`"city": "{{city()}}"`
Sample Mock Data response	`"city": "Belva"`

state()

Description	Generates a random state name.
Syntax	`"{{state()}}"`
Sample Mock Data template	`"state": "{{state()}}"`
Sample Mock Data response	`"state": "Pennsylvania"`

company()

Description	Generates a random company name.
Syntax	`"{{company()}}"`
Sample Mock Data template	`"company": "{{company()}}"`
Sample Mock Data response	`"company": "Gorganic"`

lastName()

Description	Generates a random last name.
Syntax	`"{{lastName()}}"`
Sample Mock Data template	`"lastname": "{{lastName()}}"`

Sample Mock Data response	`"lastname": "Randolph"`

firstName()

Description	Generates a random first name.
Syntax	`"{{firstName()}}"`
Sample Mock Data template	`"firstname": "{{firstName()}}"`
Sample Mock Data response	`"firstname": "Mays"`

username()

Description	Generates a random username based on the first initial of your random first name and random last name in lowercase.
Syntax	`"{{username()}}"`
Sample Mock Data template	`"username": "{{username()}}"`
Sample Mock Data response	`"username": "nbarr"`

email()

Description	Generate a random email address in the standard format. For example: the email standard format is `<firstName>.<lastName>@<domain>.<com>`
Syntax	`"{{email()}}"`
Sample Mock Data template	`"email": "{{email()}}"`

Sample Mock Data response	`"email": "irma.england@mazuda.com"`

email("mydomain.com")

Description	Generates a random email address with the specified domain name in the standard format. For example: the email standard format is `<firstName>.<lastName>@<"mydomain.com">`
Syntax	`"{{email("mydomain.com")}}"`
Sample Mock Data template	`"email": "{{email("mydomain.com")}}"`
Sample Mock Data response	`"email": "wilson.allison@mydomain.com"`

ssn()

Description	Generates a random social security number.
Syntax	`"{{ssn()}}"`
Sample Mock Data template	`"ssn": "{{ssn()}}"`
Sample Mock Data response	`"ssn": "285-59-5039"`

ipv4()

Description	Generates a random ipv4 address.
Syntax	`"{{ipv4()}}"`
Sample Mock Data template	`"ipv4": "{{ipv4()}}"`

Sample Mock Data response	`"ipv4": "218.110.1.153"`

ipv6()

Description	Generates a random ipv6 address.
Syntax	`"{{ipv6()}}"`
Sample Mock Data template	`"ipv6": "{{ipv6()}}"`
Sample Mock Data response	`"ipv6": "ipv6": "e801:bde0:c898:3a0e:2401:1b23:2199:4d3f"`

ipv6("upper")

Description	Generates a random ipv6 address with all the alphabetical characters in uppercase.
Syntax	`"{{ipv6("upper")}}"`

ipv6("lower")

Description	Generates a random ipv6 address with all the alphabetical characters in lowercase.
Syntax	`"{{ipv6("lower")}}"`

concat(var arg)

Description	Generates a string by concatenating all the strings given as inputs.
Syntax	`"{concat("A","B","C","D")}}"`
Sample Mock Data template	`"id": "{{concat("EMP",index())}}"`
Sample Mock Data response	`"id": "EMP2942"`

substring("word",3)

Description	Generates a substring from the given string and the starting position.
Syntax	`"{{substring("word",3)}}"`
Sample Mock Data template	`"substring":"{{substring("SampleApps", 3)}}"`
Sample Mock Data response	`"substring":"pleApps"`

substring("long word", 1, 6)

Description	Generates a substring from the given string, the starting position, and the end position.
Syntax	`"{{substring("long word", 1, 6)}}"`
Sample Mock Data template	`"substring":"{{substring("Sampleword" , 1,6)}}"`
Sample Mock Data response	`"substring":"ample"`

random("SampleValue1","SampleValue2","SampleValue3","SampleValue4")

Description	**random** function provides string values to each record randomly based on the predefined sample string values provided while invoking the function.
Syntax	`"{{random ("SampleValue1","SampleValue2","SampleValue3","SampleValue4")}}"`

Sample Mock Data template	`"office":"{{random("HR Head Office","Sales Head Office","Marketing Head Office","Development Center")}}"`
Sample Mock Data response	`"office": "Marketing Head Office"`

alpha()

Description	Generates a random string with alphabetic characters of length between 10 to 20 characters.
Syntax	`"{{alpha()}}"`
Sample Mock Data template	`"alpha": "{{alpha()}}"`
Sample Mock Data response	`"alpha": "jFulYDTuFQBk"`

alpha(min,max)

Description	Generates a random string with alphabetic characters and length in the given range.
Syntax	`"{{alpha(min,max)}}"`
Sample Mock Data template	`"alphaRange": "{{alpha(15,20)}}"`
Sample Mock Data response	`"alphaRange": "hTyLpLYMHGJkYrEB"`

alpha(length)

Description	Generates a random string with alphabetic characters of given length.
Syntax	`"{{alpha(length)}}"`
Sample Mock Data template	`"alphaLength": "{{alpha(7)}}"`
Sample Mock Data response	`"alphaLength": "ZKziDST"`

alphaNumeric()

Description	Generates a random string with alpha-numeric characters of length between 10 to 20 characters.
Syntax	`"{{alphaNumeric()}}"`
Sample Mock Data template	`"alphaNumeric": "{{alphaNumeric()}}"`
Sample Mock Data response	`"alphaNumeric": "hIrUkxDzdH4VIR86g6G"`

alphaNumeric(min,max)

Description	Generates a random string with alpha-numeric characters of length in the specified range.
Syntax	`"{{alphaNumeric(min,max)}}"`
Sample Mock Data template	`"alphaNumericRange": "{{alphaNumeric (15,20)}}"`
Sample Mock Data response	`"alphaNumericRange": "k5BBh4U45o19vhaO4LQ"`

alphaNumeric(length)

Description	Generates a random string with alpha-numeric characters of the given length.
Syntax	`"{{alphaNumeric(length)}}"`
Sample Mock Data template	`"alphaNumericLength": "{{alphaNumeric(7)}}"`
Sample Mock Data response	`"alphaNumericLength": "Zphdrag"`

toLowerCase("text")

Description	Converts the string value to lowercase letters.
Syntax	`"{{toLowerCase(company())}}"`
Sample Mock Data template	`"company": "{{toLowerCase(company())}}"`
Sample Mock Data response	`"company": "rodeomad"`

toUpperCase("text")

Description	Converts the string value to uppercase letters.
Syntax	`"{{toUpperCase(company())}}"`
Sample Mock Data template	`"company": "{{toUpperCase(company())}}"`
Sample Mock Data response	`"company": "RODEOMAD"`

##Escape braces

Description	If you want to escape braces from within a function use a single escape character as seen in the example below:
Syntax	`"{{concat("\{", "test", "\}")}}`

##Nesting functions

Description	Mock Data service supports nesting functions as well. For example, if you wanted to create results that looked like dollar amounts, you can do the following:

Syntax	`"{{concat("$",float(0.90310, 5.3421, "%.2f"))}}`
	
	or something like this if you wanted a capitalized F or M:
	
	`{{toUpperCase(substring(gender(),0,1))}}`

21.5.8 Kony SAP Gateway Adapter

With Kony Fabric, you can access external Kony SAP services by using the Kony SAP Gateway adapter. Based on your Kony SAP Gateway authentication, you can use Kony SAP libraries and objects along with the supported HTTP methods in your app.

21.5.8.1 Configure Kony SAP Gateway Endpoint Adapter

To configure Kony SAP Gateway service in the Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.

2. From the **Service Type** list, select **Kony SAP Gateway**.
3. Provide the following details to create the Kony SAP Gateway service:

Field	Description
Select Authentication Services	• **Use Existing Identity Provider** - to select an identity provider. This drop-down lists all identity providers only if you have already created identity providers for SAP in the Identity page. Fill in the details for the following fields: i. From the **Identity** list, select your Kony SAP Gateway identity. The details for the selected identity are displayed in the **Gateway address & port** text box. You cannot modify these details. ii. Under the **User ID** and **Password**, provide valid log-in credentials that you created while registering with Kony SAP services. iii. In the **Default Caller ID**, provide the ID that Kony SAP Gateway uses for logging and auditing. iv. In the **Default Caller Group**, provide the ID that Kony SAP Gateway uses for logging and auditing. This information is optional. • **Specify Login Endpoint**- to configure a new endpoint. Fill in the details for the following fields: i. In the **Gateway address**, enter the domain. For example, connect.kony.com. ii. In the **Port** text box, enter a valid port number ranging from 1 to 65535. iii. In the **Header parameter name prefix *** text box, enter the header. For example, KonySAP. iv. Under the **User ID** and **Password**, provide valid log-in credentials that you created while registering with Kony SAP Gateway services. v. In the **Default Caller ID**, provide the ID that Kony SAP Gateway uses for logging and auditing. vi. In the **Default Caller Group**, provide the ID that Kony SAP Gateway uses for logging and auditing. This information is

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-premises.

> ***Note:*** All the fields in the Advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.
6. To enable the proxy, select the **Use proxy from settings** check box. By default, the check box is cleared.
   The **Use proxy from settings** check box dims when no proxy is configured under the **Settings > Proxy**.
7. Click **Save** to save your service definition.

21.5.8.2 Create Operations for Kony SAP Gateway

The **Operations List** tab appears only after the service definition is saved.

> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

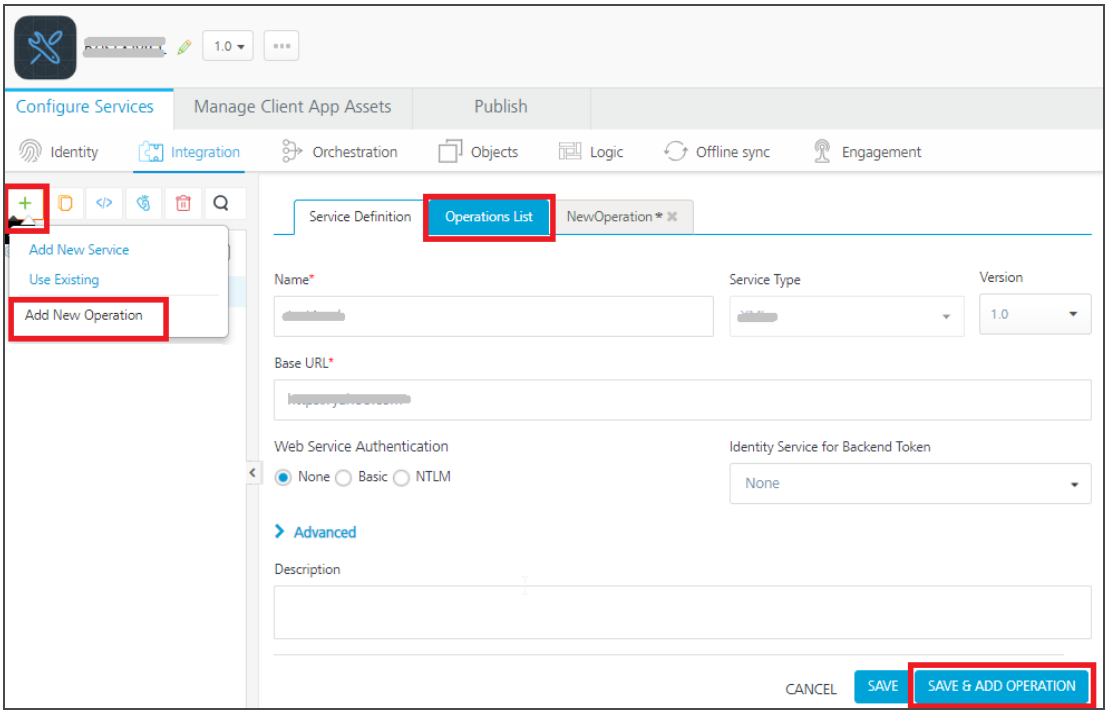
Based on your Kony SAP Gateway authentication, the system loads all tables such as libraries and objects along with supported HTTP methods.

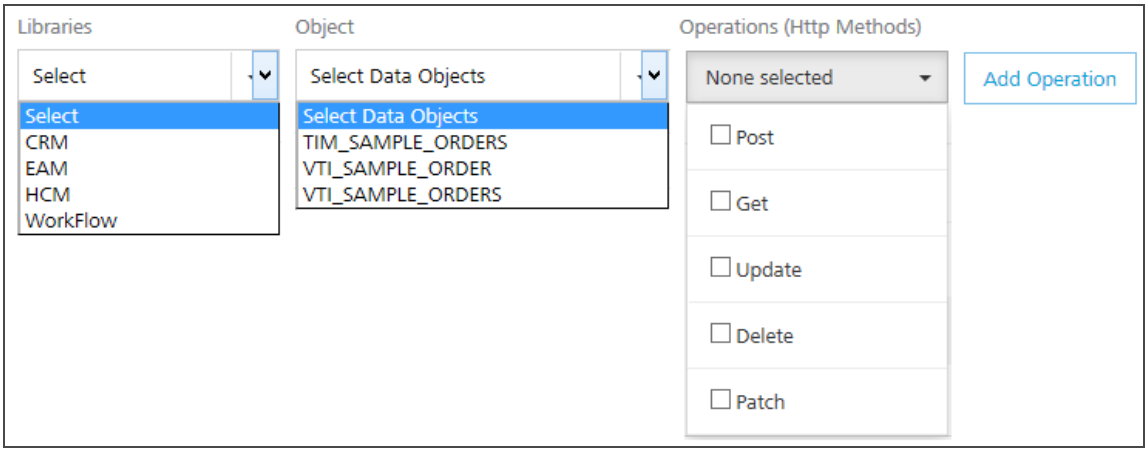
To create operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.





> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

2. Under **Operations List** tab, from the **Libraries** list, select a library - for example, CRM. The system loads available objects and operations for the selected library.

3. From the **Object** list, select an object.

4. From the **Operations (HTTP Methods)** list, select the required check boxes for each operation.
5. To configure more operations for your Kony SAP Gateway integration service, repeat Steps a through b. You can select a new library and object, and supported operations.
6. Click **ADD OPERATION**. The system adds your operation to the Operations List page, and it also adds your new Kony SAP Gateway service into the **Integration** page.
7. To edit an operation, either click on the required operation name or click **Edit** from the **Settings** in the **Operations List** screen. The operation details page is displayed.
8. To configure an operation, provide the following details:

Field	Description
Name	Enter a unique name for your operation.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.
Front End HTTP Method	Select a HTTP method that you want to invoke on the integration server. By default, the field is set to **Post** method. ***Note:*** The front-end HTTP methods are used for all non-SDK clients such as API Management users. Invoking a service from an SDK will continue to use the POST method for operations.
Target HTTP Method	Select a HTTP method that you want to invoke on the back-end service from integration server.

Field	Description
Operation Path	Modify the path if required. ***Note:*** If you provide incorrect Salesforce endpoint details, the **Object** list will contain only *_Login* object.

9. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Front End API	It allows you map your endpoint (or) backend URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

Note: All options in the Advanced section are optional.

21.5.8.3 Configure Request Operation for Kony SAP Gateway

Integration services accept only `form-url-encoded` inputs for all the input parameters provided in the service input parameters (request input).

You can perform the following actions in Request Input tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.

3. To delete an entry, select the check box for an entry and click **Delete** .

4. Under the **Body** tab, provide the following details:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select Request or Session. It is set to **Request** by default. • **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. • **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.
Datatype	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.

5. Under the **Header** tab, provide the **Custom HTTP Headers**. For example, **POST** or **GET**. The following Custom HTTP Headers are required by the external source:
    - **ID**: The rows are created based on the selected operation. Change the value if required.
    - **TEST VALUE**: Enter a value. A test value is used for testing the service.
    - **Scope**: Select request or session. By default, this field is set to Request.
6. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

21.5.8.4 Configure Response Operation for Kony SAP Gateway

1. Click the **Response Output** tab to view the output test values, such as name, scope, data type.

    > ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.
3. Click **SAVE OPERATION** to save the operation.

    > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.8.5 How to Edit or Test an Existing Kony SAP Gateway Adapter

If you want to edit an existing Kony SAP service, you can edit details such as service name, authentication service information, operations.

Each operation contains four tabs, including input, attributes, output, and advanced. If you want to test an existing operation for Kony SAP service - for example, get or put - enter necessary test values in the input and the advanced tabs. The results are displayed in the JSON format. The input values can be data types, test values, and session keys.

To edit or test an existing Kony SAP integration service, follow these steps:

1. In the **Integration** page, click one of your SAP services.
2. Make the necessary changes in the **Service Definition** and **Operations** sections. You can test an operation by inputting values. To test an operation, refer to **How to configure Kony SAP Gateway Operations**.
3. Click **Done** to save the changes. The system displays the **Integration** page.

21.5.9 Mulesoft Adapter

MuleSoft (Anypoint Platform™) is a platform that helps app developers to design custom APIs and deploy to a Mule Enterprise Service Bus runtime (ESB). With MuleSoft integration service in Kony Fabric, developers can interact with more than 50 types of adapters.

To integrate a MuleSoft service in Kony Fabric, developers need to create a project in MuleSoft Studio, export the project to a local system, and then deploy it to Cloud Hub. On top of the project deployment, a RESTful API modeling language (RAML) file needs to be built, which defines all the API definitions in the project. A RAML file also contains the defined schemas with properties. When a user creates a project from AnyPoint API Studio with any adapter and builds a RAML file over the project, all the APIs can be used in Kony Fabric integration service. When a Kony Fabric user selects a MuleSoft adapter from Kony Fabric Console, based on the cloud hub portal credentials of the MuleSoft adapter, the system retrieves metadata from a RAML file and displays all APIs of a RAML file. Developers can add these APIs as operations in MuleSoft integration service in Kony Fabric Console.

Kony Fabric discovers the MuleSoft endpoints through a RAML file. Kony Fabric parses a RAML file and exposes all the MuleSoft endpoints through the integration service. Mobile app developers can use the configured MuleSoft integration service and access the backend systems supported by MuleSoft's adapters.

> ***Note:*** In Kony Fabric Console, you can provide login credentials of MuleSoft or upload a RAML file to configure an integration service.

21.5.9.1 Advantages

- Developers can design custom APIs
- Developers can use more than 50 types of connectors.

21.5.9.2 Limitations

- When an APIGroup is created, only one RAML file needs to be created. Multiple files are not supported.
- Reconfiguration of apps during publish is not supported for MuleSoft.
- The schema defined must always be in JSON format. XML schema is not supported.
- API Gateway (On-premises / Cloud) and Mule ESB (On-premises) are not supported.

21.5.9.3 Prerequisites

- Log in to MuleSoft with your credentials at https://anypoint.mulesoft.com/#/signin.
- Download IDE of AnyPoint Studio (MuleSoft) from https://www.mulesoft.com/platform/studio.
- Create a project and deploy the project.
- Build a RAML file. For more details, refer to MuleSoft Documentation at https://developer.mulesoft.com/anypoint-platform.

Adding a MuleSoft service involves the following steps:

- How to Configure Service Definition for MuleSoft Service
- How To Create Operations for MuleSoft Service
- How to Configure Operations for MuleSoft Service

21.5.9.4 Configure MuleSoft End-point Adapter

To configure a Mulesoft adapter in Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service. When you enter the name, the name is updated for the active service under the **Services** section in the left pane.
2. From the **Service Type** list, select **Mulesoft**.

   > ***Note:*** XML is selected, by default.

3. Provide the following details to create a SOAP service.

Fields	Description
Version	Select the version for the service.

Fields	Description
Choose API Discovery Type	Click one of the following modes • **RAML File** - Select the option and upload RAML file from your local machine. The system adds your RAML file to the console. The system displays the added RAML file's name under the **Choose API Discovery Type** section. • **AnyPoint Platform URL** - Select this option and the system displays the MuleSoft URL in the text box. Under the **User ID** and **Password**, provide valid log-in credentials that you created while registering with `MuleSoft AnyPoint` Platform. ***Note:*** You cannot modify the details in AnyPoint Platform URL. Click **Test Login** to test the Mulesoft connection details.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Custom code	To specify a JAR associated to the service, select one from the **Select Existing JAR** drop-down menu or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device.

Field	Description
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-

Field	Description
Use proxy from settings	To enable the proxy, select the check box. The **Use proxy from settings** check box dims when no proxy is configured under the **Settings > Proxy**.

> ***Note:*** All options in the Advanced section are optional.

5. Enter the **Description** for the service.
6. Click **SAVE** to save your service definition.

21.5.9.5 Create Operations for MuleSoft

The **Operations List** tab appears only after the service definition is saved.

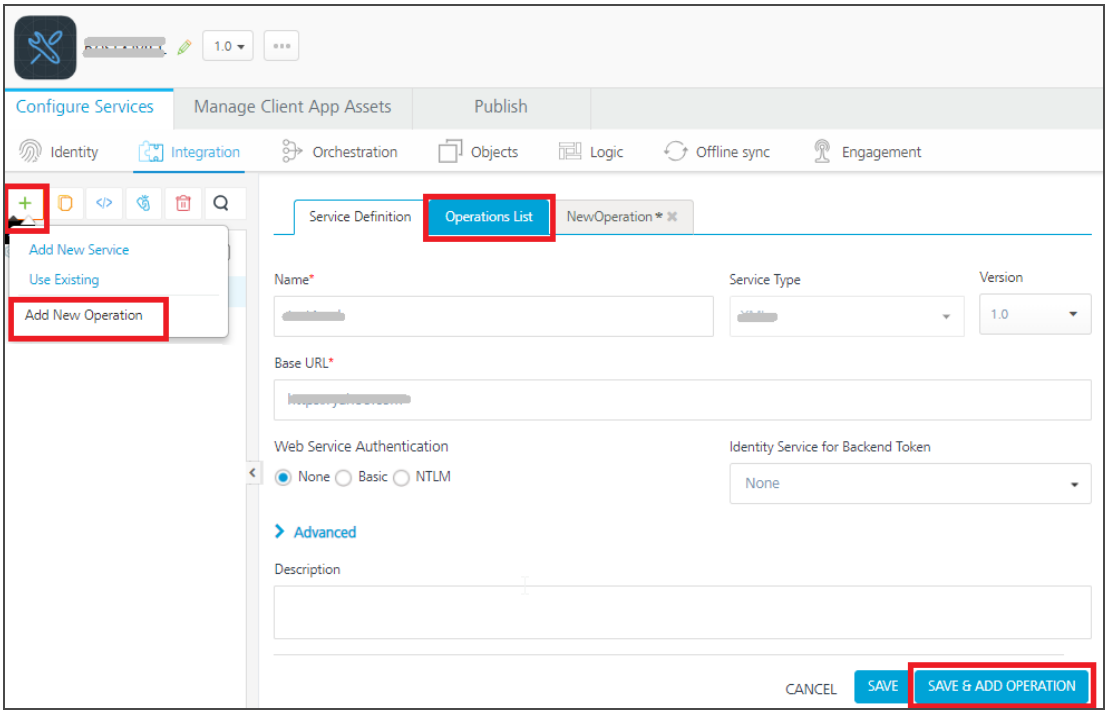
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

1. In **Operations List** tab, click **Operation Name** list and select **Operations**.
2. Click **Add Operation**. The **Operation List** tab appears.
3. Provide the following details to configure operations.

Field	Description
Groups	Select the title of the RAML file that is uploaded at AnyPoint CloudHub. Based on the selected groups, the **Versions** drop-down list is loaded with the versions of the RAML file.

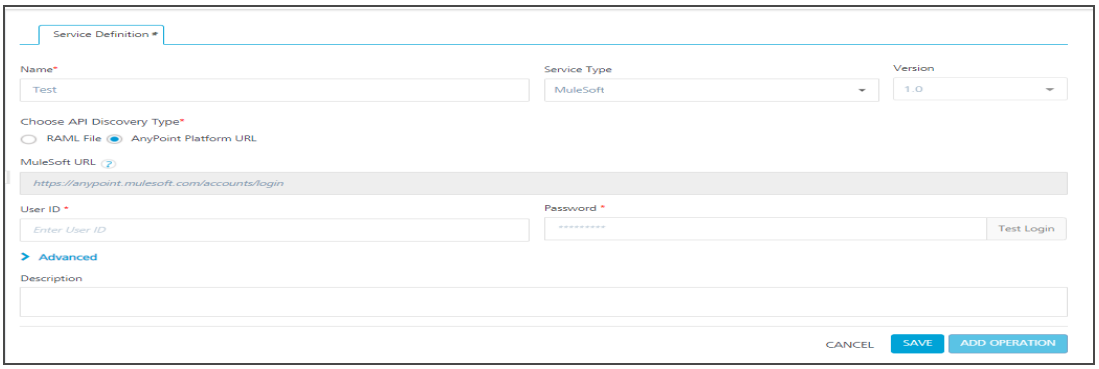
Field	Description
Versions	Select the required version. You can select only one version of the RAML file.
Resources	Select the required resource.
Operation (HTTP Methods)	Select the required check boxes for the defined methods in the RAML file. You can click **Select all** check box to select all the operations.

4. Click **Add Operation**. The added operation appears under the **Configured Operations** section.

   The system creates a JSON service for the MuleSoft service. Operation names are auto-generated in the format. The default name format of a MuleSoft operation is `<operation_name><resource_name>`. You can change the operation name if required.
   For example, `postfolder`.

5. Once you create operations for MuleSoft service, you can configure the operations such as adding parameters, adding test values, and fetching the response. Under **Configured Operations**, hover your cursor over the create operation, click the **Settings>> Edit**.

> ***Note:*** To edit an operation, you can also click the operation from the service tree pane.



The system displays the selected operation in the edit mode. Provide the following details to configure the operation.

Fields	Description
Name	The **Name** field is pre-populated with fields names of the selected database. You can edit this field.

Fields	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default. The **Operation Path** field is prepopulated with MuleSoft URL. You can edit this field if required.

6. For additional configuration of request (or) response operations, provide the following details in the **Advanced** section.

Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the **Advanced** section for operations are optional.

7. Enter the **Description** for the operation.

21.5.9.6 Configure Request Operation for MuleSoft

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

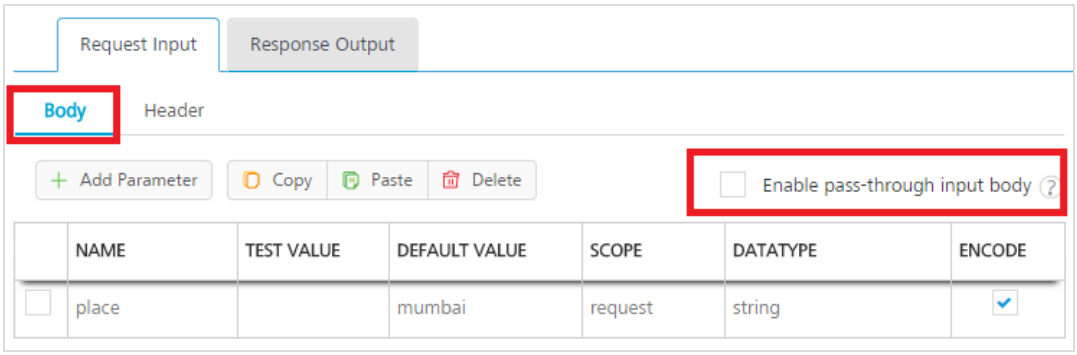
You can perform the following actions in **Request Input** tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.

3. To delete an entry, select the check box for an entry and click **Delete** .

4. Under the **Body** tab, perform the following actions:

   a. To forward the body of the client's request to backend as it is, select the **Enable pass-through input body** check box. For more details on API Proxy service, refer to [How to Enable Pass-through Proxy for Operations](#).



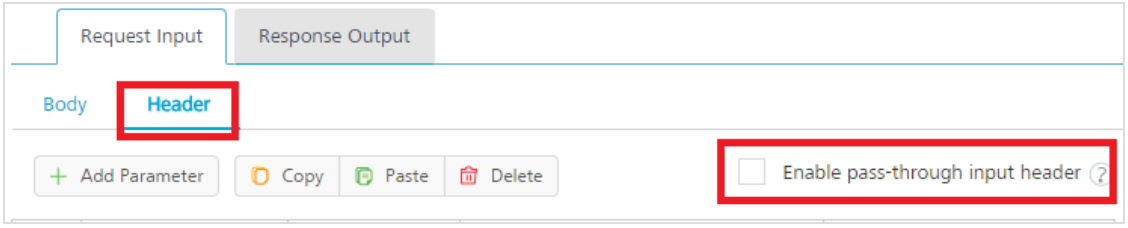
   b. To configure parameters in the clients body, do the following:

Field	Description
Name	Enter the name for the request input parameter. ***Note:*** In the **Body** tab > **NAME** field, the input parameters are prepopulated based on the properties of the input schema of a RAML file.

Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.
TEST VALUE	Enter a value. A test value is used for testing the service.

Field	Description
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select request or session. This field is set to **Request**, by default. The default datatype for the selected column is loaded under **DATATYPE** field.
Encode	Select the checkbox to enable an input parameter to be encoded. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere the HTML URL encoding standards.

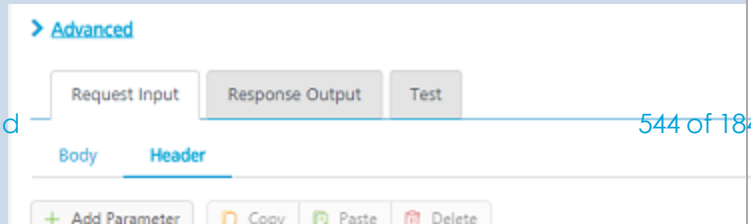
5. Click the **Header** tab to provide the following customer headers, perform the following actions:



You must provide the custom HTTP headers based on the operation. For example, post or get.

a. To configure parameters in the client's header, do the following:

Field	Description
Name	Provide custom HTTP headers required by the external source.

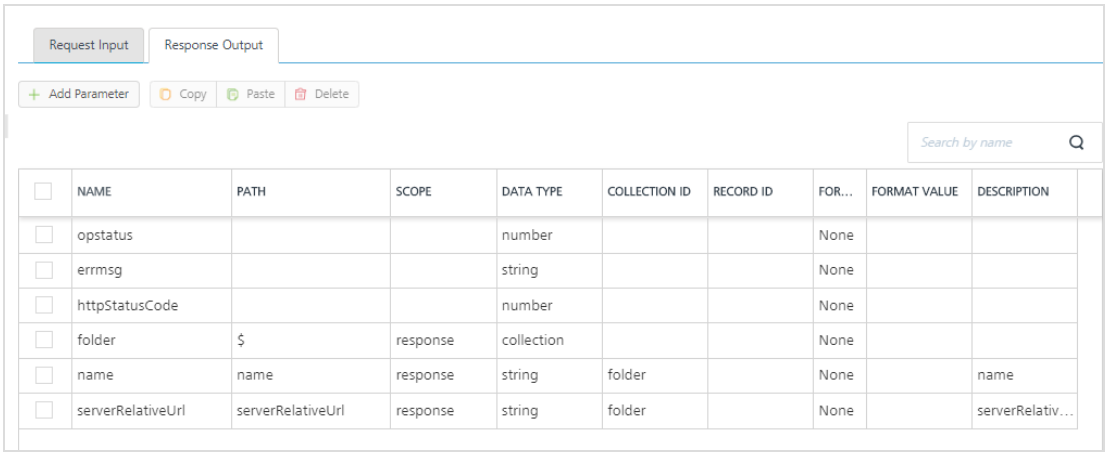
Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default. ***Note:*** If the header value is scoped as a **Request** (or) **Session** and the same header is accessed under the **Expression** header value, then the expression must be represented as $request.header (or) $session.header. **Example**: If a header 1 value is a request and header 2 value is an expression, then the value of the expression must be $Request.header1. 

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Change the syntax, if required. The default value will be used if the test value is empty.
SCOPE	Select request or session. The field is set to **Request**, by default.
Description	Enter the **Description** for the request input.

6. To validate the details, click **Fetch Response**. You can refer to Test a Service Operation for the steps to test a service.The result of the operation appears.

21.5.9.7 Configure Response Operation for MuleSoft

Click the **Response Output** tab to configure the fields of the table for displaying the data.



> ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

In the **Response Output** tab, configure the fields of the table for displaying the data:

1. The **Name** field in the Response Output tab is prepopulated with the properties of output schema of a RAML file.

   Enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format and format value

   > ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

2. To validate the details, click **Test**. You can refer to Test a Service Operation for the steps to test a service. The result of the operation appears.

3. Click **SAVE OPERATION**.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.10 AWS API Gateway Adapter

AWS API Gateway is a new data adapter of Kony Fabric integration services. With Kony Fabric AWS API Gateway integration service, you can connect to the services configured and deployed under API Gateway Service in Amazon Web Services.

Kony Fabric AWS API Gateway supports the following integration types of Amazon API Gateway:

- Lambda Function
- HTTP Proxy
- Mock Integration
- AWS Service Proxy

> ***Note:*** Kony Fabric supports AWS API Gateway services, which returns JSON response.

21.5.10.1 Configure AWS API Gateway Endpoint Adapter

To configure AWS API Gateway service in the Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **AWS API Gateway**.
3. Provide the following details in the AWS API Gateway service definiton:

Field	Description
Regions	Select your region that you have configured AWS API Gateway for your AWS account.
AWS URL	It displays the AWS API gateway URL for the selected region. You cannot modify this field.
Access Key ID	Enter the access key ID that you received from your AWS account.
Secret Access ID	Enter the secret access key ID that you received from your AWS account.
Test Connection	Click **Test Connection** to test you AWS connection details. The system displays `Valid Gateway address and port` if the details are correct.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. This step allows you to further filter the data sent to the back end. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device.

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-premises.

Note: All options in the Advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.
6. To enable the proxy, select the **Use proxy from settings** check box. By default, the check box is

cleared.

The **Use proxy from settings** check box dims when no proxy is configured under the **Settings > Proxy**.

7. Click **Save** to save your service definition.

21.5.10.2 Create Operations for AWS API Gateway

The **Operations List** tab appears only after the service definition is saved.

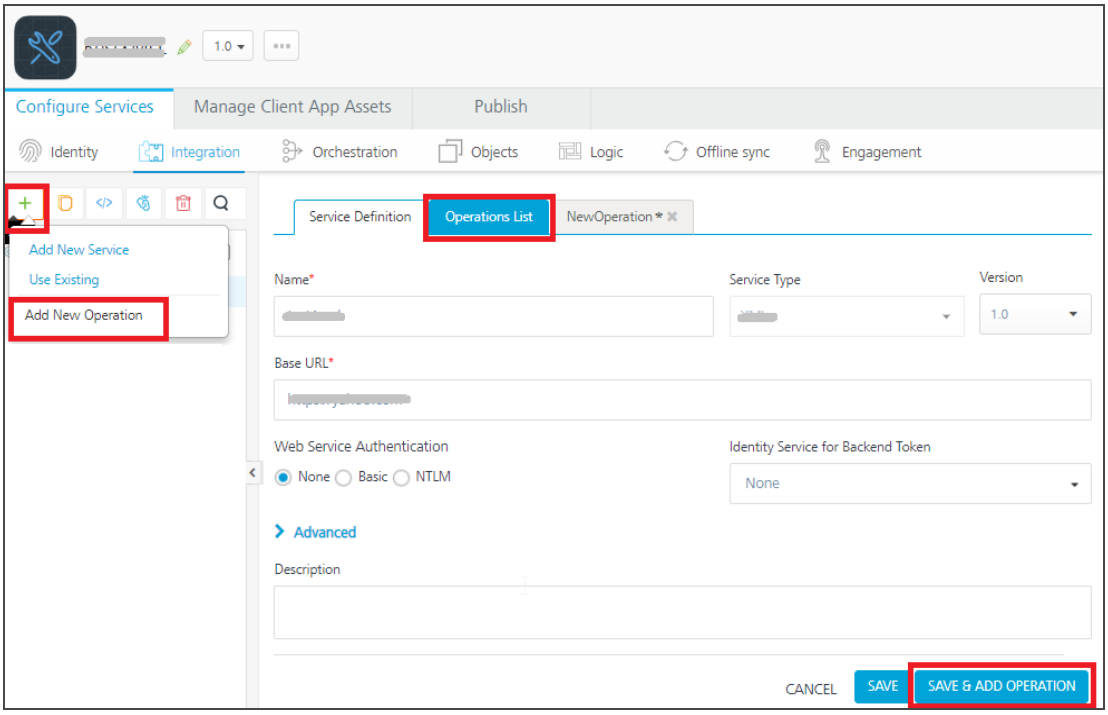
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



Note: To use an existing integration service, refer to How to Use an Existing Integration Service.

2. To create an operation, provide the following details:

Field	Description
API	From the drop down list, select an API that is deployed at Amazon Services.
Resource	Select the required resource from the drop down list.
Stages	Select the required stage from the drop down list.
Methods	Select the required method from the drop down list.
Add Operation	Click Add Operation to add the created operation in the **Configured Operations** Section. Operation names are auto-generated in the format. The default name format of a AWS API operation is `<method_name><api_name><resource_name>`. You can change the operation name if required.

21.5.10.3 Configure Operation for AWS API Gateway

Once you create operations for an AWS API Gateway service, you can configure operations as follows:

1. To edit an operation, either click on the required operation name or click **Edit** from the **Contextual Menu** in the **Configured Operations** screen. The operation details page is displayed.
2. Provide the following details in the operation details page:

Field	Description
Name	It is prepopulated with the operation name. You can change the name if required.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

3. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation - Preprocessor and Postprocessor (for Java and JavaScript)	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Additional configuration properties (timeout, cachable, unescape embedded xml in response, response encoding, number of connection retries allows you to configure service call time out cache response
Front End API	It allows you map your endpoint/back-end URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

21.5.10.4 Configure Request Operation for AWS API Gateway

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

You can perform the following actions in **Request Input** tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.

3. To delete an entry, select the check box for an entry and click **Delete** .
4. Under the **Body** tab, provide the following details:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select Request or Session. It is set to **Request** by default. • **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. • **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.

Field	Description

Datatype	Select one of the following data types.
	• **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit.
	• **Date** - Date format
	  ▪ If data type is string, then the options in the Format Type are Currency, Number and Date.
	  ▪ If the data type is number, then the options in the Format Type are Currency and Date.
	  ▪ If the data type is boolean, then the options in the Format Type and Format Value text box are disabled.
	• **Boolean** - A value that can be true or false.
	• **Number** - An integer or a floating number.
	• **Collection** - A group of data, also referred as data set.

Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.

5. Under the **Header** tab, provide the following details:

> ***Important:*** If AWS APIs with API Key Required is set to "true", the api key needs to be sent as request header as x-api-key.

Field	Description
Name	Provide custom HTTP headers required by the external source.
Scope	Select one of the following options. It is set to **Request** by default. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the value pairs to the back-end. The option helps a user to configure the default value. The default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from the session context based on the user configuration. • **Constant**: Constant is used to configure the value that is picked and sent to back end by the Integration Server during run time. • **Expression**: Select this option to configure the velocity template expressions for the header values. ***Note:*** You cannot edit the default value for the expression.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Description	Provide a suitable description of the service.

6. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

21.5.10.5 Configure Response Operation for AWS API Gateway

1. In the **Response Output** tab, provide the following details:.

   The **Name** field in the Response Output tab is prepopulated with properties of the output API.

   Enter the values for required fields such as name, scope, data type, collection ID, record ID, format and format value.

   > ***Note:*** If you define parameters inside a record as a session, the session scope will not get reflected for the parameters.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

3. Click **Save Operation** to save the operation. The system displays the **Operation** section for your service.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.11 Relational Database Adapter

With Kony Fabric database adapter, you can connect to your own database as an endpoint. After you configure the database adapter in Kony Fabric Console, you can perform create, read, update, and delete (CRUD and Binary CRUD) operations on data in the tables and invoke stored procedures, functions, and views.

For example, banks maintain a store of users and their details. With Kony Fabric database adapter, banks can connect to their own databases and manage customers data.

21.5.11.1 Advantages of Kony Fabric Database adapter

Following are the advantages of using Kony Fabric Database Adapter:

- Admins can connect to the given database.
- Admins can manage the databases using CRUD operations.
- When an admin is creating CRUD operations, the admin can access the configured schema.
- Data types: All major data types are supported.
- Binary Support is available in Database Adapter with Range header.
- Kony supports eight ODATA parameters for the read operation such as `$filter`, `$orderby`, `$top`, `$skip`, `$select`, `$expand`, `$batchsize`, and `$batchid`.

  > ***Note:*** From V8 SP4 FP2 onwards, Kony supports $batchsize and $batchid for MySQL database.

  > ***Note:*** You must provide the required batch size to the **$batchsize** parameter.

  > ***Note:*** If you set **$batchsize** in the first batch call, the response of the batch generates details for the **nextBatchId** and **hasMoreRecords**. If the **hasMoreRecords** is set to **true** in the response, for the second batch call onwards, you must specify the **$batchid** with the value of **nextBatchId**. It indicates that there are more records to be downloaded in the next batch.

- Kony supports ODATA parameter `$filter` for the Delete operation.
- Kony supports ODATA Methods: substringof and indexof. For more information, refer to Support for ODATA Methods (substringof and indexof).
- You can execute custom SQL queries if you want to operate on multiple tables through single operation like Joins.

21.5.11.2 Create a Database Adapter

Creating a database adapter involves the following steps:

- How to Configure a Service Definition for a Database Service
- How to Create CRUD Operations for a Database Service
- How to Configure CRUD Operations for a Database Service

Configure Database End-point Adapter

To configure the Database Adapter in the Integration service definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **Database**.

   If you select **Database**, the **Database Type, Database Connection URL**,and other details are displayed.

   > ***Note:*** XML is selected, by default.

3. Provide the following details in the Connection Parameters section to create a Database Adapter.

Field Name	Description
Database	Type the database driver class details manually in the box or select the database type from the database list. If you select the database type from the database list, the driver class details of the selected database will be filled automatically in the field. The database driver class for each database type is as follows: • MySQL: com.mysql.jdbc.Driver • PostgreSQL: org.postgresql.Driver • SQL Server: com.microsoft.sqlserver.jdbc.SQLServerDriver • Oracle: oracle.jdbc.OracleDriver
Database Connection URL	Type the connection URL in the format given in the help text. The help text in the box changes based on the selected database driver class. You can also copy and paste the connection URL format by clicking the help icon at Database Connection URL, and make the required changes to format. The format of the connection URL is `jdbc:<type of Database>://<ip_address>:<port>` The connection URL of the supported database types are as follows: • Oracle: `jdbc:oracle:thin:@10.10.1.192:8081:sid` • MySQL: `jdbc:mysql://10.10.1.192:8081` • SQL Server: `jdbc:sqlserver://10.10.1.190:8081` • PostgreSQL: `jdbc:postgresql://10.10.1.192:8081`

Field Name	Description
User ID	Type the user ID for the connection URL.
Password	Type the password for the user ID which you have entered.

Field Name	Description

Advanced Settings	Specify the advanced settings like Soft Delete Flag and Autogenerated fields.
	
	`{'softdeleteflag':'<columnName>', 'softdeleteactivevalue':<active-value>, 'softdeleteinactivevalue':<inactive-value>, 'AutogeneratedFields': {'TableName': ['Field1Name','field2Name'...] } {"issoftdeletedefault ": "false", "lastupdatetimestamp": "<columnname>"}`
	
	Note: When the **lastupdatetimestamp** is set, the RDBMS adapter identifies this column as the ChangeTrackingColumn to track delta/updated records. All the objects in the given Object Service must have the same ChangeTracking column name.
	
	• To soft delete a record - 'softdeleteflag':'<columnName>.
	
	Note: To soft delete a column mention the softdelete column name. Ensure all the tables have the same softdelete column name.
	
	• Specify the active/inactive values for softdelete column by setting -.
	
	`{'softdeleteactivevalue':<active-value>, 'softdeleteinactivevalue':<inactive-value> }`
	
	• To specify the autogenerated fields - `''AutogeneratedFields': {'TableName': ['Field1Name','field2Name'...] }}`
	
	• To turn off the default behavior of soft delete set - `{"issoftdeletedefault ": "false"}`.

Field Name	Description
Max Pool Size	Specify maximum number of connections in the connection pool.
Connection Timeout (ms)	Type the connection timeout in milliseconds.

4. Click **Test Connections** if you want to check the database connection. If the entered details are correct, the system displays the following message: Valid Database connection details.

5. If your database is configured with a proxy server, you must select an **environment**and then click **Test Connection** to test the database connectivity. The environment should be `=> V8.3`. For example, you have the Kony Fabric Console installed on one machine, and the Runtime and Database servers installed on another machine. When you create an integration service of type Relational Database, the Console must be established with a VPN connection to the RDBMS server. So that, when you test the Database connection, the test case will be successful. You can do this by selecting the correct environment for your Runtime Server which will ensure a VPN connection between the Console and the Runtime Server and test the database.
   If the entered details are correct, the system displays the following message: Valid Database connection details.

6. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Custom Code	To specify a JAR associated to the service, select one from the **Select Existing JAR** drop-down menu or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration.
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration.

> ***Note:*** All options in the Advanced section are optional.

7. Enter the **Description** for the service.
8. Click **SAVE** to save your service definition.

Create CRUD Operations for Database Adapter

The **Operations List** tab appears only after the service definition is saved.

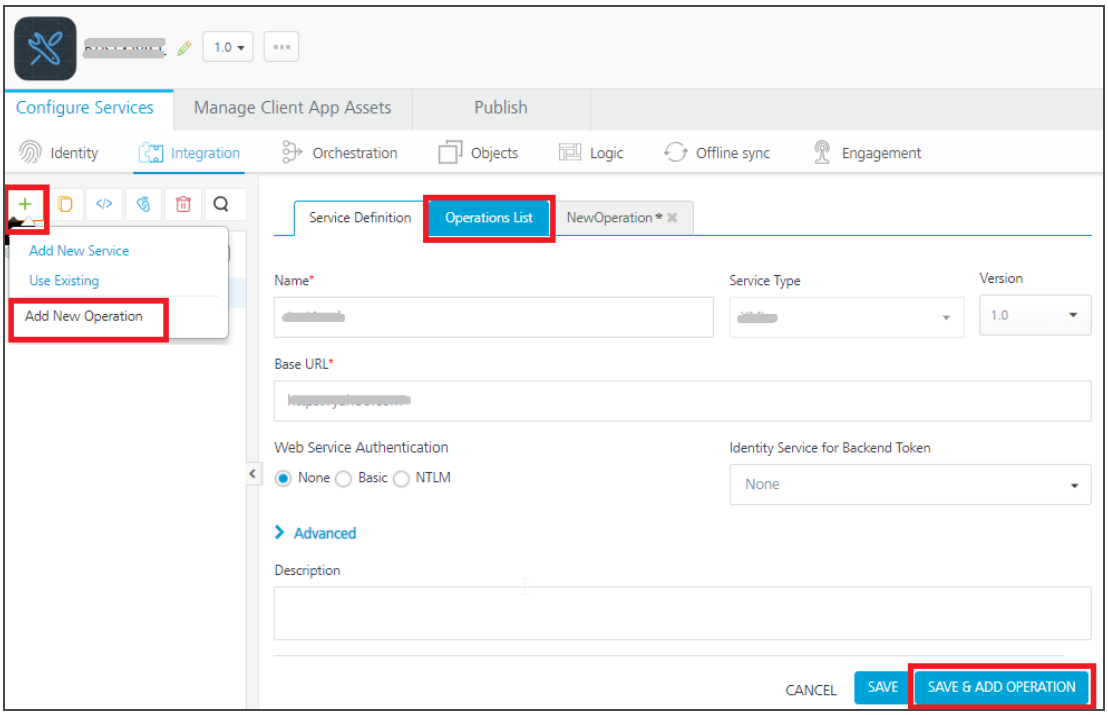
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



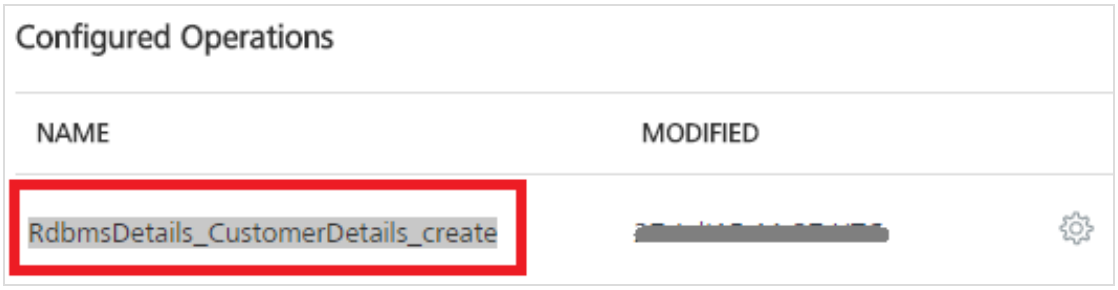
> ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

1. Provide the following details to configure an operation.

Field	Description
Schema	Select the schema loaded based on your database configuration.
Object Type	Select the object from the list. • **Table** - Select available tables from the **Object**list, and select CRUD operations for the tables. • **View** - Select available views from the **Object** list and, select CRUD operations for the tables. • **Stored Procedures** - Select available stored procedures from the **Object** list. ***Note:*** While adding operations to a database integration service, Kony Fabric allows you to select CRUD operations only for table and view object types.
Object	Select the checkboxes for the selected object type. You can select one or more objects.
Operations	From the operations list, select the required check boxes for CRUD operations for tables. For views, only GET is supported. You can select one or more CRUD operations.

2. Click **Add Operation**. The new operations are created under **Configured Operations** section.

   Operation names are auto-generated in the format. The default name format of a database operation is `<schema_name>_<table_name>_<operations>`. You can change the operation name if required.
   For example, `RdbmsDetails_CustomerDetails_create`.



When an admin creates CRUD operations for a database adapter, the admin is under a particular schema. To customize fields, refer to How to Configure CRUD Operations for Database Service.

Configure CRUD Operations for a Database Adapter

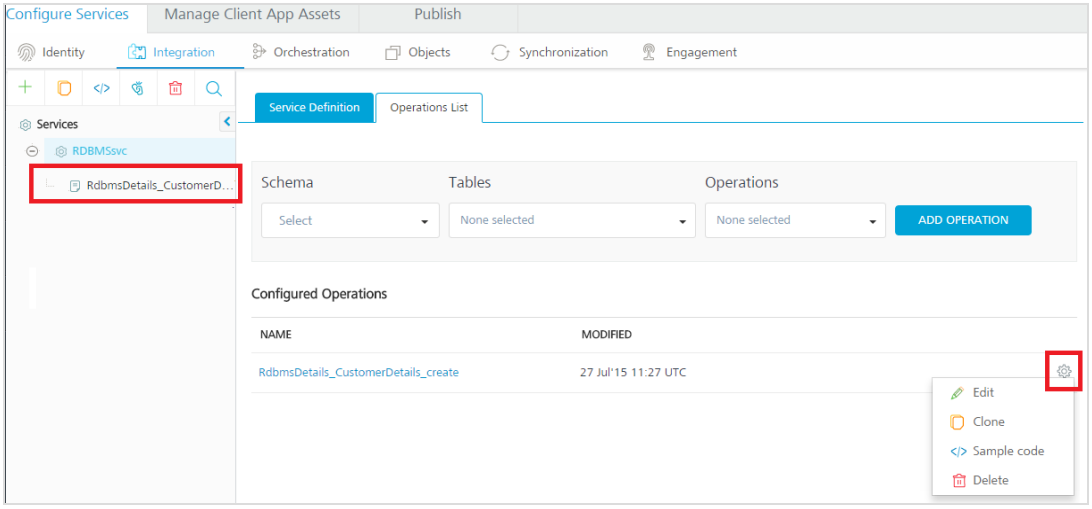
After you create an Database service, configure CRUD operations as following:

- Create a Database Record with Create Operation
- Query a Database and Display Information with Read Operation
- Update a Database Record with Update Operation
- Delete a Database Record with Delete Operation
- Create SQL Custom Query

Create a Database Record with Create Operation

Under **Configured Operations** list, hover your cursor over the create operation and click **Settings (the three dots) >> Edit**.

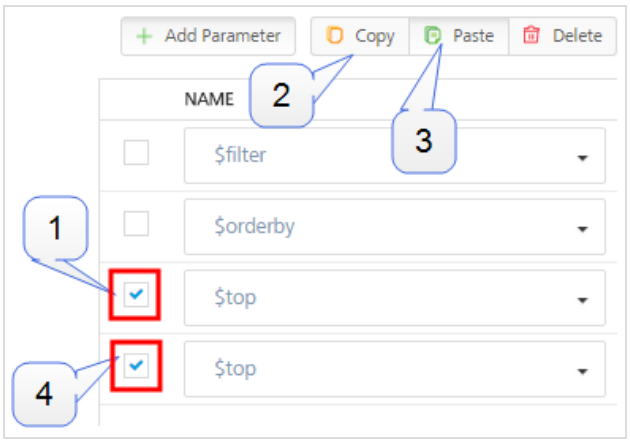
> ***Note:*** To edit an operation, you can also click the operation from the service tree pane.



The system displays the selected operation in the edit mode. You can perform following actions in this window:

1. You can add an entry by clicking **Add Parameter**, if entries for the input and the output tabs do not exist.

2. To make duplicate entries, select the check box for the entry, click **Copy**, and then click **Paste**.



3. To delete an entry, select the check box for an entry, and click **Delete**.
4. Enter the following fields to create a database record.

Fields	Description
Name	The **Name** field is pre-populated with fields names of the selected database. You can edit this field.

Fields	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default.

Fields	Description
Action	The field is pre-populated with operation names of the selected database. You cannot edit this field.

9. For additional configuration of request (or) response operations, provide the following details in the **Advanced** section.

Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the **Advanced** section for operations are optional.

Configure Request Operation for Database Adapter

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

a. To configure the parameters, do the following:

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
SCOPE	Select request or session. This field is set to **Request**, by default.
DATA TYPE	The default datatype for the selected column is loaded under **DATATYPE** field.
Encode	Select the checkbox to enable an input parameter to be encoded. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere the HTML URL encoding standards.
Description	Enter the description for request input.

b. To validate the details, click **Fetch Response**. For more information, refer Test a Service Operation. The result of the operation appears.

c. Click **SAVE OPERATION** to save the changes in the create operation.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

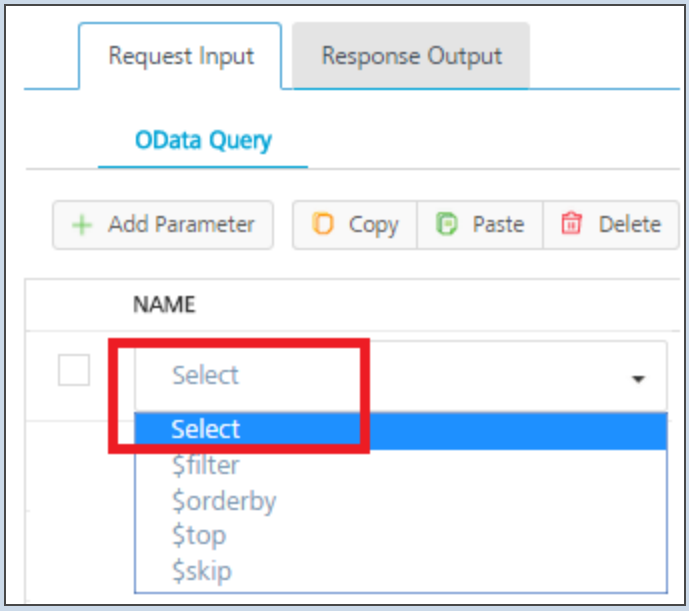
How to Query a Database and Display Information with the Read Operation

1. Under **Configured Operations**, hover your cursor over the **Read** operation, click **Settings**>> **Edit**.

   The system displays the selected operation in the edit mode. The Read operation has the **Request Input** and **Response Output** tabs.

   Note: You can add an entry by clicking the **Add Parameter** button if entries for the input and the output tabs do not exist.

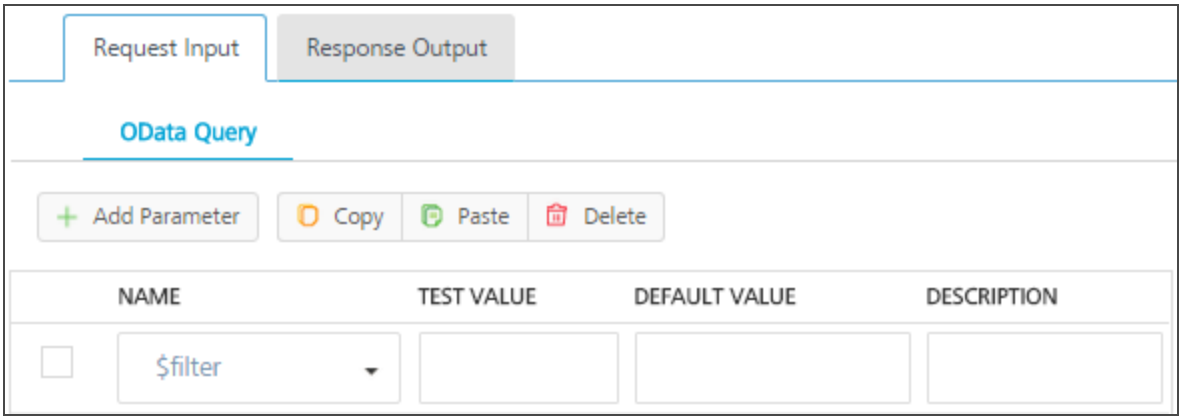
   In the read operation, the **Name** drop-down list contains a **Select** option that acts as a label for the list. **Select** itself is not a command.



   - To make duplicate entries, select the check box for the entry, click **Copy**, and then click **Paste**.

   - To delete an entry, select the check box for an entry, and then click the **Delete** button.

2. In the **Request Input**, configure the following ODATA commands to filter the data:

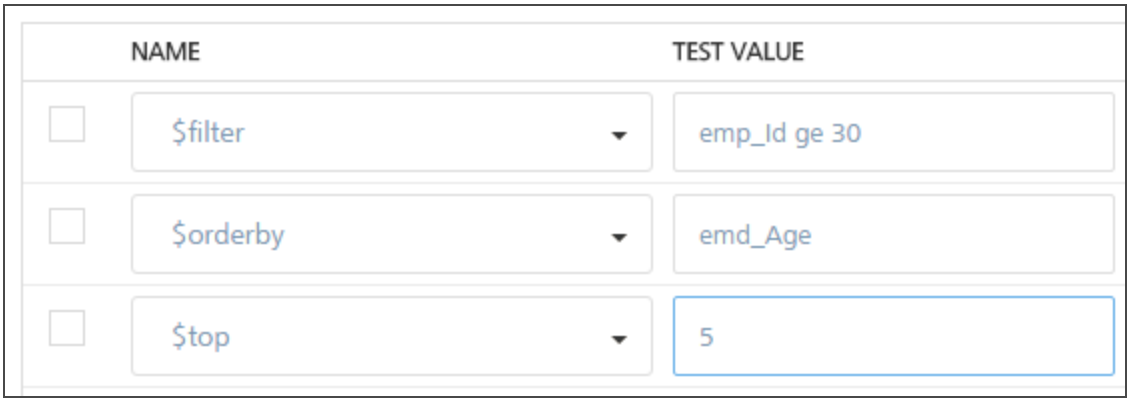


1. The **NAME** field in the **Request Input** is prepopulated with ODATA commands.
2. In the **TEST VALUE** filed, enter the query parameter for the selected ODATA command.

   For example (sample employee table), shown below:

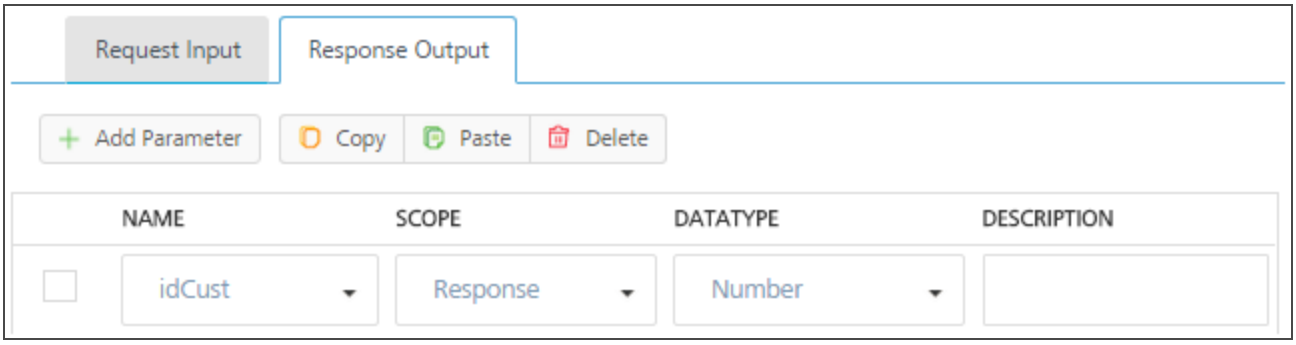
Command Name	Test value for the command	Result
$filter	emp_Id ge 30	Filters and displays data in the table based on age of employes who are older than 30.
$orderby	emp_Age	Arranges data in the table based on employees' age.
$top	5	Displays top five records in the table.
$skip	5	Displays all records in the table except top five records.

For example (sample configuration for ODATA commands), shown below:



3. In the **DEFAULT VALUE**, enter the value if required.
4. In the **DESCRIPTION**, provide the description.

3. To validate the details, click **Fetch Response**. For more information, refer Test a Service Operation. The result of the operation appears.
4. Click **SAVE OPERATION** to save the changes in the read operation.
5. In the **Response Output** tab, configure the fields of the table for displaying the data.



6. To configure the Response Output operation, provide the following details .

Field	Description
Name	The **Name** field in the Response Output tab is pre-populated with database columns
Scope	Select request or session. This field is set to **Request**, by default. ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.
DATA TYPE	The default datatype for the selected column is loaded under **DATATYPE** field.
Description	Enter the description for response output.

7. To validate the details, click **Test**. For more information, refer Test a Service Operation. The result of the operation appears.

8. Click **SAVE OPERATION** to save the changes in the read operation.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

How to Update a Database Record with Update Operation

1. Under **Configured Operations**, hover your cursor over the **Update** operation, click the **Settings** button, and then click **Edit**.

The system displays the selected operation in the edit mode. The update operation has the Request Input tab.

2. The **NAME** field contains primary key of the table. You cannot modify these details.

   The Name column is prepopulated with fields names in the database.

   > ***Note:*** You can add an entry by clicking the **Add Parameter** button if entries for the input and the output tabs do not exist.
   >
   > - To make duplicate entries, select the check box for the entry, click **Copy**, and then click **Paste**.
   >
   > - To delete an entry, select the check box for an entry, and then click the **Delete** button.

3. Update the values in the fields, such as **TEST VALUE**, **DEFAULT VALUE**, and **SCOPE**, if required.

   To validate the details, click **Fetch Response**. The result of the operation appears.

4. Click **SAVE OPERATION** to save the changes in the update operation.

How to Delete a Database Record with Delete Operation

1. Under **Configured Operations**, hover your cursor over the **Delete** operation, click the **Settings** button, and then click **Edit**.

   The system displays the selected operation in the edit mode. The delete operation has the Request Input tab.

   1. Under **Configured Operations**, hover your cursor over the **Delete** operation, click the **Settings** button, and then click **Edit**.

The system displays the selected operation in the edit mode. The delete operation has the Request Input tab.

2. The **NAME** field contains the primary key of the table. You cannot modify these details. The **Request Input** tab contains only the primary key of the table.
3. In the **TEST VALUE** field, enter the valid primary key value.
4. Click **Fetch Response** to validate the details. If the test value matches the primary key in the database, the system deletes the record from the database.
5. Click **SAVE OPERATION** to save the changes in the delete operation.

Create SQL Custom Query

If you want to write your own SQL query in the operation, do the following:

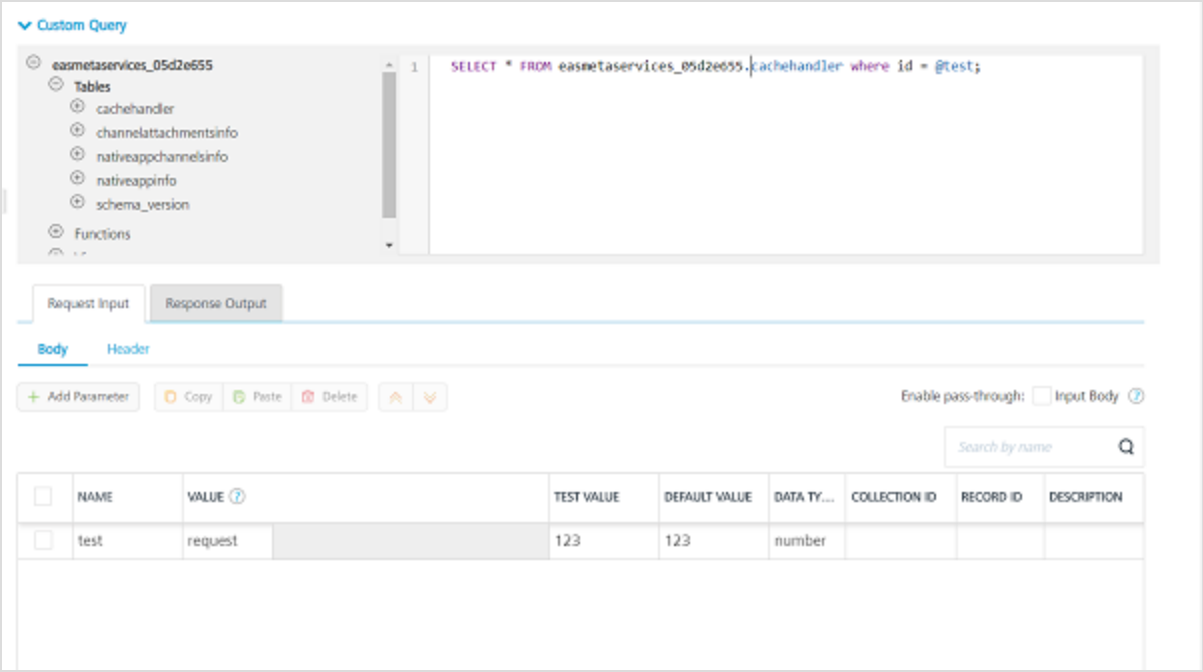
- From the schema list, expand the required schema > CustomQueries.

- Click Add Operation. A new operation named <schema>_CustomQuery is created by default and will be added to the operations list.

Perform the following steps to write a custom query:

> ***Note:*** Currently only MySql, SQL Server, Oracle and PostgreSQL are supported.

- From the operations list, click <schema>_CustomQuery. <schema>_CustomQuery is a default name given to the custom query operation, and it is advised to rename the operation name immediately after you create it. Expand the operation.
- Provide the details in the Operation tab as mentioned earlier.
- To write the custom query, expand the **Custom Query** section. This section contains the read only list of tables and field names in the left pane, and a SQL statement box in the right pane.



- In the right pane, write the required SQL query.
- To include any input parameters in statement, mark them with @. For example, @inputparam1. Here is the sample code for it.

```
Select * from KONYUNITTESTDATABASE.products where LASTUPDATETIME
> @timestamp and SOFTDELETEFLAG = @boolean limit 100;
```

> ***Important:*** Make sure that you always provide a limit in the query to prevent server overload.

- Create the corresponding input parameter in the Request Input section.
- If you want to define any output parameters, mention the same in the Response Output section.

> ***Note:*** If there are no output parameters mentioned, then you will get the query result set as is from the adapter.

- Click Save and Fetch Response to view the response from the database based on the SQL statement written.

> ***Important:*** If any error occurs in the database due to the SQL query, Kony Fabric will return the same error message which it receives from the database. If the error message contains any sensitive information, it is the responsibility of the developer to handle the sensitive information and display a generic error message.

Limitations

- You can write only one query in an operation.
- For now, this feature supports only Select query. Support for Create, Update, and Delete will be available in later releases.
- Record and Collection data types are not supported for a custom query.
- Blob is not supported, so it is the application developer's responsibility to specify the columns (excluding blob/binary) in Select statement or to define the output response to exclude blob.

21.5.11.3 Database Adapter Limitations

Following are the limitations to use the Kony Fabric Database Adapter:

- Currently Kony Fabric database adapter supports MySQL, Oracle, MS SQL, and PostgreSQL databases.
- Table Valued MSSQL server functions are not supported from Kony Database adapter
- Execution of SQL commands using database adapter is not supported - for example, insert, select, and alter.
- Using the database adapter:
    - An admin can only alter the data in tables - Data Manipulation Language (DML).
    - An admin cannot alter the structure of the table - Data Definition Language (DDL). For example, an admin cannot add a column in the table.
- Any CRUD operation can be performed on only one record at a time based on Primary Key.
- For related tables, you can perform hierarchical Create/update/read.
    - `$top` in case of Oracle DB follows the ordering imposed by the backend, unless a `$orderby` is specified.
    - `$expand` on self-referenced table is not supported.
    - Kony supports `eq`, `ne`, `lt`, `le`, `gt`, `ge`, and `or` operators with the `$filter` ODATA parameter.
        - The `$expand` fails, if any related table has binary (bolb (or) bit) columns where MySQL itself fails to convert the binary data into a valid JSON.
        - If the child payload is higher than 1024 bytes, MySQL truncates the excess data with its default properties on `group_concat()`. To resolve the issue, you must configure a higher value than the max payload size for `group_concat_max_len` property.

- With the `$expand` ODATA parameter, the database adapter supports multi-level query for related tables. This is applicable for objects services as well.

  > ***Important:*** For Kony Fabric V8 SP2 or below versions, the following are the limitations with `$expand`:
  >
  > - For SQL Server, Oracle, and PostgreSQL databases, `$expand` is limited to one level.
  >
  > - For using the `$expand` ODATA parameter, MySQL version must be 5.7.7+.
  >
  > - For SQL Server, the `$expand` with a column of type `timestamp` is not supported.
  >
  > - For PostgreSQL, the `$expand` with a column of type `Boolean` is not supported.

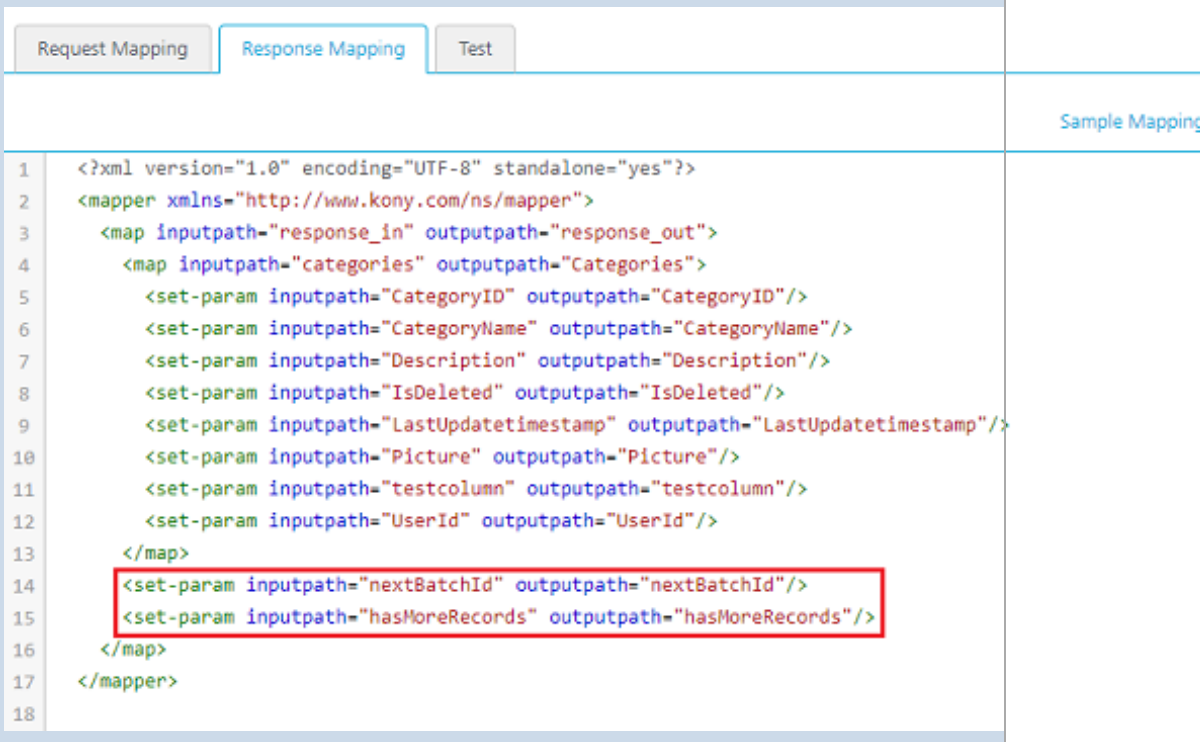
> ***Note:*** Open Data Protocol (OData) is an open protocol to allow the creation and consumption of queryable and interoperable RESTful APIs in a simple and standard way. For more details, refer to http://www.odata.org/

- The following are the limitations of batching:
  - For Batching to work, specify the **ChangeTrackingColumn** because batching is done with respect to the ChangeTrackingColumn to track delta/updated records.
  - $batchsize and $batchid are supported only for MySQL database.
  - You cannot download the binary columns.
  - The $expand along with batching, the JSONExpand is only supported.
  - Download throws an error if there are empty record in the backend.

- If you want specific columns to be selected using $select, you must have the ChangeTrackingColumn in $select.
- You must not give **$top/$skip** pair along with the **$batchsize/$batchid** pair.

Note:

- For the batching to work, in the RDBMS SDO flow, add the following in the **Response Mapping**:
  - <set-param inputpath="nextBatchId" outputpath="nextBatchId"/>
  - <set-param inputpath="hasMoreRecords" outputpath="hasMoreRecords"/>



Note: If you specify **$orderby**, $orderby with ChangeTrackingColumn is executed first followed by your $orderby value.

> ***Note:*** The number of records downloaded can be more than the batch size. This is because the records are downloaded with a **timestamp** equal to the last record's timestamp of the current batch.

21.5.12 MongoDB Data Adapter

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

With Kony Fabric MongoDB database adapter, you can connect to your own MongoDB database as an endpoint. For example, you can use MongoDB data adapter to implement a data store that provides high performance, high availability, and automatic scaling such as e-commerce product catalog, blogs and content management, and mobile and social networking sites.

After you configure the MongoDB document database adapter in Kony Fabric Console, you can create, read, update and delete (CRUD operations) on MongoDB collections and documents.

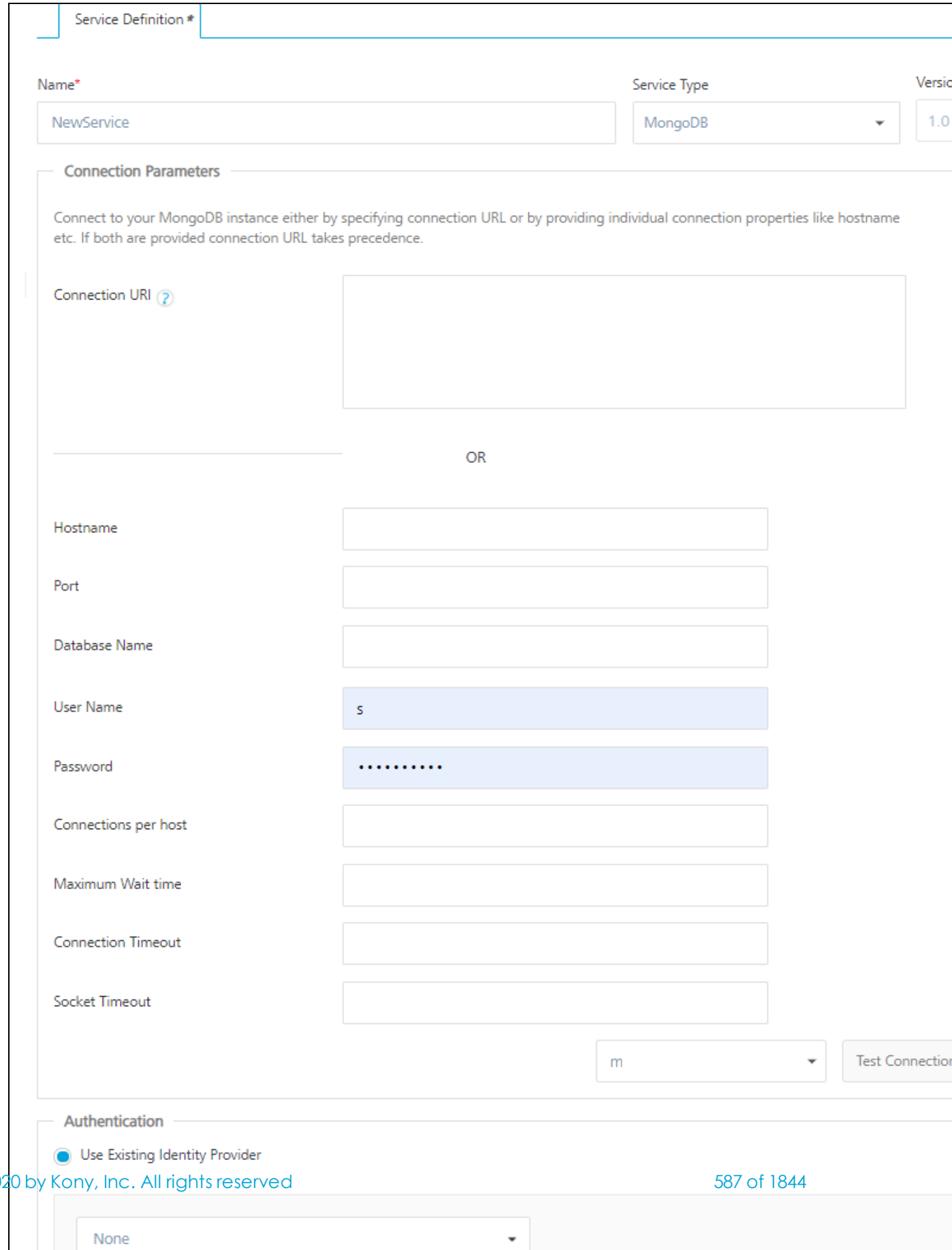
Perform the following steps to configure the MongoDB database adapter with Kony Fabric:

- Configure a Service Definition for the MongoDB Adapter
- Create and Configure Operations for the MongoDB Adapter

21.5.12.1 Configure MongoDB End-point Adapter

To configure the MongoDB Adapter in the Integration service definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **MongoDB**. The following sections are displayed:
    - Connection Parameters
    - Authentication
    - Advanced



3. In **Connection Parameters**, you can either provide **Connection URI** or provide **Connection Properties**.

   - **Connection URI**: If you want to provide Connection URI, you must construct the URI and create a MongoDB client instance to connect to the database server.
     Connection URI format:

     ```
     mongodb://[username:password@]host1[:port1][,host2
     [:port2],...[,hostN[:portN]]][/[database][?options]].
     ```

     The elements required to construct a Connection URI are as follows:

Element	Description
mongodb://	This is a mandatory prefix required to identify that this string is in standard connection format. Use a connection string prefix of mongodb+srv: rather than the standard mongodb: to leverage the DNS seedlist. The +srv is added as an indicator to the client that the hostname that follows it corresponds to a DNS SRV record. The driver or mongo shell will query the DNS for the record to determine the hosts that are running the MongoDB instances.
username:password@	This field is optional. Use this to add authentication to access the MongoDB database.
host[:port]	Hostname, and port number. The port number is optional
/database	Name of the database. This is optional.
?<options>	String that specifies the connection specific options as <name>=<value> pairs.

- **Connection Properties**: If you want to provide Connection Properties, provide the following details:

Parameter	Description
Hostname	Database connection URL.
Port	The port number of the MongoDb server to which you want to connect.
Database Name	Name of the database name.
User Name	User ID for the connection URL
Password	User Password
Connections per host	Maximum number of connections allowed per host.
Maximum Wait time	Number in milliseconds that a thread may wait for a connection to become available.
Connection Timeout	Number of milliseconds the driver will wait before a new connection attempt is aborted.
Socket Timeout	Number of milliseconds a send or receive on a socket can take before timeout.

4. To test the database connection, click **Test Connection**.

   If the entered details are correct, the system displays the message: Valid Database connection details.

> ***Important:*** If your database is configured with a proxy server, you must select an **environment** and then click **Test Connection** to test the database connectivity. The environment should be `=> V8.3`.
>
> For example, you have the Kony Fabric Console installed on one machine, and the Runtime and Database servers installed on another machine. When you create an integration service of type MongoDB Database, the Console must be established with a VPN connection to the MongoDB Database server. So that, when you test the Database connection, the test case will be successful. You can do this by selecting the correct environment for your Runtime Server which will ensure a VPN connection between the Console and the Runtime Server and test the database. If the entered details are correct, the system displays the message: Valid Database connection details.

5. In the **Authentication** section, you can select an identity provider from **Use Existing Identity Provider**. This drop-down lists the identity providers created for MongoDB data adapter in the Identity page. If you select any identity provider, you have to enter valid login credentials to access the database. Click **Test Login** and enter the credentials to verify the authentication service.

   > ***Note:*** The Authentication section is optional.

6. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Custom Code	To specify a JAR associated to the service, select one from the **Select Existing JAR** drop-down menu or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration.
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration.

> ***Note:*** The Advanced section is optional.

7. Enter the **Description** for the service.
8. Click **SAVE** or **SAVE & ADD OPERATION** to save your service definition.

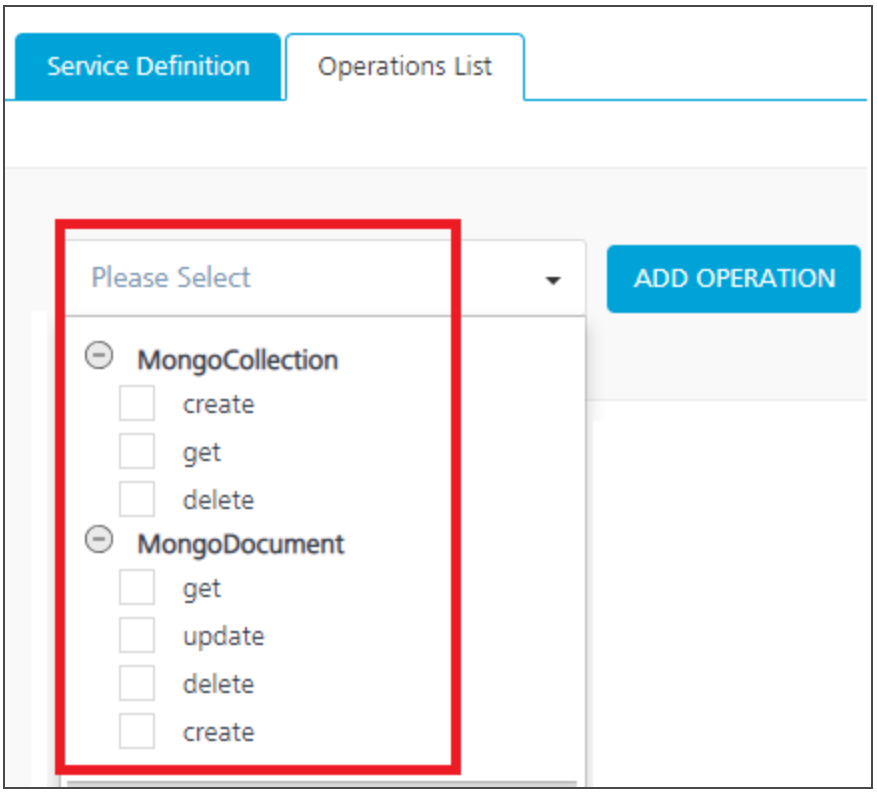
21.5.12.2 Create and Configure Operations for MongoDB Adapter

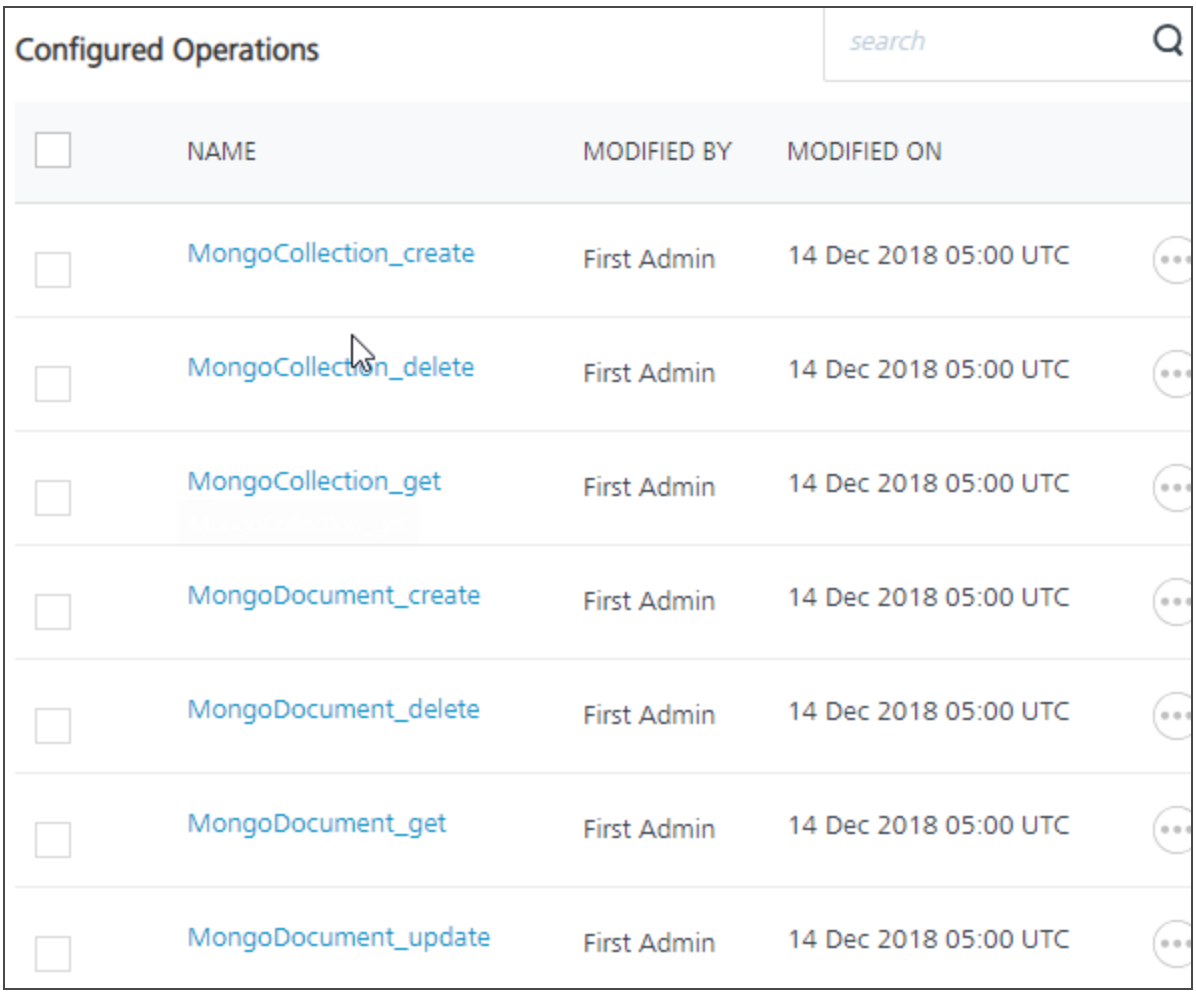
The **Operations List** tab appears only after you save the service definition. Alternatively, you can also click **Add > Add New Operation** to create or configure operations.

Create

To create an operation, follow these steps:

1. Select the check boxes for operations available for the **MongoCollection** and **MongoDocument** entities.
2. Click **Add Operation**. The selected operation is added to the **Configured Operations** list. The default name format of a database operation is `<MongoCollection/MongoDocument>_<operation_name>` and these names are auto-generated.
   For example, `MongoCollection_create` and `MongoDocument_create`.





Configure

After you create operations, they are configured with request and response parameters by default.

The following table details pre-configured parameters for request and response operations :

MongoDB Operations	Request Parameters	Response Parameters
MongoCollection_Create	• CollectionName	• Collection name
MongoCollection_Read	NA	• Collection names
MongoCollection_Delete	NA	NA

MongoDB Operations	Request Parameters	Response Parameters
MongoDocument_Create	• Record • CollectionName	• MongoDocument • DocumentID • Record • Collection name
MongoDocument_Update	• DocumentID • Record • Collection name	• MongoDocument • DocumentID • Record • Collection name
MongoDocument_Get	• OData Query	• MongoDocument • DocumentID • Record • Collection name
MongoDocument_Delete	• DocumentID • Collection name	NA

21.5.12.3 Key Terminology and Concepts of MongoDB

MongoDB stores BSON documents, for example, data records in collections; the collections in databases.

Refer the following table for key terminology.

Terminology	Concept
Database	In MongoDB, a database holds collections of documents.
Collection	A collection is analogous to a **table** of an RDBMS. A collection may store any number of documents.
Document/Object	A document in MongoDB is analogous to a record in RDBMS and it is a data structure composed of field and value pairs have the following structure: `{ field1: value1, field2: value2, ... fieldN: valueN }` MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.
Field	A name-value pair in a document. A document has zero or more fields. Fields are analogous to **columns** in relational databases.

21.5.12.4 Advantages of MongoDB Data adapter

Following are the advantages of using MongoDB Adapter:

- Admins can connect to the given database.
- Admins can manage the databases using CRUD operations.
- Data types: All major data types are supported.
- Kony supports three ODATA parameters for the read operation on MongoDB documents such as `$filter`, `$top`, and `$skip`.

21.5.12.5 Limitations

Following are the limitations to use the Kony Fabric MongoDB Adapter:

- Any CRUD operation which involves bulk activity using multiple Document IDs cannot be performed through this adapter.

21.5.13 RAML Adapter

To integrate a RAML service in Kony Fabric, a RAML file or Zip (with all dependent files) must be created, which defines all the API definitions and schemas with properties in the project. When a Kony Fabric user creates a service from Kony Fabric Console, the system retrieves metadata from a RAML file and displays all APIs of a RAML file.

Kony Fabric parses a RAML file and exposes all the endpoints through the integration service.

21.5.13.1 Configure RAML Endpoint Adapter

To configure RAML service in the Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **RAML**.
3. Provide the following details to create the RAML service:

Field	Property
Connection Parameters	Click **Upload** and select a RAML file from your local machine. The system adds your main RAML file to the console. The system displays the added RAML file's name under the **Connection Parameters** section.
Authentication	Select an identity provider from the list to link your service.

Field	Property
Web Service Authentication	Select one of the following modes: • **None**: Select the option if you do not want to provide any authentication for the service. • **Basic**: Provide User ID and Password if the external Web service requires form or basic authentication. • **NTLM**: Your service follows the NT LAN Manager authentication process. You are required to provide the User ID, Password, NTLM Host and NTLM Domain.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. This step allows you to further filter the data sent to the back end. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device.

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration. ***Note:*** Enable API throttling in a clustered environment by configuring the KONY_SERVER_NUMBER_OF_NODES property in the **server_configuration** table available in Admin database. This property indicates the number of nodes configured in the cluster. The default value is 1.

Note: All options in the Advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.
6. To enable the proxy, select the **Use proxy from settings** check box. By default, the check box is cleared. The Use proxy from settings check box dims when no proxy is configured under the Settings > Proxy.
7. Click **Save** to save your service definition.

21.5.13.2 Create Operations for RAML

The **Operations List** tab appears only after the service definition is saved.

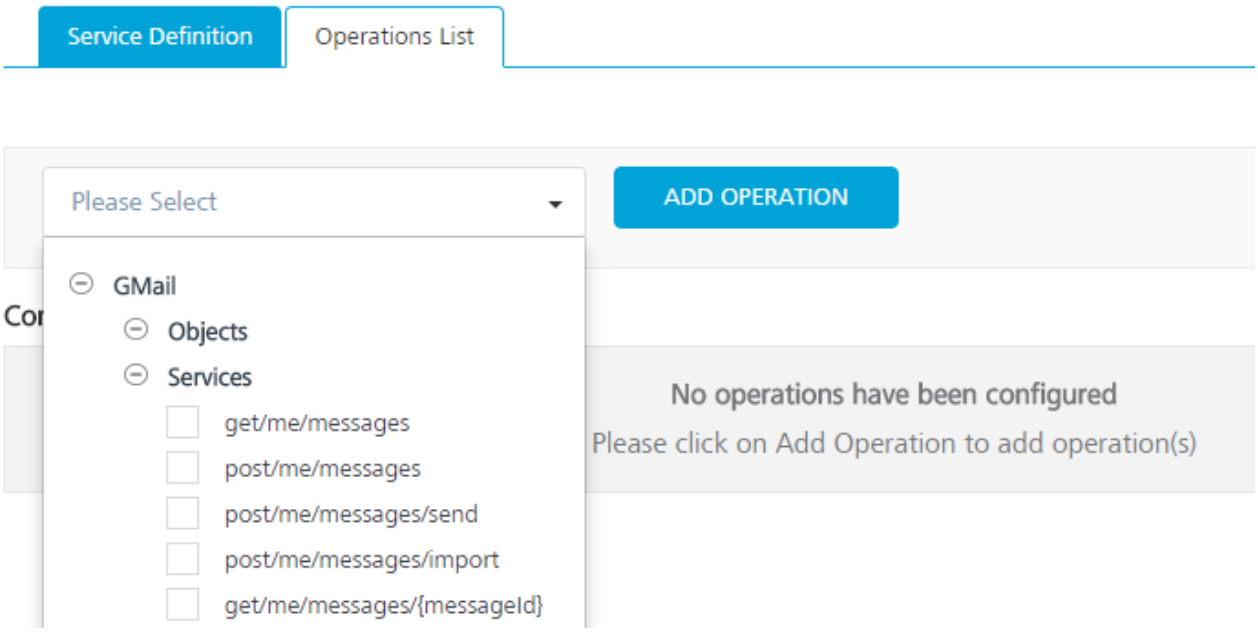
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



2. Under **Operations List** tab, **Please Select** drop-down lists all the supported operations based on the uploaded RAML file..
   i. Expand an operation.
   ii. Under **Objects and Services**, select the required check boxes.

3. Click **ADD OPERATION**. The system adds your operation to the Operations List page.
4. To configure an operation, click on the required service under **Operations List** and provide the following details:

Field	Description
Name	It is prepopulated with the operation name. You can change the name if required.
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

5. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation - Preprocessor and Postprocessor (for Java and JavaScript)	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Additional configuration properties (timeout, cachable, unescape embedded xml in response, response encoding, number of connection retries allows you to configure service call time out cache response
Front End API	It allows you map your endpoint/back-end URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

21.5.13.3 Configure Request Operation for RAML

1. In the **Request Input** tab, provide the following details:

> ***Note:*** Input and Output must be defined in the RAML file only.

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select Request or Session. It is set to **Request** by default. • **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. • **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.

Field	Description
Data Type	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Date** - Date format ▪ If data type is string, then the options in the Format Type are Currency, Number and Date. ▪ If the data type is number, then the options in the Format Type are Currency and Date. ▪ If the data type is boolean, then the options in the Format Type and Format Value text box are disabled. ***Note:*** Currently the date data type is not supported. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number.
Record ID	Enter an ID.
Description	Provide a suitable description.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

21.5.13.4 Configure Response Operation for RAML

1. Click the **Response Output** tab to view the output test values, such as name, scope, data type.

   > ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.
3. Click **SAVE OPERATION**.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.14 Salesforce Adapter

21.5.14.1 Configure Salesforce Endpoint Adapter

To configure Salesforce service in the Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service.
2. From the **Service Type** list, select **Salesforce**.
3. Provide the following details to create the Salesforce service:

Field	Description
Authentication	• **Use Existing Identity Provider** - to select an identity provider. This drop-down lists all identity providers only if you have already created identity providers for SAP in the Identity page. Fill in the details for the following fields: i. From the **Select Identity Provider** list, select your Salesforce identity. The details for the selected identity are displayed in the **Endpoint URL** text box. You cannot modify these details. ii. Under the **User ID** and **Password**, provide valid log-in credentials that you created while registering with Salesforce services. • **Specify Login Endpoint**- to configure a new endpoint. Fill in the details for the following fields: i. In the **Endpoint URL**, enter the URL - for example, `https://login.salesforce.com/services/oauth2/token`. ii. In the **Client ID** text box, enter a valid client id. iii. In the **Client Secret** text box, enter a valid client secret. iv. In the **User ID** text box, enter a valid user ID. v. In the **Password** text box, enter a valid password.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section:

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. This step allows you to further filter the data sent to the back end. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appear when a service is used from a device.

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-premises.

Note: All the fields in the Advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.

6. To enable the proxy, select the **Use proxy from settings** check box. By default, the check box is cleared. The Use proxy from settings check box dims when no proxy is configured under the Settings > Proxy.

7. Click **Save** to save your service definition.

21.5.14.2 Create Operations for Salesforce

The **Operations List** tab appears only after the service definition is saved.

> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.

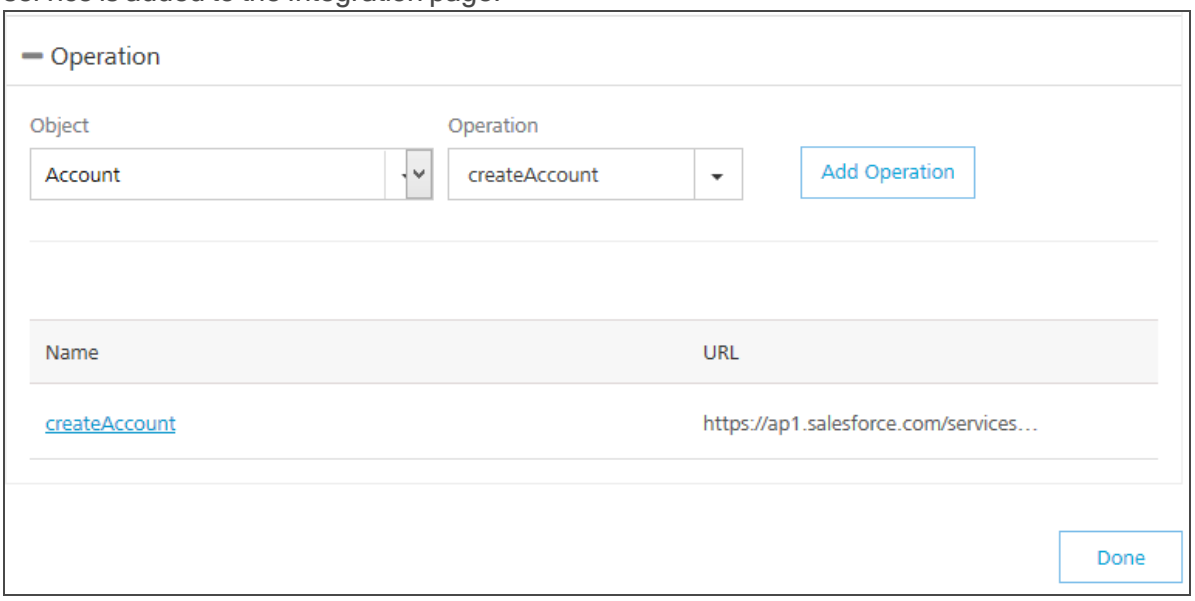
2. Select an object from the **Object** list that is auto-populated with all the existing Salesforce objects.

   > ***Note:*** If you provide incorrect Salesforce endpoint details, the **Object** list will contain only _ *Login* object.

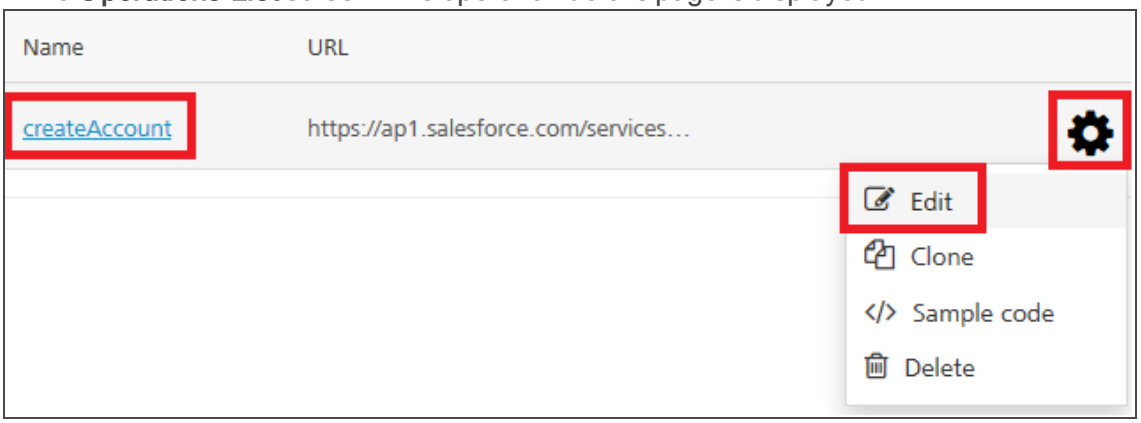
3. Based on the object added in the previous step, the **Operation Name** is listed with operations. Select the check boxes for required operations.

4. Based on the object selected, a list of operations is displayed in the dropdown. Select the required operations and click **Add Operation**.

5. The selected operations are added to the Operations List section and the new Salesforce service is added to the Integration page.



6. To edit an operation, either click on the required operation name or click **Edit** from the **Settings** in the **Operations List** screen. The operation details page is displayed.



7. To configure an operation, provide the following details:

Field	Description
Name	Enter a unique name for your operation.
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.
Operation Path	Modify the path if required. ***Note:*** If you provide incorrect Salesforce endpoint details, the **Object** list will contain only *_Login* object.

8. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation	You can add pre and post processing logic to services to modify the request inputs. When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor.
Properties	Additional configuration properties (timeout, cachable, unescape embedded xml in response, response encoding, number of connection retries allows you to configure service call time out cache response
Front-end API	It allows you map your endpoint (or) backend URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the Advanced section are optional.

21.5.14.3 Configure Request Operation for Salesforce

Integration services accept only `form-url-encoded` inputs for all the input parameters provided in the service input parameters (request input).

You can perform the following actions in Request Input tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .
4. To configure the **Request Input** tab, provide the following details:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Test Value	Enter a value. A test value is used for testing the service.
Default Value	Enter the value, if required. The default value will be used if the test value is empty.
Scope	Select Request or Session. It is set to **Request** by default. • **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. • **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.
Datatype	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Date** - Date format ▪ If data type is string, then the options in the Format Type are Currency, Number and Date. ▪ If the data type is number, then the options in the Format Type are Currency and Date. ▪ If the data type is boolean, then the options in the Format Type and Format Value text box are disabled. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.

Field	Description
Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.

5. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.

21.5.14.4 Configure Response Operation for Salesforce

1. Click the **Output** tab, and enter the values for required fields such as ID, scope, data type, collection ID, record ID, format and format value.

   > ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

2. To validate the operation details, click **Save and Test**. For more details, refer to Test a Service Operation.
3. Click **Save Operation** to save the operation. The system displays the **Operation** section for your service.
4. Click **Done** to navigate to the **Integration** page.

   > ***Note:*** To use an existing integration service, refer to How to Use an Existing Integration Service.

   > ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.14.5 How to Use Log-in Endpoint with Different Credentials for Design Time and Runtime

If the service is using a log-in endpoint, ensure that you specify the same set of credentials (Client ID, Client Secret, User ID, Password) for design time and run-time.

If the log-in endpoint credentials are different for design time and run-time, the system throws an error while accessing a service from an app.

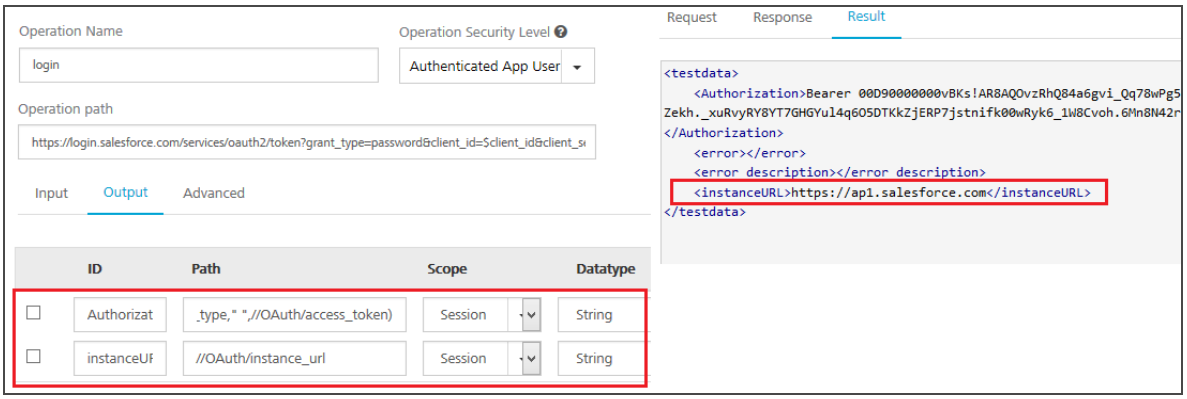
Error 401: Request Unsuccessful Server responded with 401.

If you want to the use log-in endpoint with different credentials for design time and run-time, then parametrize instance URL in the operation path, and send that URL from an app.

For example:

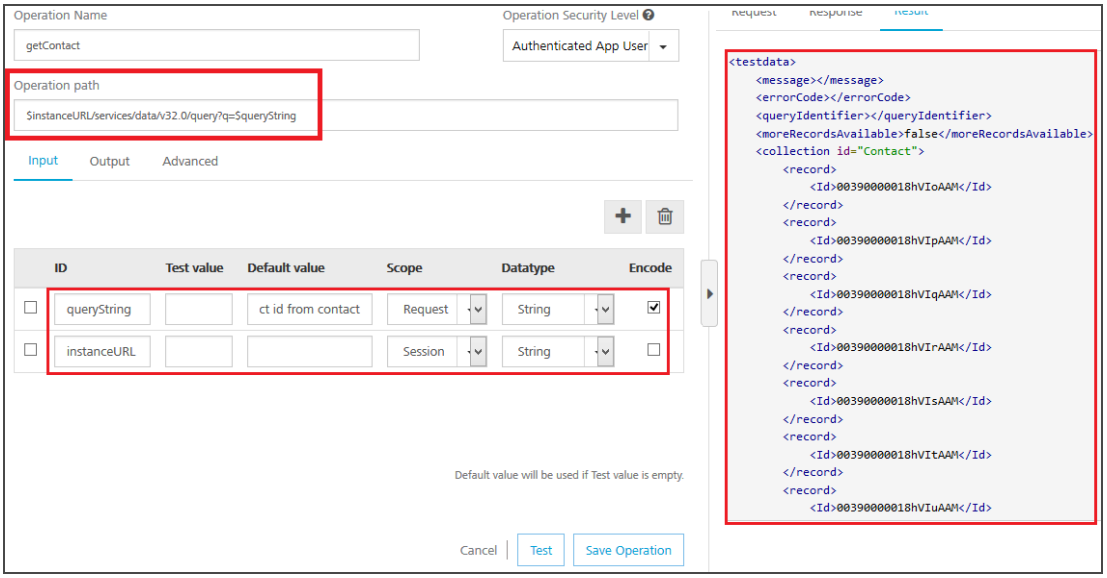
To parametrize an instance URL, follow these steps:

1. In the **login** operation, click **Output** tab, and make the following changes:



ID	Path	Scope
Authorization	concat(//OAuth/token_type," ",//OAuth/access_token	session
instanceURL	//OAuth/instance_url	session

2. Click **Test** to view the results.
3. Click **Save Operation** to save the changes.
4. In the **getContact** operation, make the following changes:
   a. In the **Operation Path** text box, change the URL, for example, from `https://ap1.salesforce.com/` to `$<instanceURL>/`.



   b. In the **Input** tab, configure the following fields as shown:

ID	Path	Scope	Encode
queryString	Select ID from contact	session	
instanceURL	//OAuth/instance_url	session	No

5. Click **Test** to fetch the contacts from Salesforce.
6. Click **Save Operation**.

21.5.15 IBM MQ Adapter

IBM MQ server 9.0.0 version is a messaging middleware that simplifies and accelerates the integration of diverse applications and business data across multiple platforms. It uses message queues to facilitate the exchanges of information and offers a single messaging solution for cloud, mobile, Internet of Things (IoT) and on-premises environments. Kony uses the IBM MQ service to secure the message delivery and reduce the risk of data loss.

21.5.15.1 Configure IBM MQ End-point Adapter

To configure your IBM MQ Service in the Integration Service Definition tab, follow these steps:

1. In the **Service Name** text box, enter a unique name for your service.
2. From the **Service Type** list, select **IBM MQ**.
   By default, XML is selected. If you select **IBM MQ**, the **Connection Parameters** section is

displayed.

3. Provide the following details to create an IBM MQ Service:

Fields	Description
Connection Parameters	Fill in the details for the following fields. • **Host**: Enter the IP/host of the IBM queue manager. • **Port**: Enter the MQ port number of the queue manager. • **Channel**: Enter the IBM channel configured for the queue manager. The default channel is `SYSTEM.DEF.SVRCONN`. • **Queue Manager**: Select or type the name of the queue manager that hosts the destination queue. Queue manager is list of queues. • **Destination**: Default destination of the queue inside queue manager or hosted in a queue manager. • **Consumer Timeout (milliseconds)**: Default timeout is 10000 milliseconds. • **Advanced Connection Properties**: Enter the **Connection Mode** in **Advanced Connection Properties** field. You can also enter the user name and secret, if required. `{"connectionMode":"CLIENT", "username":"", "secret" : ""}` • **Test Connection**: Click **Test Connection** to validate the user data. • If your database is configured with a proxy server, you must select an **environment** and then click **Test Connection** to test the database connectivity. The environment should be `=> V8.3`.

Fields	Description
Use Existing Identity Provider	Select **IBM MQ** identity from the list.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Specify JAR	To specify a JAR associated to the service, select one from the **Select Existing IBM JAR** from the list or click **Upload New** to add a new JAR file. **To specify dependent JAR, follow these steps:** Select the **IBM** `JAR` files from the drop-down list, or click **Upload New** to browse the `JAR` file from your local system. You can upload any one of the following JARs to connect the IBM MQ using JMS. • IBM All client JAR(8 MB) - `com.ibm.mq.allclient.jar` ***Important:*** Make sure that the MySQL packet size should be more than 8 MB. By default, the MySQL packet size is 4 MB. If the MySQL packet size is less than 8 MB, an error will appear when the service is used from a device. You can check the error logs and reset the MySQL packet size to more than 8 MB. • Four mandatory JARs - ◦ `com.ibm.mqcore.jar` or `com.ibm.mq.jar` ◦ `com.ibm.mq.headers.jar` ◦ `com.ibm.mq.jmqi.jar` ◦ `com.ibm.mqjms.jar`

Field	Description
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-

5. In the **Description** field, enter the description.
6. Click **SAVE** to save your service definition.

21.5.15.2 Create Operations for IBM MQ Service

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.
2. Select an operation from the drop-down list and click **ADD OPERATION**.
3. Click the configured operation and provide the following details to create an operation.

Field	Description
Name	The operation name appears in the Name field. You can modify the name, if required.
Mapped To	The operation name to which the configured operation is mapped.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default.

4. Click the **Advanced** tab to configure the preprocessor and postprocessor forJava and JavaScript. For more details, refer to Preprocessor and Postprocessor. You can also configure **Server Events** from here. For more details, refer Server Events.

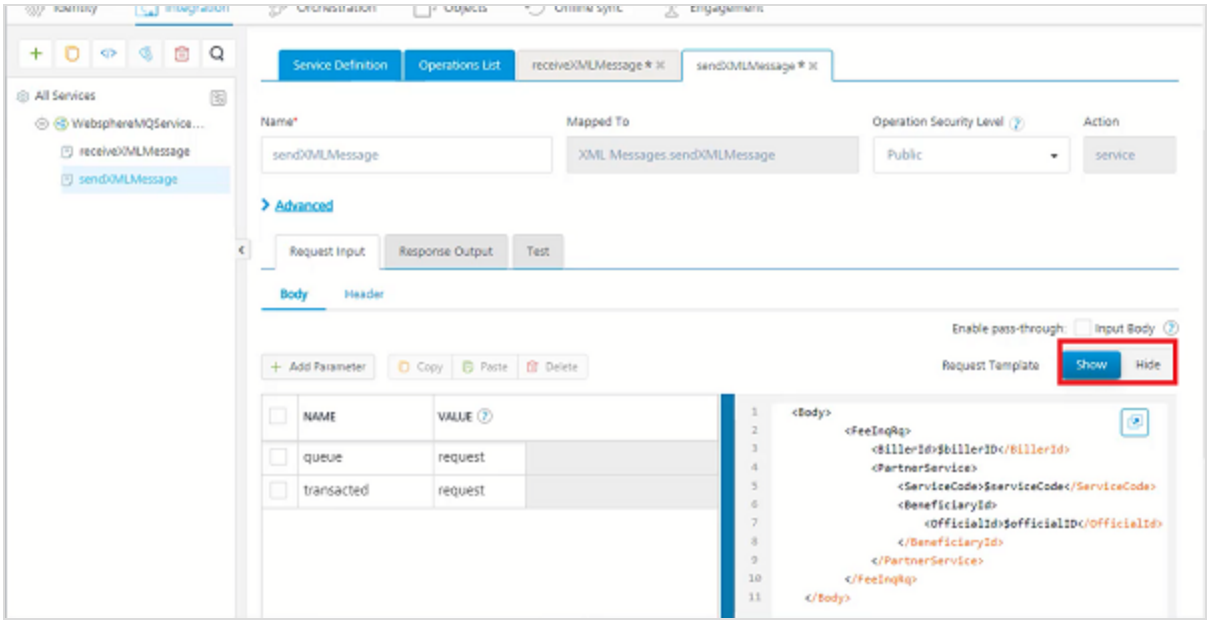
> ***Note:*** All options in the Advanced section are optional.

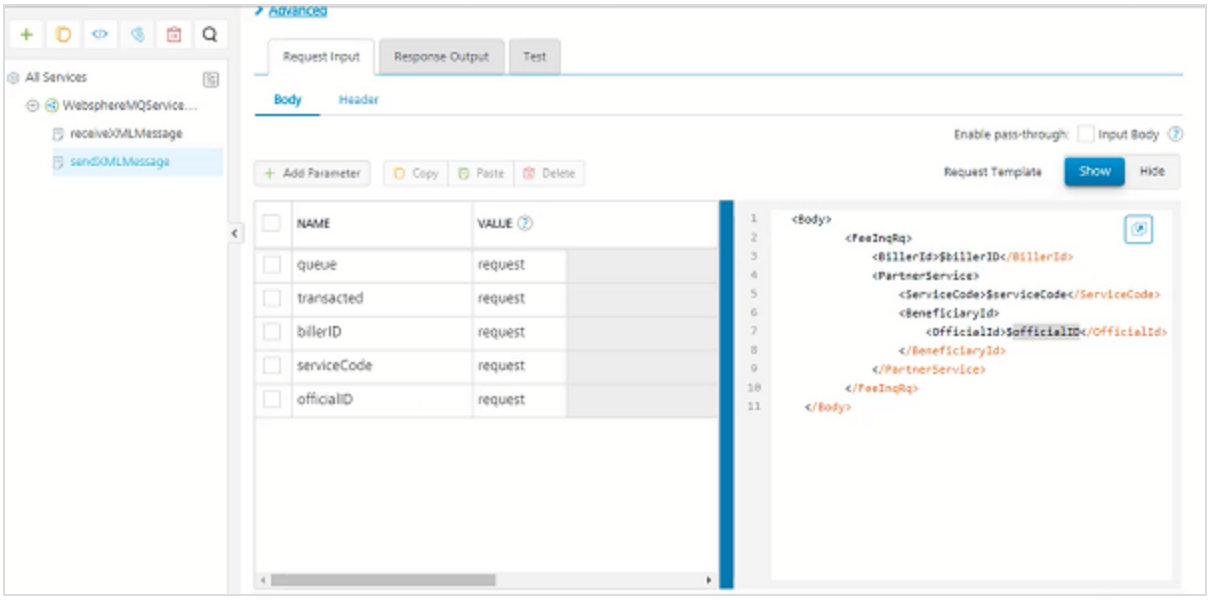
21.5.15.3 Configure Request Operation for IBM MQ

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

You can perform the following actions in the Request Input tab,

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs does not exist).
2. To make duplicate entries, select the check box for the entry, click **Copy** and **Paste**.
3. To delete an entry, select the check box for an entry and click **Delete** .
4. Click **Show** button and enter the message template in the right pane. The message template is used to send a message to IBM MQ server.





5. Add the template parameters as request parameters in the **Request Input** tab.

> ***Note:*** Enter **Queue** and **Transacted** in the request parameters.
>
> - Queue parameter is used to change queue in the queue manager at run time.
> - Transacted parameter accepts **True**or **False** to set JMS message to be in the transacted session scope or not.

6. Provide the following details in the request input parameters tab:

Field	Description
Name	Enter a unique identifier for the request input parameter. Change the identifier, if required.

Field	Description
Value	Three different options are available in Kony Fabric under **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header. Select request or session or Identity. • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Identity**: If this option is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.

Field	Description
DATA TYPE	**Select a data type in the Data Type field.** • **String** - A combination of alphanumeric and special characters. String supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Date** - A value that is defined as date values. • **Boolean** - A value that is true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data or data set.
Record ID	Enter the record ID.
Collection ID	Enter the collection ID.

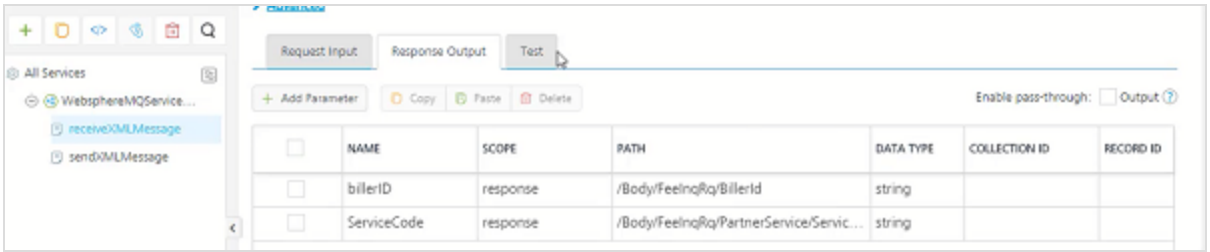
7. Enter the description for the Request Input.
8. You can test a MQ service operation from Kony Fabric. You can refer to Test a Service Operation for the steps to test a service.

21.5.15.4 Configure Response Operation for IBM MQ

Click the **Response Output** tab, and the values in the required fields such as name, scope, path, data type, collection ID, record ID, and description are automatically populated.
The following JMS service output parameters are auto-populated, by default.

- JMSCorrelationID
- JMSDeliveryMode
- JMSDestination
- JMSExpiration
- JMSMessageID
- JMSPriority
- JMSRedelivered
- JMSTimestamp
- JMSReplyTo
- JMSType
- JMSXDeliveryCount
- MessageBody
- JMSREPLYTO
- JMSTYPE
- JMSXDELIVERYCOUNT

You can delete the default parameters and add the custom parameters and **XML** path or **JSON** path to receive the response from IBM MQ server.



Click **SAVE OPERATION** to save the operation. The system updates the operation definition.
If you click **Cancel**, the **Edit Operation** tab closes without saving any information.

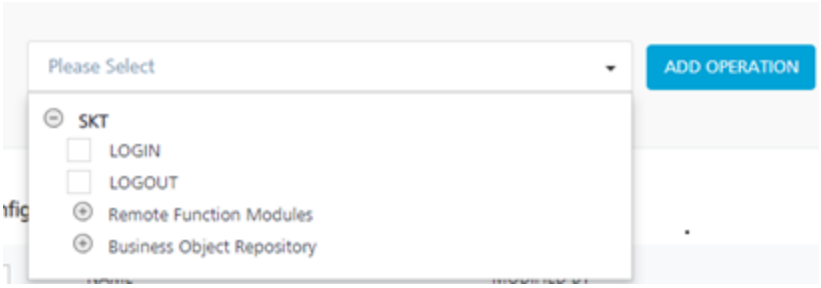
> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.16 SAP JCo Adapter

Kony Fabric allows you to connect and access the BAPI functions inside an ABAP SAP backend via SAP JCo Adapter that was built using sapjco java client. Using SAP JCo adapter, you can explore the BAPI functions inside Business Object Repositories (BOR) and Remote Function Call (RFC) and add them as Kony Fabric operations in the console. When a BAPI function is added as an operation, its corresponding import params and export params are auto-generated as request input params and response output params.

You must load the required **Business Application Programming Interface** (BAPI) functions to define a SAP service. The BAPI files contain the SAP methods and functions. These methods have the logic defined for a service. A BAPI is a function module as it can be invoked from remote programs such as standalone Java programs, web interfaces, and so on.

Login and Log Out - At design time Kony Fabric generates the LOGIN and LOGOUT services along with BOR's and RFM's in the root explore, by default. Usually every BAPI function execution requires username and password to be sent in the request. If you want to skip passing the user name and password, you can use the LOGIN and LOGOUT operations.



Login - The Login operation contains two input params, username and password, used to skip passing the user name and password for subsequent BAPI function calls. When you make a LOGIN call, the credentials you provide are stored into the middleware session.

Log out - The log out operation is used to remove the stored user name and password from the middleware session.

21.5.16.1 Single Sign On (SSO) Login

SAP offers several mechanisms for authenticating users. If there are multiple systems in your system network, using a single sign-on (SSO) environment helps to reduce the number of passwords you must remember. SSO eases user interaction with the available systems, components, and applications . After authentication , users can use the portal to access different systems without repeatedly entering their user information for authentication.

To use SSO login, set the **Login Type** to SSO Login and provide the values for Portal Host and Portal Port (which are mandatory for SSO login) in connection properties. When you add a BAPI function from an SSO enabled backend as an operation (along with BAPI import params list from backend), two additional input params such as username, password, and a header named SSO-TOKEN are generated by Kony Fabric. Change the scope for username, password, and SSO-TOKEN from request to session.

> ***Note:*** In SSO Login type, user must call LOGIN before invoking a BAPI function.

21.5.16.2 Non-SSO Login

To use NON-SSO login, the Login Type must be set to NON-SSO in connection properties. BAPI function of a NON-SSO enabled SAP backend is added as an operation (along with BAPI import param list from backend), two additional input params username and password are generated by Kony Fabric. For every call, the BAPI function is invoked directly by passing the username and password. You can also invoke by changing the scope of username and password input parameters from request to session and using the LOGIN and LOGOUT operations.

This video tutorial walks you through steps to connect an application to your enterprise SAP system using SAP JCo connector.

21.5.16.3 Configure SAP JCo End-point Adapter

To configure your SAP JCO Service in the Integration Service Definition tab, follow these steps:

1. In the **Service Name** field, enter a unique name for your service.
2. From the **Service Type** list, select **SAP JCO**.
   By default, XML is selected. If you select **SAP JCO**, the **Connection Parameters** section

appears.

3. Provide the following details to create a SAP JCO Service:

Fields	Description
Connection Parameters	Fill in the details for the following fields. • **Server Name** -Enter a unique server name to assign for an SAP connection. • **SAP application server host** - Enter the application server IP address. • **SAP system number** - Enter the SAP system number. • **SAP client**- The SAP Client ID. The default is based on the SAP server configuration. • **Logon Language** - SAP Backend logon language. • **Login Type** - Kony Fabric captures log in details for SAP ABAP Server. You can select SSO or non-SSO login. For SSO Login, you must provide portal hostname and port. Kony Fabric generates properties files with SSO details. ◦ In **Login Type**, select an option from the list. • **Non-SSO Login**- To use normal sign on feature, select **Non-SSO Login** and provide user name and password to connect to the server. • **SSO Login** - To use single sign-on feature, select **SSO Login**. • **Logon user** - Enter the user name for the server log on. • **Logon password** - Enter the password for the server log on. • For **SSO-LOGIN**, you must provide the following details: ◦ **Portal Host** - Enter the SAP portal server IP address. ◦ **Portal Port** - Enter the SAP portal server port address • **SAP Router String** - Enter the address of the SAP router that is used to connect to a provider system. • **Advanced Properties** - Specify a JSON variable with SAP Client Properties and corresponding values. For example, to set the pool capacity value, specify the JSON as { 'jco.destination.pool_capacity': <pool capacity> }. Following are few SAP client properties.

SAP Property	Description

Fields	Description
Authentication	Use existing Identity Provider. Select **SAP JCO** Identity provider from the list.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Custom Code	Custom Code enables you to specify dependent JAR. To specify a dependent JAR, select the JAR containing preprocessor or postprocessor libraries from the list, or click **Upload New** to select the JAR file from your local system. This step allows you to further filter the data sent to the backend. ***Important:*** Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. For example, if the JDK version on the machine where Kony Fabric Integration is installed is 1.6, you must use the same JDK version to build your custom jar files. If the JDK version is different, an unsupported class version error will appears when a service is invoked from a device.

Field	Description
API Throttling	• If you want to use **API throttling** in Kony Fabric Console to limit the number of request calls within a minute, do the following: i. In the **Total Rate Limit** text box, enter a value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration. ***Note:*** In case of On-premises, the number of nodes in a cluster environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1. Refer to The Runtime Configuration tab on the Settings screen of App Services. The total throttling limit set in Kony Fabric Console is divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node. This is applicable for Cloud and On-premises.

> ***Note:*** Additional configurations in the advanced section are optional.

5. In the **Description** field, provide a suitable description for the service.
6. Click **Save** to save your service definition.

21.5.16.4 Create Operations for SAP JCO

The **Operations List** tab appears only after the service definition is saved.

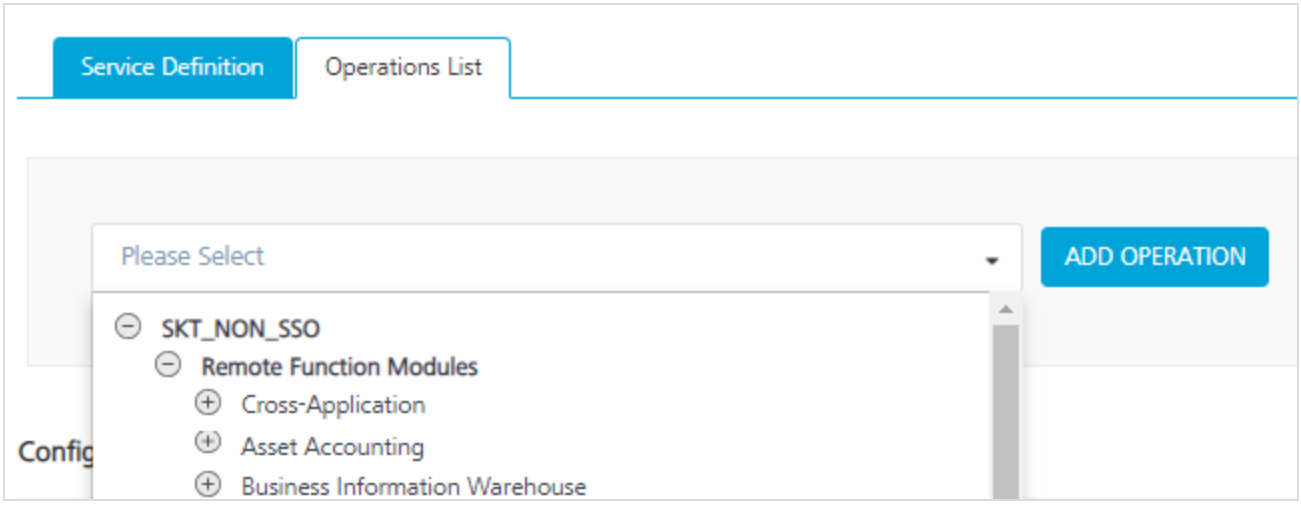
> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears. The **New Operation** window.

To create an operation, follow these steps:

1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.



1. Under the **Operations List** tab, click the list to display all the supported operations based on the uploaded SAP JCO file.
2. Expand an operation and select the required check boxes.
3. Click **ADD OPERATION**. The system adds your operation to the **Operations List** tab.
4. To configure an operation, click on a service under **Operations List** and provide the following details:

Field	Description
Name	It is pre-populated with the operation name. You can change the name if required.
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

5. For additional configurations of request (or) response operations, provide the following details in the **Advanced** section:

Field	Description
Custom Code Invocation - Preprocessor and Postprocessor (for Java and JavaScript)	You can add pre and post processing logic to modify the request inputs of service. When you test, the service details of various stages in the service execution are displayed for better debugging. For more details, refer to Preprocessor and Postprocessor.
Properties	You can configure the additional properties (timeout, cachable, unescape embedded xml in response, response encoding, number of connection retries) for service call time out cache response.
Front End API	You can map your endpoint/back-end URL of an operation to a front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** Additional configurations in the advanced section are optional.

21.5.16.5 Configure Request Operation for SAP JCO

All the request input parameters must be `form-url-encoded` .

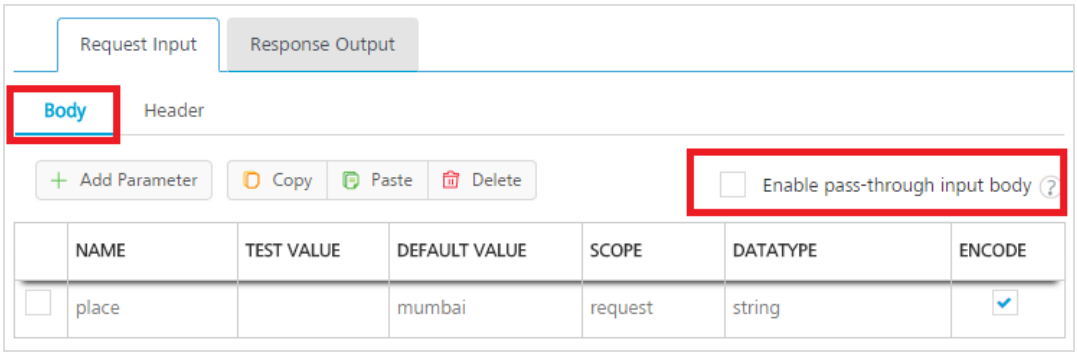
You can perform the following actions in Request Input tab:

1. Click **Add Parameter** to add an entry (if the entries for input and the output tabs do not exist).
2. To make duplicate entries, select the check box for an entry and click **Copy** and then click **Paste**.

3. To delete an entry, select the check box for the entry and click **Delete** .

4. Under the **Body** tab, perform the following actions:

   a. Select the **Enable pass-through input body** check box to forward the body of the client's request to the back end. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.



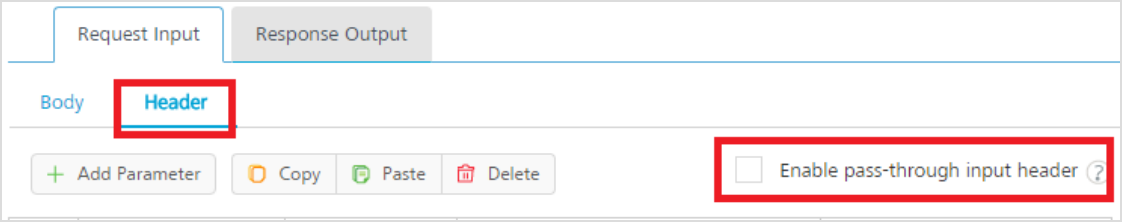
   b. To configure the request input parameters, perform the following actions:

Field	Description
Name	It contains a unique identifier. Change the name if required.

Field	Description
Value	Kony Fabric provides four options while configuring an operation. Dependent identity services are auto populated when you edit this field, These options primarily determine the source of the header value . Select an option from the following options. • If you select **Request**, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. • If you select **Session**, the value of the header is picked from the session context. You cannot edit the Default value and test value. • **Constant** is used to configure the value that is picked and sent to back end by the Integration Server during the run-time. • If **Identity** is selected, you can filter the request parameters based on the response from the identity provider. For more details on configuring the identity filters, refer Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.
Test Value	A test value is used for testing the service. Enter a value.
Default Value	The default value is used if the test value is empty. Enter a value, if required.

Field	Description
DATA TYPE	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Date** - Date format ▪ If data type is string, the options in the format type are Currency, Number, and Date. ▪ If the data type is number, the options in the format type are Currency and Date. ▪ If the data type is boolean, the options in the format type and format value text box are disabled. ***Note:*** Currently the data type date is not supported. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number.
COLLECTION ID	Collection is a group of data, also referred as data set. Enter a collection ID.
RECORD ID	Enter an ID.
DESCRIPTION	Provide a suitable description.

5. Click **Header** tab to provide the custom headers.



You must provide the custom HTTP headers based on the operation. For example, POST or GET.

a. To configure parameters in the client's header, perform the following actions:

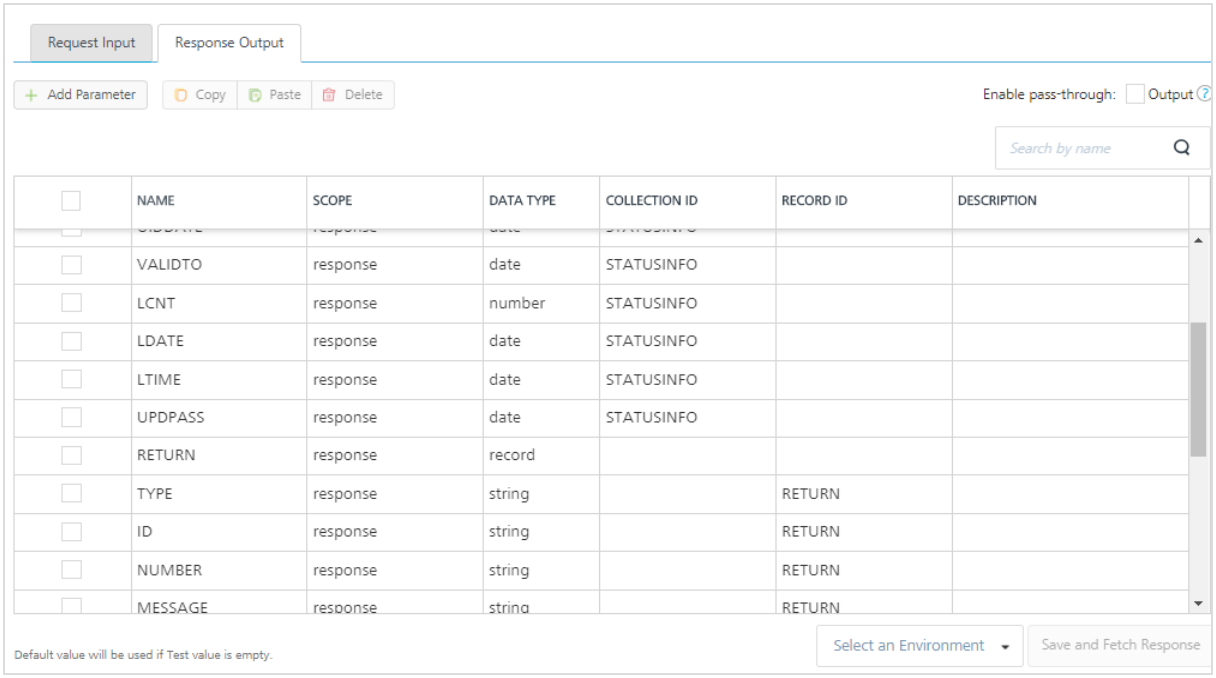
Field	Description
Name	Provide custom HTTP headers required by the external source.
Value	Kony Fabric provides four options while configuring an operation. Dependent identity services are auto populated when you edit this field, These options primarily determine the source of the header value . Select an option from the following options. • If you select **Request**, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. • If you select **Session**, the value of the header is picked from the session context. You cannot edit the Default value and test value. • **Constant** is used to configure the value that is picked and sent to back end by the Integration Server during the run-time. • If **Identity** is selected, you can filter the request parameters based on the response from the identity provider. For more details on configuring the identity filters, refer Enhanced Identity Filters - Integration Services. ***Note:*** The field is set to **Request**, by default.

Field	Description
TEST VALUE	A test value is used for testing the service. Enter a value.
DEFAULT VALUE	The default value will be used if the test value is empty. Change the syntax, if required.
SCOPE	Select request or session. The field is set to **Request**, by default.
Description	Enter the **Description** for the request input.

6. To validate the details, select the environment from the list and click **Save and Fetch Response**. Refer Test a Service Operation for the steps to test a service. The result of the operation appears.

21.5.16.6 Configure Response Operation for SAP JCO

Click the **Response Output** tab to configure the fields of the table for displaying the data.



In the **Response Output** tab, configure the fields of the table for displaying the data:

1. The **Name** field in the Response Output tab is pre-populated with the properties of output schema.

   Enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format, and format value.

   > ***Note:*** If you define parameters inside a record as a session, the session scope will not get reflected for the parameters.

2. To validate the details, click **Test**. Refer Test a Service Operation for the steps to test a service. The result of the operation appears.

3. Click **SAVE OPERATION**.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

> ***Note:*** To use SAP JCO adapter in On-Premises environment, the following artefacts must be placed in the application server -
>
> - `sapjco3.jar`
> - `sapjco.dll` (if Kony Fabric instance is on Windows) or `libso.so` (if Kony Fabric instance is on Linux machine).
> - `SapJCoDestinationProvider.jar`
>
> `sapjco3.jar` and `sapjco.dll/libso.so` artefacts must be downloaded from SAP site.

21.5.17 Open API (Swagger) Adapter

Swagger is a powerful open API specification framework backed by a large ecosystem of tools that helps you design, build, document, and consume your RESTful APIs.

To configure an OpenAPI (Swagger) service in Kony Fabric, a JSON or YAML file(s) (with all dependent OpenAPI (swagger) files or a single specification file) must be created which defines all the APIs and schemas. When an OpenAPI (swagger) service is created and configured, the system retrieves the meta data from the imported single or the dependent JSON or YAML file(s) and displays the APIs of that file.

Kony Fabric parses the JSON or YAML file(s) and exposes all the endpoints through the integration service.

21.5.17.1 Configure Open API (Swagger) End-point Adapter

To configure a Open API (Swagger) adapter in Integration Service Definition tab, follow these steps:

1. In the **Name** field, provide a unique name for your service. When you enter the name, the name is updated for the active service under the **Services** section in the left pane.
2. From the **Service Type** list, select **Open API (Swagger)**.

   > ***Note:*** XML is selected, by default.

3. Provide the following details to create a Open API (Swagger) service.

Fields	Description
Version	Select the version for the service.
Connection Parameters	Click Upload to upload a JSON or YAML file(s) with OpenAPI(swagger) specifications.
Host URL	The host URL available in the uploaded swagger file is displayed here. You can change this URL if required. If you change the host URL, the updated URL will be honored over the URL available in the uploaded file.
Base Path	The base path available in the uploaded swagger file is displayed here. You can change this path if required. If you change the base path, the updated path will be honored over the path available in the uploaded file.
Authentication	Select an existing Identity Provider from the drop-down list.

Fields	Description
Web Service Authentication	Select one of the following modes: • **None**: Select this option if you do not want to provide any authentication for the service. • **Basic**: Provide User ID and Password if the external Web service requires a form or basic authentication. • **NTLM**: Your service follows the NT LAN Manager authentication process. You are required to provide the User ID, Password, NTLM Host, and NTLM Domain.

4. For additional configuration of your service definition, provide the following details in the **Advanced** section.

Field	Description
Custom code	To specify a JAR associated to the service, select one from the **Select Existing JAR** drop-down menu or click **Upload New** to add a new JAR file. Make sure that you upload a custom JAR file that is built on the same JDK version used for installing Kony Fabric Integration. You can download the uploaded jars to your local system.

Field	Description
API Throttling	• If you want to use **API throttling** in Kony Fabric Console, to limit the number of request calls within a minute. do the following: i. In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API. ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • To override throttling from Kony Fabric App Services Console, refer to Override API Throttling Configuration.

> ***Note:*** All options in the Advanced section are optional.

5. Enter the **Description** for the service.
6. Click **SAVE** to save your service definition.

21.5.17.2 Create Operations for Open API (Swagger)

The **Operations List** tab appears only after the service definition is saved.

> ***Note:*** Click **Operations List** tab > **Configure Operation**. The **Configured Operations** list appears.

To create an operation, follow these steps:

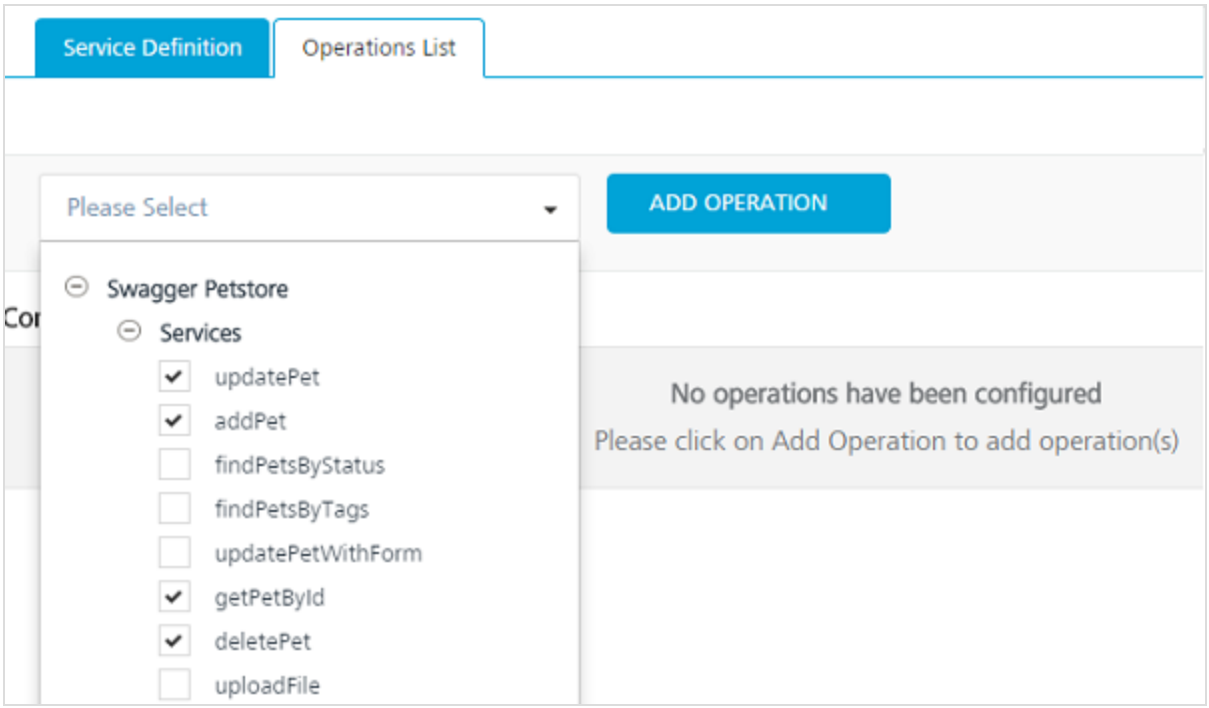
1. Click **SAVE & ADD OPERATION** in your service definition page to save your service definition and display the **NewOperation** tab for adding operations.

   OR

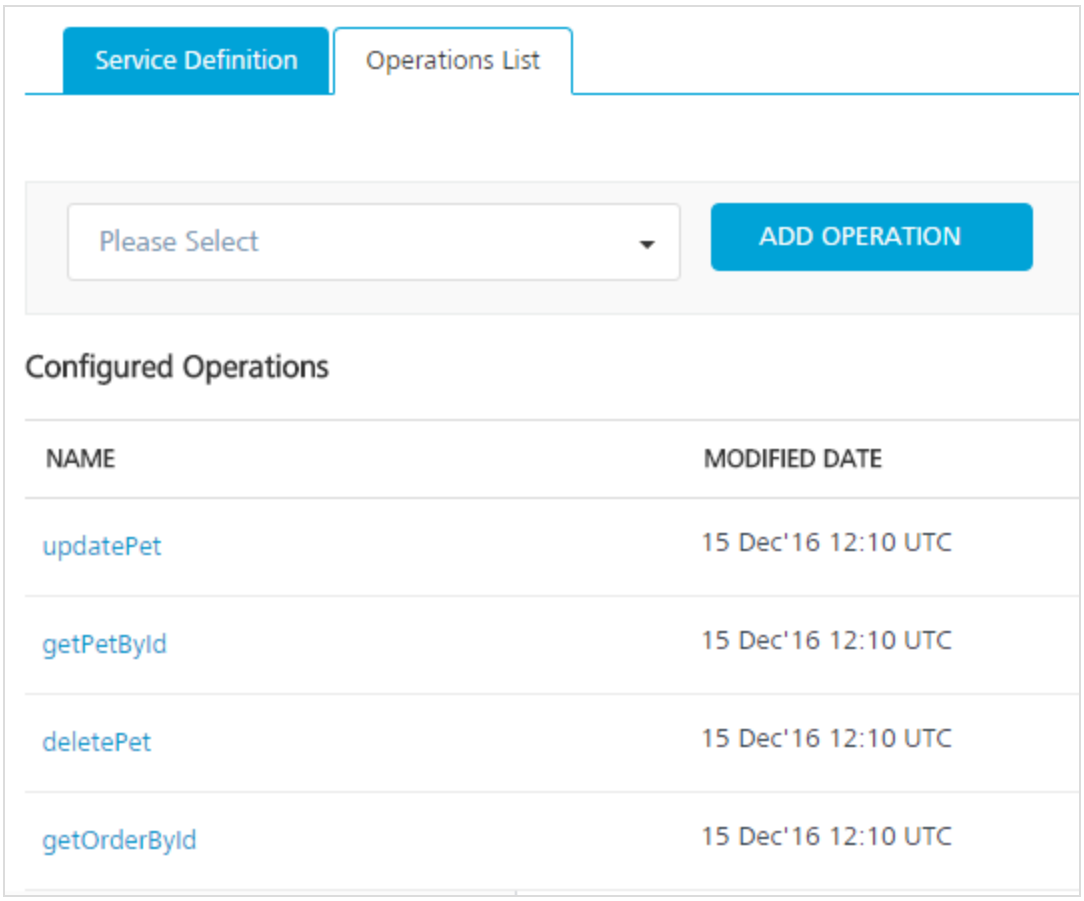
   Click **Add Operation** to add a new operation or from the tree in the left pane, click **Add > Add New Operation**.

2. Under **Operation List** tab. expand the **Please Select** drop-down list. Based on the uploaded JSON or YAML file, all the supported operations will be displayed.

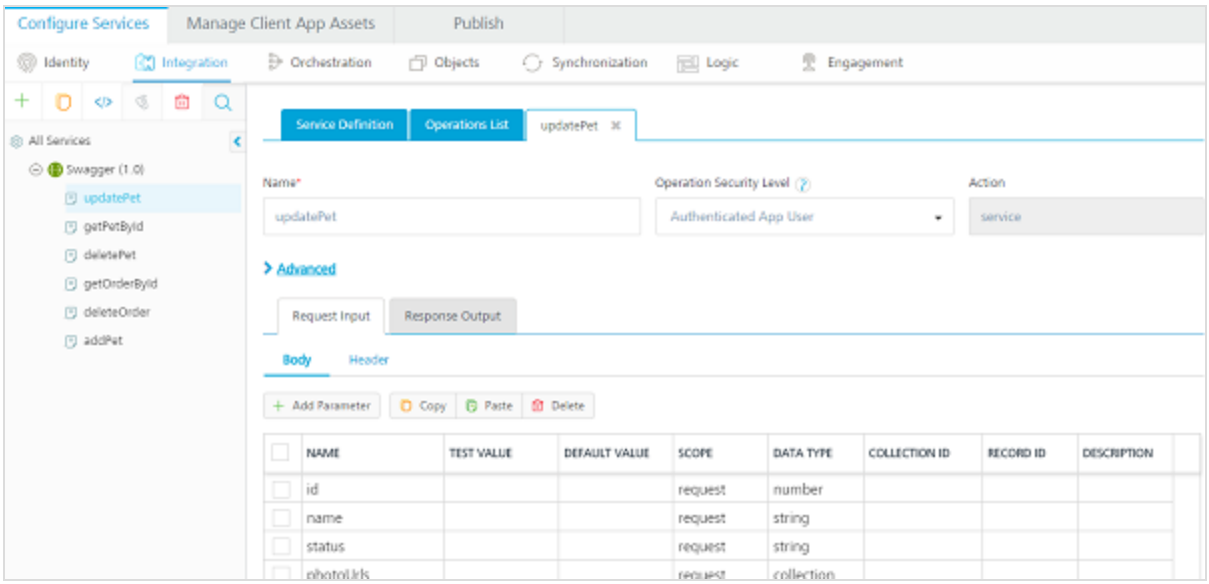
3. Expand the uploaded OpenAPI (swagger) file and under Services, the paths of the uploaded swagger file is exposed. When a particular path / operation is selected, all the fields under the corresponding is populated.



4. Click **ADD OPERATION**. The system adds your operation to the Operations List page.



5. Under **Configured Operations** List, click an operation to view the details of the operation.



6. The system displays the selected operation in the edit mode. Provide the following details to configure the operation.

Field	Description
Name	The operation name appears in the **Name** field. You can modify the name, if required.

Field	Description
Operation Security Level	It specifies how a client must authenticate to invoke this operation. Select one of the following security operations in the **Operation Security Level** field. • **Authenticated App User** - It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App User** - It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required. • **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible. • **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment. ***Note:*** The field is set to Authenticated App User, by default. For more details, refer Security Level.

7. For additional configuration of request (or) response operations, provide the following details in the **Advanced** section.

Custom Code Invocation	When you test, the services details of various stages in the service execution are presented to you for better debugging. All options in the Advanced section are optional. For more details, refer to Preprocessor and Postprocessor. ***Note:*** The Pre and post-processing logic feature is not supported for a Swagger service.
Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

Note: All options in the **Advanced** section for operations are optional.

8. Enter the **Description** for the operation.

21.5.17.3 Configure Request Operation for Open API (Swagger)

The request input parameters are picked from the uploaded OpenAPI definition file and any changes made are ignored. Make sure that the input parameters are `form-url-encoded`.

You can perform the following actions in **Request Input** tab:

1. Under the **Body** tab, perform the following actions:

a. To configure parameters in the clients body, do the following:

Field	Description
Name	Enter the name for the request input parameter. field contains a unique identifier for a parameter.
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
SCOPE	Select request or session. This field is set to **Request**, by default. The default datatype for the selected column is loaded under **DATATYPE** field.
DATA TYPE	Select a data type in the **DATA TYPE** field. • **String** is a combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** a value that can be true or false. • **Number** an integer or a floating number. ***Note:*** For any operation in OpenAPI (Swagger), the binary data is not supported.

Field	Description
Record ID	For nested payloads the **RECORD ID** is populated.
Collection ID	For arrays, the **Collection ID** is displayed.
Description	Enter a description for the Request Input.

2. To validate the provided details, you must test the service operation. You can refer to Test a Service Operation for the steps to test a service.

21.5.17.4 Configure Response Operation for Open API (Swagger)

Click the **Response Output** tab to view the output test values, such as name, scope, and data type.

The **Restrict Parameters to OpenAPI definition** check-box is selected by default. It makes sure that the OpenAPI definition provided is used as the source for the Response output parameters. You cannot change the output parameters.

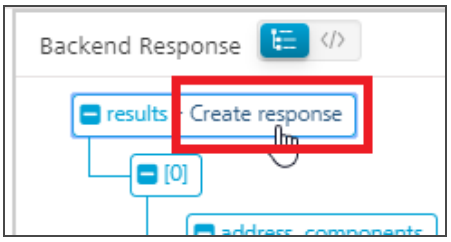
If you want to choose the required output parameters that you want to show in the response manually, you have to deselect **Restrict Parameters to OpenAPI definition** check-box, select the required environment from the **Select Environment** list, and then click **Save and Fetch Response**. The back-end response of the service appears.

In the response, the **Test > Backend Response** pane displays the nodes that will be added to the response output in a **Tree** view. You can do the following to select the required nodes that you want to show in the response from here:

1. Click or hover on the required node which you want to add in the response output.

   The **Create response** button appears next to that node.

2. Click **Create response**.



A new row is created automatically in the Response Output tab and the details of the selected node are added to it.

3. You can also check the JSON response, click Add Parameter, and write the JSON path of the required output parameter manually.

> ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

To validate the details, select an environment from the **Select Environment** list and click **Save and Fetch Response**. The result of the operation appears. For more details, refer Test a Service Operation.

> ***Note:*** For nested payloads, you cannot test the service from Kony Fabric Console in the case of Post/Put methods. You can only send a request through Admin Console or postman.

Click **SAVE OPERATION**.

> ***Note:*** **Following are few limitations to be followed before using OpenAPI (Swagger):**
>
> - Only OpenAPI (Swagger) 2.0 is supported.

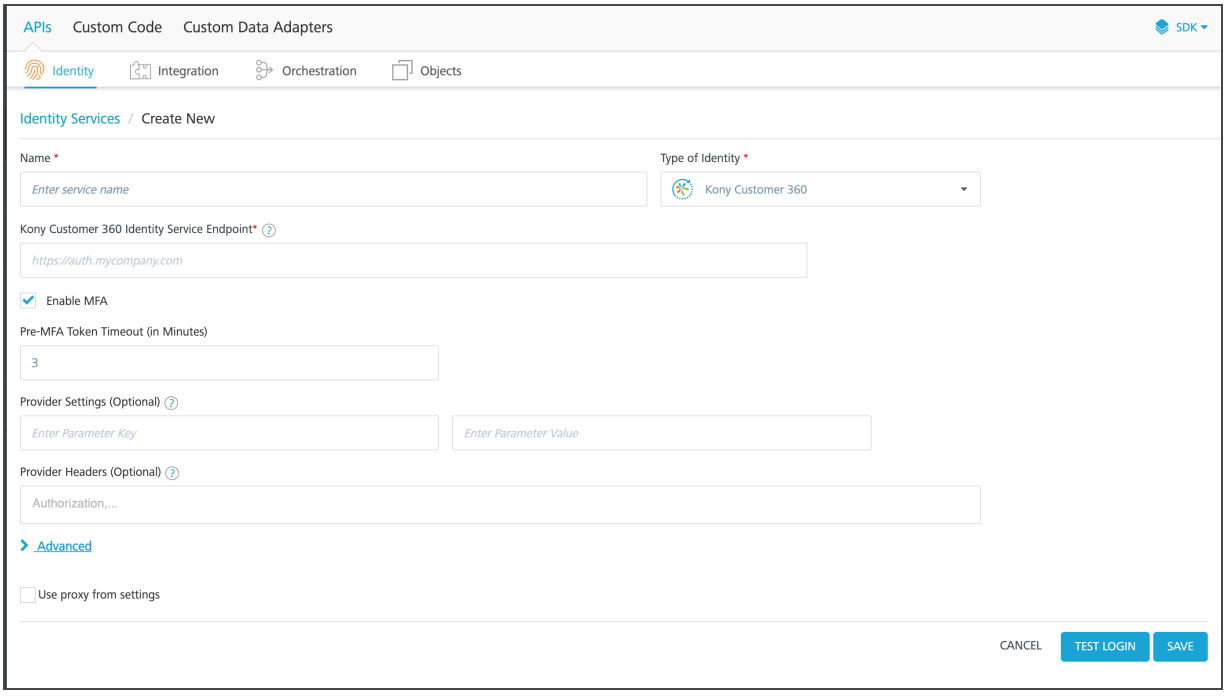
If any scheme is not specified in your swagger file, by default `HTTPS` will be considered.

If an authentication is linked to your swagger file, you must define it at each operation level under security tag.

Note: You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

21.5.18 Kony Customer 360 Adapter

Kony Customer 360 Adapter is an identity connector that you can use in the Kony DBX application suite. This adapter supports multi-factor authentication (MFA) (if enabled), and it fetches profile and security attributes of users from the configured DBX server.



To configure an Identity service by using Kony Customer 360 adapter, follow these steps:

1. On the Identity service Design screen, in the **Name** box, type a name for the service.
2. From the **Type of Identity** list, select **Kony Customer 360**.
3. In the **Kony Customer 360 Identity Service Endpoint** box, type the URL to which the Identity service must navigate to get the meta data.
4. The **Enable MFA** check box is selected by default. This helps you in the test login flow to test the status of the short-lived known_user token (whether it is issued or not). This setting has no impact on the DBX server, as the server's adaptive algorithm decides the MFA status, and then responds accordingly in the login flow.
5. In the **Pre-MFA Token Timeout (in Minutes)** box, type the required value. It determines the timeout value for the short-lived known_user token.
6. In the **Provider Settings (Optional)** section, do the following:
    - In the **Enter Parameter Key** box, type the parameter that you want to configure as an additional attribute. For example, `<Backend-config-key>`.
    - In the **Enter Parameter Value**box, type the necessary parameter value. For example, `<backend-config-key-value>`.
7. In the **Provider Headers (Optional)** section, type the header parameters from the login request.
8. In **Advanced > Post Authentication URL**, type the URL that you want to invoke after an Identity session is created. After this URL is invoked, the Identity response is returned to the client.
9. Click **Test Login** to verify the credentials. The **Test Login** dialog box appears.
    - Type the required details in the **Header** and **Body** boxes for the custom Identity provider.
    - The entries for the Header and Body boxes are auto-inserted to the login request. You can delete an entry by clicking the **Delete** beside it.
    - Click Sign In. The test results are displayed in the Identity Response dialog.
10. Click **Save**. Kony Fabric displays the Identity screen, and the custom Identity service is

configured.

21.5.18.1 Initial Call Process by using Kony Customer 360 Adapter

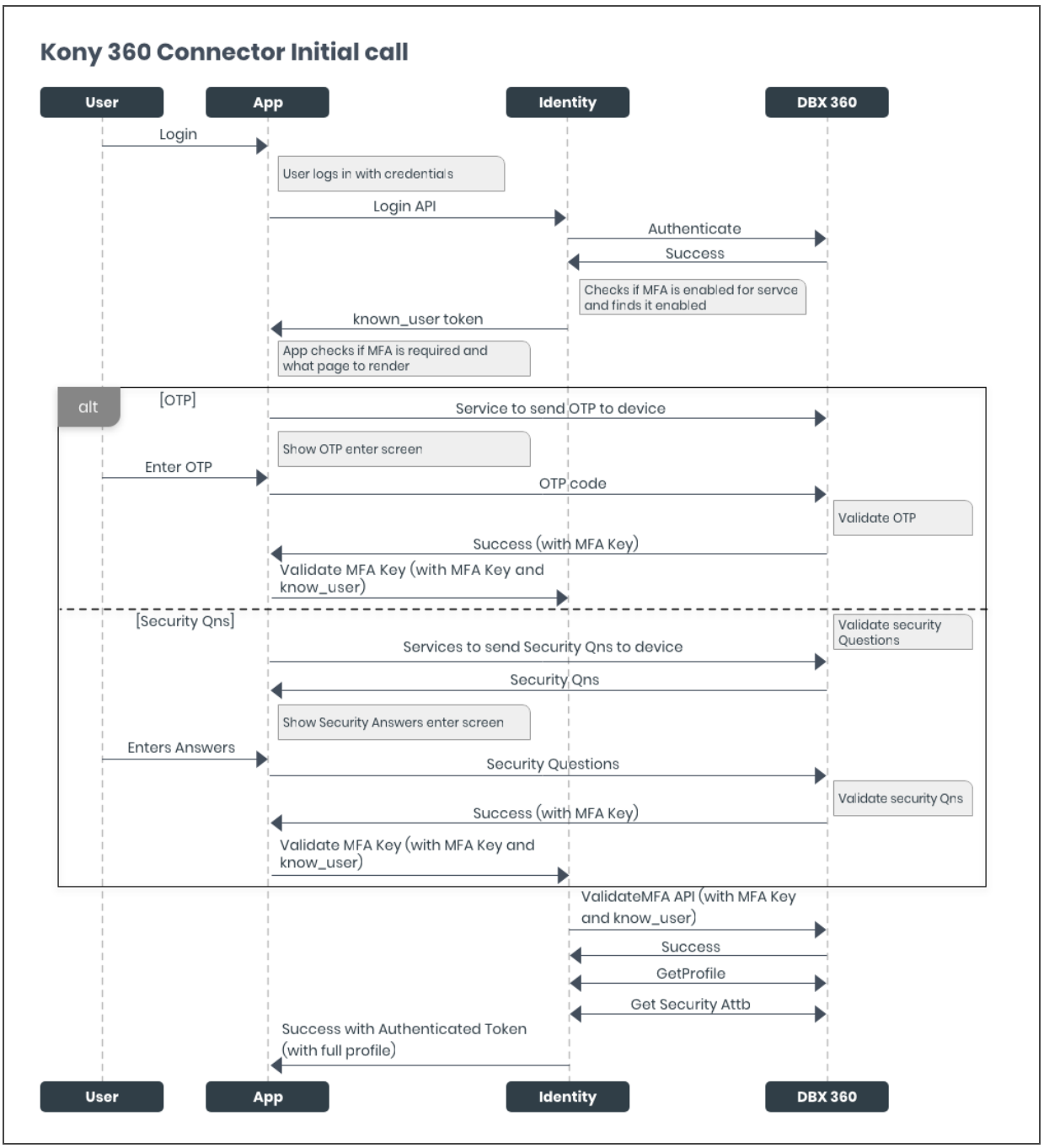
When a user signs in to the DBX application for the first time, the following calls occur among the DBX app, Identity service, and DBX server:

- The login request is sent to the Kony Fabric Identity service through the Login API.
- The Identity service sends the login request to the DBX server to be authenticated.
- On successful authentication of the user from the DBX server, the Identity service verifies the MFA status. By default, MFA is enabled. If MFA is not enabled, the Identity service returns the authenticated token.
- If MFA is enabled, the Identity service sends a "**known_user token**" to the DBX application.
- The DBX app contains the custom logic internal to its implementation to fetch MFA. The configured MFA can be of any type, namely: OTP, and/or security questions, and/or any custom logic to obtain an MFA key.
- **OTP flow**
    - If the bank or user enables OTP as the type of MFA, the DBX app displays the OTP screen. And the app invokes a service that sends an OTP to the user's registered device.
    - After the user type the OTP, it is sent to the DBX server to be validated.
    - On successful validation, the DBX server sends an MFA key to the DBX application.
    - The DBX app sends the MFA key along with the Known User token to the Identity service.
- **Security Questions Flow**
    - If the bank or user enables Security Questions as the type of MFA, the DBX app displays the Security Questions screen. And the app invokes a service that requests the DBX

server to send the security questions.

- After the user provides the answers to the questions, they are sent to the DBX server to be validated.
- On successful validation, the DBX server sends an MFA key to the DBX application.
- The DBX app sends the MFA key along with the Known User token to the Identity service.

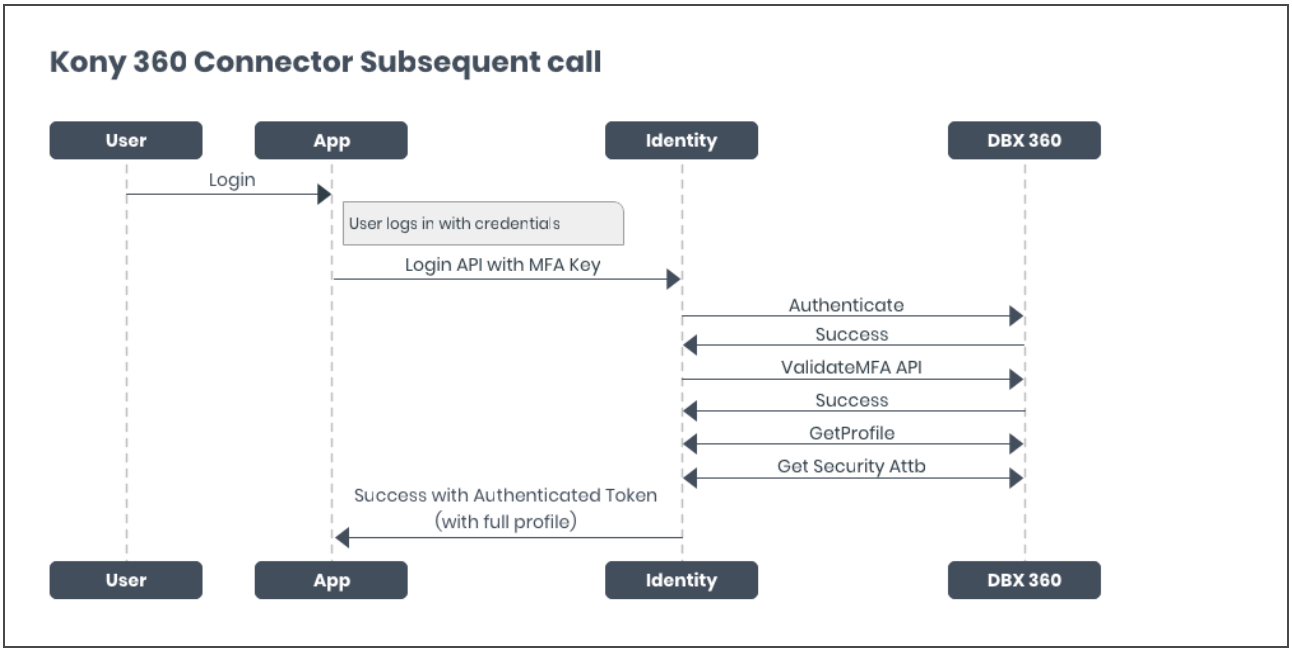
- The Identity service sends the MFA key and the Known User token to the DBX server to validate the user who is trying to access the DBX application.
- On successful validation, the DBX server sends the profile and security attributes of the user to the Identity service.
- The Identity service sends the authenticated token with full profile details to the DBX application.



21.5.18.2 Subsequent Call Process by using Kony Customer 360 Connector

On subsequent logins, the following calls occur among the DBX app, Identity service, and DBX server:

- The DBX application invokes the Login API along with the stored MFA key, and sends the details to the Identity service.
- The Identity service sends the MFA key and the known_user token to the DBX server to validate the user who is trying to access the DBX application.
- On successful validation, the DBX server sends the profile and security attributes of the user to the Identity service.
- The Identity service sends the authenticated token with full profile details to the DBX application.



Important: This document contains the information about the intended implementation for the connector. As the actual implementation can vary, kindly refer the DBX documentation (DBX > Digital Banking Platform > Feature Description and Specification - Digital Banking Platform > Infrastructure Services > Local Services > Local Services - API Reference > Login Local Services > dbxUserLogin (Orchestration Service)) to know the actual implementation.

21.5.18.3 SDKs

Note: These APIs are valid from V8 SP4 Fix Pack 14 onwards.

getMfaDetails

The getMfaDetails function helps you to identify whether multi factor authentication (MFA) is enabled or disabled in the Identity service. If MFA is enabled, this function also lets you know the type of MFA that is active.

Signature

```
<IdentityObj>.getMfaDetails()
```

Parameters

None

Return Type

JSON - containing enablement flag and metadata

Example

```
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var mfaDetails = idenidentitySvc.getMfaDetails();
if(mfaDetails["is_mfa_enabled"] == true){
     kony.print("mfa is enabled and mfa meta information is -
"+JSON.stringify(mfaDetails["mfa_meta"]));
}
```

validateMfa

The validateMfa function validates the MFA parameters. On successful validation, this function allows you to access the client use cases that were restricted which were restricted by multi-factor authentication.

Signature

```
< IdentityObj >.validateMfa(mfaParams , successCallback,
errorCallback)
```

Parameters

Parameter	Type	Description	Required
mfaParams	**JSON**	These parameters must match with the getMfaDetails() response parameters.	Yes
successCallback	**Function**	This function is invoked on successful MFA validation.	Yes
errorCallback	**Function**	This function is invoked when there is an error while MFA validation with the cause of failure as an argument.	Yes

Return Type

None

Example

```
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var mfaParams = {"mfa_Key" : "mfa value"};
function successCallback(){
  kony.print("validation successful");
}
function errorCallback(err){
  kony.print("validation unsuccessful with error - "+ JSON.stringify
(err));
}
idenidentitySvc.validateMfa(mfaParams , successCallback,
errorCallback);
```

21.6 Advanced Configurations

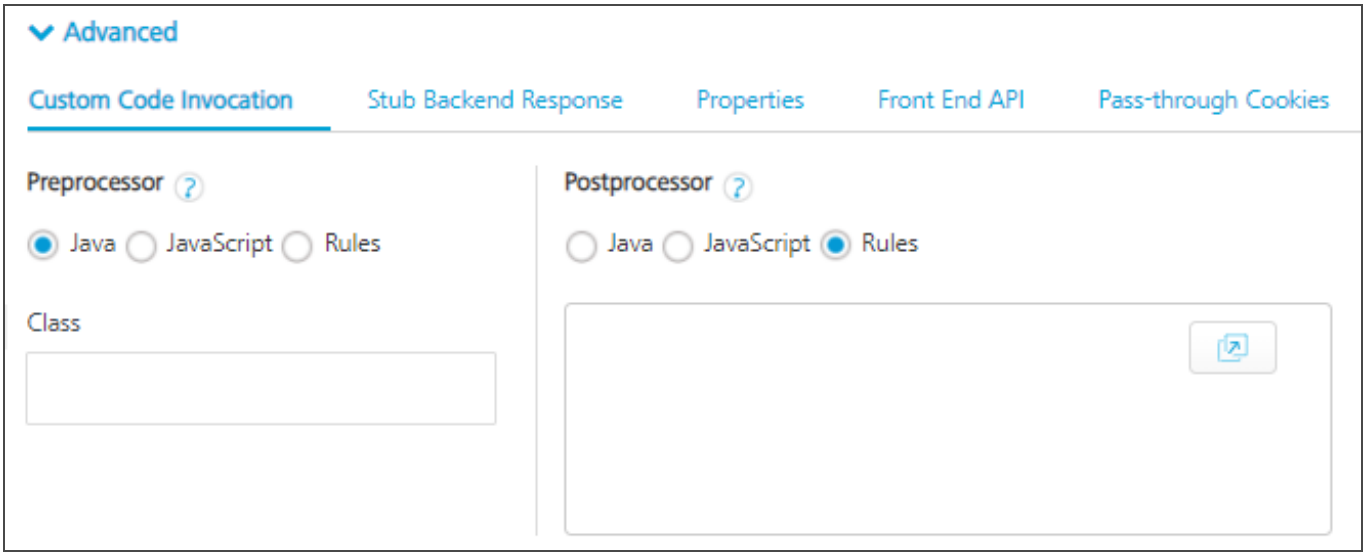
You can perform the following advanced configurations while creating an Integration Service.

- Preprocessor and Postprocessor
- Custom Code for Invoking an Integration service from Preprocessor or Postprocessor
- Sample Code for Preprocessor and Postprocessor
    - Java Sample Code for Preprocessor and Postprocessor
    - JavaScript Sample Code for Preprocessor and Postprocessor
- Enhanced Identity Filters
- Collection Support
- Override API Throttling Configuration

- Configuring Custom Front End URLs
- Developing Apps based on a Stubbed Service
- XPath

21.6.1 Custom Code Invocation - Preprocessor and Postprocessor

Configure the parameters for the preprocessor and postprocessor to filter the request and response objects for your business requirements. You can specify the Java class name or custom JavaScript code or Rules for preprocessor and postprocessor. Java class names contain the preprocessor and postprocessor.



Java Preprocessor and Postprocessor

The preprocessor and postprocessor are Java classes that implement **DataPreProcessor** / **DataPreProcessor2** and **DataPostProcessor** / **DataPostProcessor2** interfaces. A developer can write custom code in the **execute** method of the preprocessor or postprocessor class.

For a sample Java class code, refer to Java Sample Code for Preprocessor and Postprocessor.

For various objects (session and request) and the methods with sample Java class code, refer to http://docs.kony.com/konylibrary/integration/MiddlewareAPI/index.html

> ***Note:*** You need the `middleware-system.jar` for defining custom code.
>
> - **For on-premises** Kony Fabric, you can find the jar within the installation folder `<KonyFabricInstallDir>/middleware_home/`. You can also download the `middleware-system.jar` from Admin Console.
>
> - **For Cloud**, contact Kony Cloud Support for getting the appropriate middleware-system.jar for your server version.

> ***Note:*** For details on middleware APIs for preprocessor and postprocessor, contact refer to http://docs.kony.com/konylibrary/integration/MiddlewareAPI/index.html

JavaScript Preprocessor and Postprocessor

Java Preprocessor and Postprocessor

Based on the interface, the preprocessors and postprocessors implement the following objects: `<serviceInputParams>`, `<request>`, `<response>` and `<result>`.

Kony Fabric supports modifying the result object as JSON object in JavaScript. Two new APIs, **resultToJSON** and **jsonToResult** in **JavaScript**.

- For **Postprocessor**, you must call **resultToJSON** API which converts result object to JSON object. You can modify the result object as a string. For sample JavaScript code, refer to Sample code for **Postprocessor**.
- For **Preprocessor**, you must call **resultToJSON** API which returns JSON object. You can use the JSON object to modify the result as JSON and reset the result using **jsonToResult** API. For sample JavaScript code, refer to Sample code for **Preprocessor**.

You can use these objects and their corresponding methods directly in JavaScript code.
For a sample JavaScript code, refer to JavaScript Sample Code for Preprocessor and Postprocessor.

Rules Preprocessor and Postprocessor

You can use the **Rules** option to define you custom logic as a set of rules. This option makes defining pre and post processor custom logic closer to human language and is built using MVEL.

Based on the interface, the preprocessors and postprocessors implement the following structure for rules:

```
name: "<Name of the rule>"
description: "<Description of the rule>"
priority: <Priority of the rule>
condition: "<Condition to evaluate>"
actions:
   - "<Set of actions to execute>"
```

For more details on How to write Rules, objects, use cases with sample rules, refer to Rules for Preprocessor and Postprocessor.

The step allows you to further filter the data received from a service call.

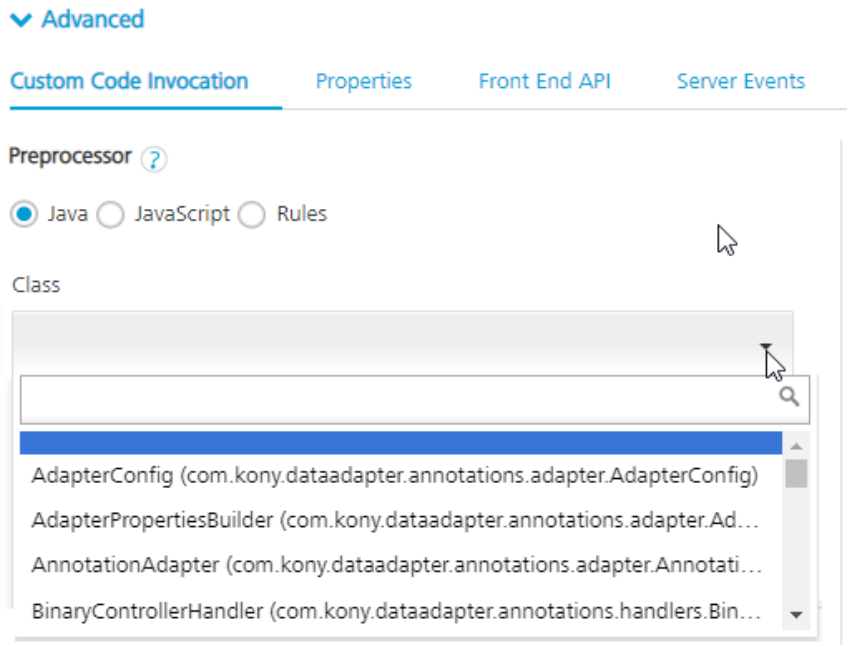
i. Under the **Custom Code Invocation**, follow these steps:

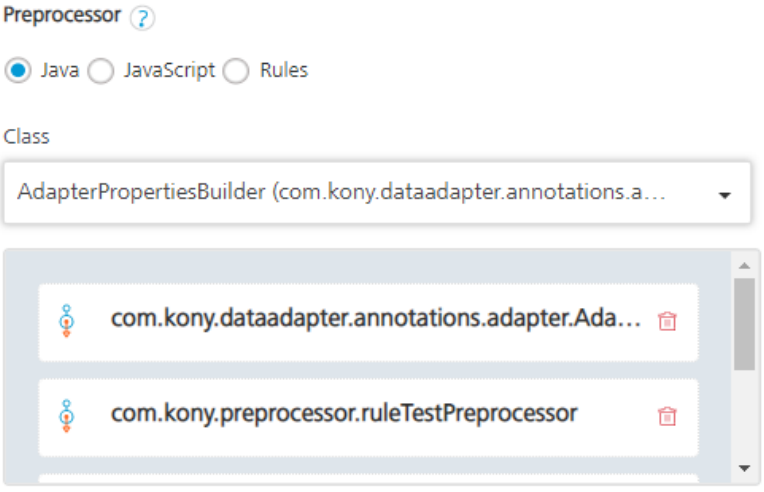
- Under **Preprocessor**, configure one of the following:
  - For **Java**, you can configure multiple Preprocessors. This is supported for Integration/Orchestration services and Object services. If you have defined your logic for multiple preprocessors in the uploaded JAR file in the service definition, you can select the available one or other preprocessors. You can arrange the selected pre-processors to be executed in the desired order during the operation call.

    ### Use Case

    When customers have a large amount of custom code, the maintainability of the code becomes an issue. The issue becomes much more complicated when multiple stakeholders working on custom code. In such cases, the custom code can be split into multiple pre/post processors so that stakeholders can work on their respective modules. This increases the upgradability and maintainability of the custom code.

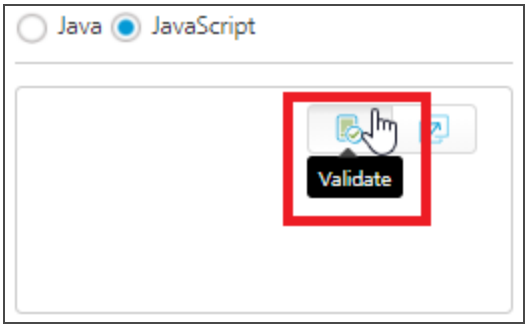
Select **Java**, and from the **Class** list, select a preprocessor class. You can select one or more classes.
This step enables a developer to include any business logic on the data before sending the response to a mobile device.





- Select **JavaScript** to open a text box. Here, you can write custom JavaScript code for the preprocessor.

JavaScript Code Validation for the Preprocessor: You can validate your JavaScript Code for the preprocessor before saving an operation. To validate the JavaScript code, click the **Validate** icon in the JavaScript code text box for the preprocessor.



The following are different scenarios that occur when you validate the JavaScript code for the Preprocessor:

- If you click **Validate**, and if the JavaScript code has **no errors**, the Validate Successful message appears. You can save the operation.
- If you click **Validate**, and if the JavaScript code **has errors**, the **Error** message dialog is displayed with a **Download** link to the validation results. When you click **Download**, a .txt file with the validation results will be downloaded to your local system. In this case you must provide a valid JavaScript code, validate the updated code, and only then you can save the operation.

  > ***Important:*** If you have validated the JavaScript code and found errors, the Console does not allow you to save the operation.

- If you do not want to validate the JavaScript code for the Preprocessor, and want to save the operation, do not click **Validate**.

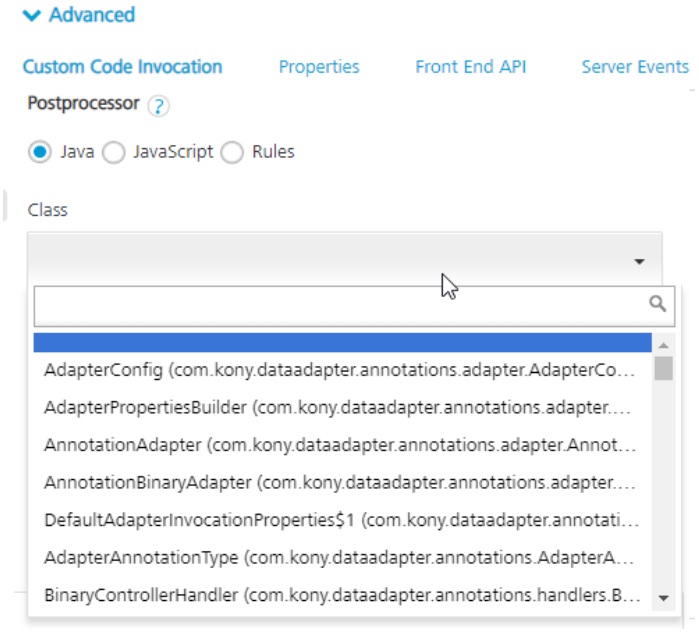
- Select **Rules** to open a text box. Here, you can write rules for the preprocessor.

- Under **Postprocessor**, configure one of the following:
  - For **Java**, you can configure multiple Postprocessors. This is supported for Integration/Orchestration services and Object services. If you have defined your logic for multiple post-processors in the uploaded JAR file in the service definition, you can select the available one or other post-processors. You can arrange the selected post-processors to be executed in the desired order during the operation call.

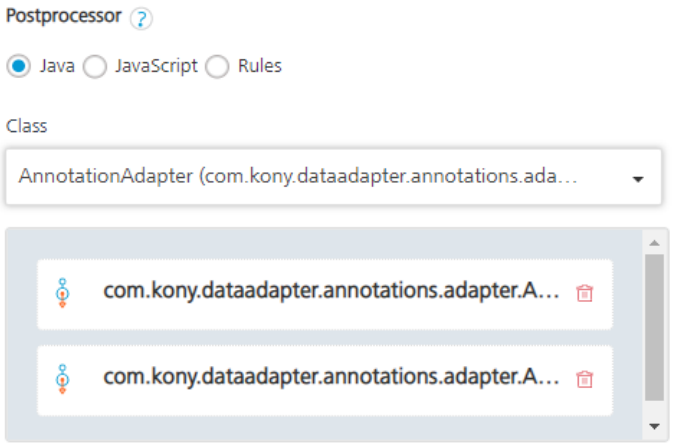
    ### Use Case

    When customers have a large amount of custom code, the maintainability of the code becomes an issue. The issue becomes much more complicated when multiple stakeholders working on custom code. In such cases, the custom code can be split into multiple pre/post processors so that stakeholders can work on their respective modules. This increases the upgradability and maintainability of the custom code.

    Select **Java**, and from the **Class** list, select a postprocessor class. You can select one or more classes.
    This step enables a developer to include any business logic on the data before sending the response to a mobile device.

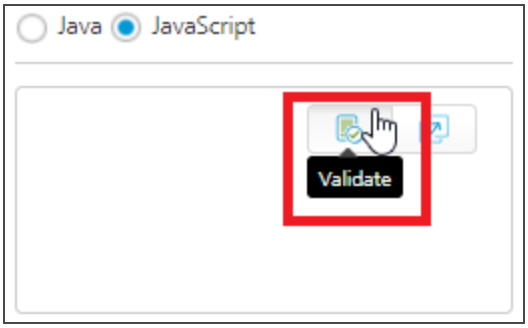
    

    You can rearrange the order of the classes to be executed, if required.



- Select **JavaScript** to open a text box. Here, you can write custom JavaScript code for the postprocessor.

  JavaScript Code Validation for the Postprocessor: You can validate your JavaScript Code for the postprocessor before saving an operation. To validate the JavaScript code, click the **Validate** icon in the JavaScript code text box for the postprocessor.



The following are different scenarios that occur when you validate the JavaScript code for Postprocessor:

  - If you click **Validate**, and if the JavaScript code has **no errors**, the Validate Successful message appears. You can save the operation.

- If you click **Validate**, and if the JavaScript code **has errors**, the **Error** message dialog is displayed with a **Download** link to the validation results. When you click **Download**, a .txt file with the validation results will be downloaded to your local system. In this case you must provide a valid JavaScript code, validate the updated code, and only then you can save the operation.

  > ***Important:*** If you have validated the JavaScript code and found errors, the Console does not allow you to save the operation.

- If you do not want to validate the JavaScript code for the Postprocessor, and want to save the operation, do not click **Validate**.

- Select **Rules** to open a text box. Here, you can write rules for the postprocessor.

ii. Under the **Properties** section, provide details for the following advanced service properties:

- **Timeout (in ms)** - the duration in milliseconds after which the service call times out. Provide the details in the text box.
- **Cache Response** - the duration in seconds within which the service response is fetched from the cache. Select the **Cache Response** check box, and provide the details in the text box.
- **Unescape embedded xml in response -** To ignore the MuleSoft response received in the XML value field, select the **Unescape embedded xml in response** check box.
- **Response Encoding -** Select the appropriate response encoding. The default value is UTF-8. For more information about different encoding schemes, refer to Response Encoding Schemes.
- **No. of connection retries -**Represents the number of times the service should be invoked in case of a failure. For example, consider a case in which an operation is invoked, and it fails due to a network issue. In this case, if you set the value as 2, the

operation will invoke the service two more times after the failure. If you set the value to 0 (zero), the operation will not invoke the service after the failure.

iii. Under the **Pass-through Cookies**, specify a comma separated list of cookie names. When cookies names are present in the incoming client request, these cookies are sent to the backend target along with preprocessor request. A sample value looks like - cookie1, cookie2.

Exception Handling in Preprocessor - Support to access config map as part of onException handler

The service config and input map are being sent to the onException Implementation of postprocessor.

For example:

```
 @OnException
  public Result executeWithAnnotation(@KonyContext Result result,
      @KonyContext DataControllerRequest request, @KonyContext
DataControllerResponse response,
      @KonyContext(parameterName = ContextParams.INPUT) Map<String,
Object> inputMap,
      @KonyContext(parameterName = ContextParams.CONFIGMAP)
Map<String, Object> configMap,
      @KonyException Exception exception) throws Exception {

    Param config = new Param();
    config.setName("className");
    config.setValue((String)configMap.get("className"));

    result.setParam(config);
    return result;
}
```

> ***Note:*** In a Java service when an exception occurs, middleware calls the OnException method of the postprocessor if configured. Extended the ability to call onException even in the case of exception in preprocessor and postprocessor.

> ***Note:*** You should not modify JVM timezone through custom code as modifying it can result in app server outage.

> ***Note:*** The Pre and post-processing logic feature is not supported for a Swagger service.

21.6.2 Rules as Pre and Post Processors

Kony Fabric provides the ability to write custom code as pre and post processors to modify request and response parameters to service calls. Historically custom code can be implemented either in Java or JavaScript. To simplify implementing custom logic in pre and post processors, a new **Rules** option is introduced from Kony Fabric V8 SP4. You can use the **Rules** option to define your custom logic as a set of rules. This option makes defining pre and post processor custom logic closer to human language and is built using MVEL.

> ***Note:*** The Rules functionality is supported only for the pre and post processor of integration services.

- **Why Use Rules**:
  - Quick and easy to write custom logic.
  - Easy to understand and change as per your business logic.
  - No maintenance overhead. For example, No need to maintain separate jars or library and export them to various environments while publishing the updated app.
- **What is the Structure of rules**: Rules have a structure in the form of statements, as shown in the following table:

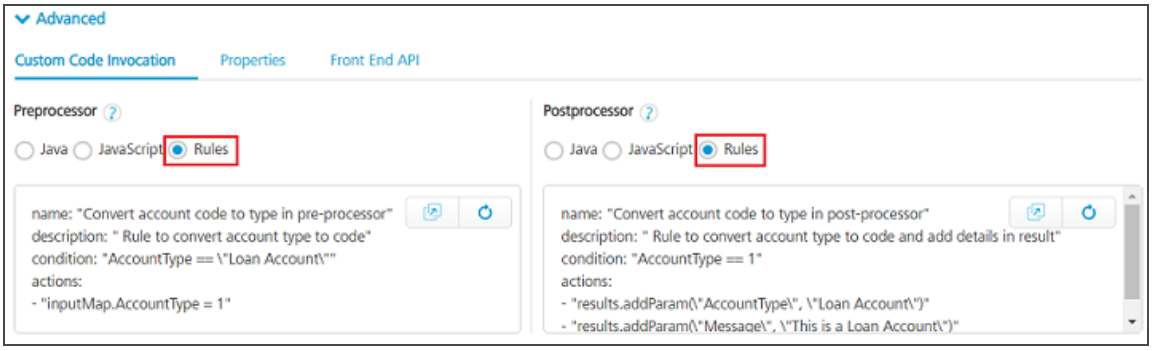
Sample Rules Structure

```
name: "<Name of the rule>"
description: "<Description of the rule>"
priority: <Priority of the rule>
condition: "<Condition to evaluate>"
actions:
   - "<Set of actions to execute>"
```

Description of Rules Structure

- **Name:** A unique name of the rule. This is a mandatory field.
- **Description:** A description for the rule.
- **Priority**: An integer value that represents the order to execute the rule. The bigger the value, the higher the priority.
- **Condition**: An expression that is evaluated by the Rules engine. When the condition evaluates to True, the engine executes a set of actions. This is a mandatory field.
  For example, `response != null` can be used to check whether the back-end response is empty.
- **Action**: A set of statements that are executed when the condition evaluates to True. This is a mandatory field.
  For example, `statusCode = 200` sets status code to 200.

- **How to Write rules:**
  a. When you are configuring an operation, in the **Advanced > Custom Code Invocation** for pre-processor or post-processor, click the **Rules** option.



21.6.2.1 Built-in Objects

The following objects help you to write rules in Kony Fabric.

Objects	Description
"configurationParameters"	Used to access the Server and Client App parameters that are set by the developer in the App Services console. This is equivalent to using ConfigurableParameters in Java.
"continueExecution"	Used in the Pre-processor to terminate back-end calls. Set the value to false if you want to terminate the back-end call. By default, the value is set to true.
"customMetrics"	This is used to access custom metrics. For more details to create custom reports and Metrics, refer Custom Reporting - Metrics, Reports, and Dashboard Guide
“deviceHeadersMap”	Used to set headers that are passed to the client and is equivalent to using setDeviceHeaders in Java.
"headerMap"	Used to access the header map of a request. A client can directly access the header map or the individual key-value pairs of the header map.
“identityHandler"	Used to access the identity attributes when a service is protected by an identity service.

Objects	Description
"inputMap"	Used to access the input map of a request. A client can directly access the input map or the individual key-value pairs of the input map.
"logger"	Used to add a log statement with the appropriate level.
"response"	Used to modify the response body and is equivalent to using setResponse in Java.
"results"	Used to modify the results. A Result is an abstraction of a back-end response. The Result is a collection of Params, Data-sets, and Records. For more details, refer Result.
"resultCache"	Used to access the cache in pre-processor and post-processor. This is equivalent to using ResultCache in Java.
"servicesManager"	Used to invoke an integration service with the specified service id, operation id and version.
"session"	Used to modify the session that is associated with the request. For more details, refer Session A client can access values from the session and the individual attributes of the session.
“statusCode”	Used to set the status code of the response and is equivalent to using setStatusCode in Java.
"ua"	Used to access the User Agent Header of the request.

21.6.2.2 Built-in Functions

The following functions help you to write rules in Kony Fabric.

Functions	Description

"Check.isWithin"	Checks if an element is in a specified range. It will return true if the element present in the specified range, otherwise false.
	• Signature: `isWithin(double fromInclusive, double toInclusive, double elementToFind)`
	• Example
	`Check.isWithin(100, 300, 250) = true`
	`Check.isWithin(100, 300, 350) = false`

"Check.isEmpty"	Checks if a CharSequence is empty ("") or null.
	• Signature: `isEmpty(final CharSequence cs)`
	• Example
	`Check.isEmpty(null) = true`
	`Check.isEmpty("") = true`
	`Check.isEmpty(" ") = false`
	`Check.isEmpty("xyz") = false`
	`Check.isEmpty("  abc  ") = false`

"Check.isNotEmpty"	Checks if a CharSequence is not empty ("") and not null.
	• Signature: `isNotEmpty(final CharSequence cs)`
	• Example
	`Check.isNotEmpty(null) = false`
	`Check.isNotEmpty("") = false`
	`Check.isNotEmpty(" ") = true`
	`Check.isNotEmpty("xyz") = true`
	`Check.isNotEmpty("  abc  ") = true`

Functions	Description

"Check.isBlank"	Checks if a CharSequence is empty (""), null or white-space only.
	• Signature: `isBlank(final CharSequence cs)`
	• Example
	`Check.isBlank(null) = true`
	`Check.isBlank("") = true`
	`Check.isBlank(" ") = true`
	`Check.isBlank("xyz") = false`
	`Check.isBlank(" abc ") = false`

"Check.isNotBlank"	Checks if a CharSequence is not empty (""), not null and not white-space only.
	• Signature: `isNotBlank(final CharSequence cs)`
	• Example
	`Check.isNotBlank(null) = false`
	`Check.isNotBlank("") = false`
	`Check.isNotBlank(" ") = false`
	`Check.isNotBlank("xyz") = true`
	`Check.isNotBlank(" abc ") = true`

Functions	Description

"Check.isEqualTo"	Compares two CharSequences, returning true if they represent equal sequences of characters.
	• Signature: `isEqualTo(final CharSequence cs1, final CharSequence cs2)`
	• Example
	`Check.isEqualTo(null, null) = true`
	`Check.isEqualTo(null, "abc") = false`
	`Check.isEqualTo("abc", null) = false`
	`Check.isEqualTo("abc", "abc") = true`
	`Check.isEqualTo("abc", "ABC") = false`

"Check.isEqualToIgnoringCase"	Compares two CharSequences, returning true if they represent equal sequences of characters, ignoring case.
	• Signature: `isEqualToIgnoringCase(final CharSequence str1, final CharSequence str2)`
	• Example
	`Check.isEqualToIgnoringCase(null, null) = true`
	`Check.isEqualToIgnoringCase(null, "abc") = false`
	`Check.isEqualToIgnoringCase("abc", null) = false`
	`Check.isEqualToIgnoringCase("abc", "abc") = true`
	`Check.isEqualToIgnoringCase("abc", "ABC") = true`

21.6.2.3 Sample Rules

Modify request input

Use Case	Changing request input before sending it to the back end. For example, you can map the account type received from the request to an account code.

Rule	`name: "Convert account type to account code in pre-processor"`
	`description: "Rule to convert account type to account code"`
	`condition: "AccountType == \"Loan Account\""`
	`actions:`
	`- "inputMap.AccountCode = 1"`
	`- "inputMap.Message = \"This is a loan account\""`
	
	The given sample rules above checks the account type, if the account type is Loan Account, then the associated account code is set to 1.
	
	The `inputMap` object is used to access the parameters in the request that comes from the device.

Modify result

Use Case	Modifying the result of an operation. For example, you can add the account type in the result depending upon the account code.

Rule	``` name: "Add parameter in result" description: "Add a parameter in result for a specific account type" condition: "AccountCode == 1" actions: - "results.addParam(\"AccountType\", \"Loan Account\")" ``` The given sample rule checks the account code, if the account code is equal to 1, then the account type parameter is set as Loan Account. The `results` object is used to modify the result of an operation.

Modify response, status code and headers sent to a device

Use Case	Changing the response body, status code, and headers that are sent to the device. For example, you can categorize the customer based on the quarterly average balance and send specific headers to the device to render appropriate UX of client application.

Rule	`name: "Modify response in rules."`
	`description: "Set response message, status code and headers sent to device."`
	`condition: "quarterlyAvgBalance >= 100000"`
	`actions:`
	`- "response = \"{\\\"category\\\": \\\"Preferred customer\\\"}\""`
	`- "statusCode = 200"`
	`- "deviceHeadersMap.put(\"X-Kony-Preferred-Customer\", \"true\")"`
	
	The given sample rule checks the quarterlyAvgBalance parameter. If the parameter is greater than or equal to 100000, then the response body, status code and headers sent to the device are changed.
	
	The `response` object is used to modify the response body.
	
	The `statusCode` object is used to set the status code.
	
	The `deviceHeadersMap` object is used to modify headers that are sent to the device.

Abort call to a back end

Use case	Aborting the call to back end in pre-processor. For example, you can allow requests only from mobile devices.

Rule	`name: "Abort call to back end"`
	`description: "Check if user agent is not mobile then abort the call to back end"`
	`priority: 1`
	`condition: "ua != \"mobile\""`
	`actions:`
	`- "continueExecution = false;"`
	`- "results.addErrMsgParam(\"Platform not supported\");"`
	`- "results.addHttpStatusCodeParam(404);"`
	`- "results.addOpstatusParam(-1);"`
	
	The given sample rule checks the user agent, if the user agent `is not equal to mobile`, then the request is canceled by setting the `continueExecution` action to false.
	
	The `continueExecution` object is used to control the further execution of the request. By default, it is set to `true`. If it is set to `false`, then request is not sent to the back end.

Access cache

Use case	Accessing data from the cache. For example, you can populate a country code in the cache if it is not present.

Rule	`name: "Access cache in rules"`
	`description: "Checks if country code for India is present in cache, if not it will store in the cache"`
	`priority: 1`
	`condition: "resultCache.retrieveFromCache(\"india\") == null"`
	`actions:`
	`- "resultCache.insertIntoCache(\"india \", \"+91\")"`
	
	The given sample rule checks the stored value in the cache. If the cache is empty, then the country code is added to the cache.
	
	The `resultCache` object is used to access the cache.

Invoke integration service

Use case	Invoking an Integration service in pre-processor and post-processor. For example, you can invoke an SMS or email service based on a request parameter.

Rule	```
	--- 	
	name: "Invoke send email integration service"
	description: "Execute integration service to send email if sendEmail is true in input map."
	priority: 1
	condition: "inputMap.get(\"sendEmail\") == \"true\""
	actions:
	    - servicesManager.invokeIntegration ("RulesIntegrationService", "SendEmail")
	
	--- 	
	name: "Invoke send SMS integration service"
	description: "Execute integration service to send SMS if sendSms is true in input map"
	priority: 1
	condition: "inputMap.get(\"sendSms\") == \"true\""
	actions:
	    - servicesManager.invokeIntegration ("RulesIntegrationService", "SendSms")
	``
	
	In the sample, based on the parameters sent from the device, we are invoking services either to send an SMS or email or both.
	
	The `servicesManager` object is used to invoke an integration service from rules pre/post processor.

Access Identity data

Use case	Accessing the Identity data. For example, you can access the Identity data such as the app id and the user profile for the request.

Rule	`name: "Access Identity Info"`
	`description: "Accesses identity info like first name, last name, email id, and app id in rules"`
	`priority: 1`
	`condition: "setIdentityDetails == true"`
	`actions:`
	`- "results.addParam(\"FirstName\", identityHandler.getUserProfile().getFirstName())"`
	`- "results.addParam(\"LastName\", identityHandler.getUserProfile().getLastName())"`
	`- "results.addParam(\"Email\", identityHandler.getUserProfile().getEmailId())"`
	`- "results.addParam(\"AppId\", identityHandler.getAppId())"`
	
	The given sample rule accesses the identity information such as first name, last name, email id, and app id and sends it to the device.
	
	The `identityHandler` object is used to access identity information.

Access session

Use case	Accessing data from the session. For example, you can fetch the authorization token from the session and set it in the header of the back-end call.

Rule	``` name: "Access session data" description: "Check if authorization token is present in session then set the same in header" priority: 1 condition: "session.containsKey("\authToken\")" actions: - "headerMap.authToken = authToken" ``` The given sample rule checks for an authorization token. If the authorization token is present in the session, then it is added to the header. The `session` object is used to access the session data.

Access Configurable Parameters defined in App Services

Use case	Accessing the Configurable Parameters defined in App Services.
Rule	``` name: "Access configuration properties" description: "Check if encryption enabled in client properties then set encryption key in input map." priority: 1 condition: "configurationParameters.getClientAppProperty (\"encrypt\") == \"true\"" actions: - "inputMap.encryptionKey = configurationParameters.getServerProperty (\"encryptionKey\")" ``` The given sample rule checks the encrypt property. If the encrypt property is enabled in the client properties, then the code fetches the encryption key from server properties and adds the key to the request. The `configurationParameters` object is used to access the properties that are defined in App Services.

Access custom metrics

Use case	Accessing custom metrics in pre-processor and post-processor.

Rule	``` name: "Access configuration properties" description: "Check if encryption enabled in client properties then set encryption key in input map." priority: 1 condition: "configurationParameters.getClientAppProperty(\"encrypt\") == \"true\"" actions: - "inputMap.encryptionKey = configurationParameters.getServerProperty(\"encryptionKey\")" ``` The given sample rule checks the encrypt property. If the encrypt property is enabled in the client properties, then the code fetches the encryption key from server properties and adds the key to the request. The `configurationParameters` object is used to access the properties that are defined in App Services.

Multiple rules in same pre/post processors

Use case	Writing multiple rules in the same pre-processor and post-processor. You can write multiple rules in the same pre/post processor by using `--- (three hyphens)` as the separator. ***Note:*** Please do not give separator after last rule.

Rule	``` --- name: "Convert account code to type for loan account" description: "Rule for loan account to convert account type to code and add message in result" condition: "AccountCode == 1" actions: - "results.addParam(\"AccountType\", \"Loan Account\")" - "results.addParam(\"Message\", \"This is a Loan Account\")" --- name: "Convert account code to type for saving account" description: "Rule for saving account to convert account type to code and add message in result" condition: "AccountCode == 2" actions: - "results.addParam(\"AccountType\", \"Saving Account\")" - "results.addParam(\"Message\", \"This is a Saving Account\")" --- name: "Convert account code to type for current account" description: "Rule for current account to convert account type to code and add message in result" condition: "AccountCode == 3" actions: - "results.addParam(\"AccountType\", \"Current Account\")" - "results.addParam(\"Message\", \"This is a Current Account\")" ``` The given sample rule invokes multiple rules.

Iterate over a set of values

Use case	Iterating over a set of values.

Rules	`name: "Iterate over records using for loop in rules."`
	`description: "Add 80% of price as discounted price of book."`
	`condition: "\"giveDiscount\" == true"`
	`actions:`
	`- "for (Record record : results.getDatasetById (\"books\").getAllRecords()) {`
	`Record bookRecord = record.getRecordById(\"book\");`
	`double price = Double.parseDouble (bookRecord.getParamValueByName(\"price\"));`
	`bookRecord.addParam(\"discountedPrice\", price * 0.8);`
	`}"`
	
	The given sample rule iterates a books dataset and adds discounted prices for each book in the dataset.

21.6.3 Custom Code for Invoking an Integration Service from a Preprocessor, Post-processor, Custom filter, Custom Servlet, or Java service

The middleware Java API helps to invoke an integration/Orchestration/object service from custom code.

> ***Important:*** The Kony Fabric Console does not send the dependent artifacts with the request for a Java Service. Therefore, invoking a service from Custom Code using the `ServicesManager` API is not supported.

How to use the API

- Initial steps is to create an Operation Data and Service Request Objects:

```
OperationData serviceData = request.getServicesManager()
                            .getOperationDataBuilder()
                            .withServiceId(<Service Id>)
```

```
                                    .withOperationId(<Operation Id>)
                                    .build();
```

- To invoke and get JSON response:

```
 ServiceRequest serviceRequest = request.getServicesManager
().getRequestBuilder (serviceData)
                                          .withInputs(<Input Map>)
                                          .withHeaders(<Header Map>)
                                          .build();

String response = serviceRequest.invokeServiceAndGetJson();
```

- To invoke and get result object:

```

ServiceRequest serviceRequest = request.getServicesManager
().getRequestBuilder (serviceData).build();
Result result = serviceRequest.invokeServiceAndGetResult();
```

> ***Note:*** In case of result object API.

- To invoke a PassThroughService:

```
 ServiceRequest serviceRequest = request.getServicesManager
().getRequestBuilder (serviceData).build();
BufferedHttpEntity response =
serviceRequest.invokePassThroughServiceAndGetEntity();
```

- To get the service manager in the custom filter or servlets using HttpServletRequest instance:

```
ServicesManager servicesManager =
ServicesManagerHelper.getServicesManager(request); OperationData
operationData = servicesManager.getOperationData();

```

```
IdentityHandler identityHandler =
servicesManager.getIdentityHandler();
```

- To fetch user profile, user attributes and security attributes:

```
UserProfile userProfile = identityHandler.getUserProfile();
Map<String, Object> userAttributes =
identityHandler.getUserAttributes();
Map<String, Object> securityAttributes =
identityHandler.getSecurityAttributes();
Map<String, Object> appAttributes =
identityHandler.getAppAttributes();
String appId = identityHandler.getAppId();
```

- To fetch user attributes and security attributes using provider name:

```
Map<String, Object> userAttributes =
identityHandler.getUserAttributes("identity provider name");
Map<String, Object> securityAttributes =
identityHandler.getSecurityAttributes("identity provider name",
true);
```

- If you want to invoke an authentication service which contains identity parameters or headers, modify the inline invocation of the service and include the new authorization token (X-Kony-Authorization token). You can pass the token to the parent service (service which invokes the inline service), read the token from the custom code and pass it in the Services Manager API.

```
ServiceRequest serviceRequest = request.getServicesManager
().getRequestBuilder(serviceData)
       .withInputs( < Input Map > )
       .withHeaders( < Header Map > )
       .withAuthorizationToken( < x - kony - authorization - token > )
       .build();

String response = serviceRequest.invokeServiceAndGetJson();
```

- To execute a sequence of RDBMS operations in a transaction you can use Transaction manager. When you execute services through Transaction Manager, the services listed in TransactionExecutor are executed in a single transaction, and the transaction roll backs if there is any error.

  For Example: Let us assume a database has a **Country** table and an **Address** table. The Address table contains a country column. You can use Transaction Manager in the transaction, where the operations update the **Country** table first and then update address with country in the **Address** table. If any operation fails, the changes done by previous operations/verbs will roll back automatically.

  Following is the sample code for it:

```
public class CreateCountryAndAddressInTransaction implements
ObjectServicePreProcessor {
    private static final Logger LOGGER = LogManager.getLogger
(CreateCountryAndAddressInTransaction.class);

    private String countryId;

    public void execute(FabricRequestManager
fabricRequestManager,
        FabricResponseManager fabricResponseManager,
FabricRequestChain fabricRequestChain)
    throws Exception {
        ServicesManager servicesManager =
fabricRequestManager.getServicesManager();


        DatabaseTransactionManager manager =
servicesManager.getDatabaseTransactionManager();
        TransactionExecutor < Result > txe = () - > {
            try {
```

```
                // Create of a country
                OperationData createCountryOperationData =
servicesManager.getOperationDataBuilder()
                    .withServiceId
("TransactionObjectService").withObjectId("country")
                    .withOperationId(OperationEnum.create.name
()).withVersion("1.0").build();

                Map < String, Object > countryInputs = new
HashMap < > ();
                countryInputs.put("CountryName", "MyCountry");
                ServiceRequest createCountryServiceRequest =
servicesManager
                    .getRequestBuilder
(createCountryOperationData).withInputs(countryInputs)
                    .build();
                Result resultForCreateOperation =
createCountryServiceRequest.invokeServiceAndGetResult();
                this.countryId =
resultForCreateOperation.getDatasetById("country").getRecord(0)
                    .getParam("CountryID").getValue();

                // Create of an Address with country value as
input

                OperationData createAddressOperationData =
servicesManager.getOperationDataBuilder()
                    .withServiceId
("TransactionObjectService").withObjectId("address")
                    .withOperationId(OperationEnum.create.name
()).withVersion("1.0").build();

                Map < String, Object > addressInputs = new
```

```
HashMap < > ();
                addressInputs.put("Address1", "MyAddress");
                addressInputs.put("AddressID", this.countryId);
                ServiceRequest createAddressServiceRequest =
servicesManager
                    .getRequestBuilder
(createAddressOperationData).withInputs(addressInputs).build();


createAddressServiceRequest.invokeServiceAndGetResult();
            } catch (MiddlewareException e) {
                LOGGER.error("Error in custom code", e);
                throw new MobileFabricRuntimeException("failed");
                // Having a catch and throw is necessary for
Transaction manager.In this particular example , If the country
creation is succesfull and the create address fails country
creation will also be rolled back
            }
            return null;
        };

        manager.executeInTransaction(txe);

        fabricRequestChain.execute();
    }
}
```

21.6.3.1 Exception Handling (for Java Preprocessor and Postprocessor)

In case of Exception from the API, result object will be available as an attribute in the request object.

For example:

```
try {
   result = serviceRequest.invokeServiceAndGetResult();
```

```
} catch (Exception e) {
    //result object with error message or result update in onException
can be retrieved from request //object as below.
    return request.getAttribute(MWConstants.RESULTS);
}
```

> ***Note:*** For server upgrade from version 7.x to version 8.3.x, if Services Manger APIs are used in the custom code, you must rebuild the jar with the latest middleware plugins to avoid IncompatibleClassChangeError.

21.6.3.2 Exception Handling (for Multiple Java Preprocessor and Postprocessor)

- If there is an exception occurs in a request with many pre-processors, the Kony Server ignores the request.
- If there is an exception occurs in a response, the Kony Server sends the response including the exception details.

21.6.4 Sample Code for Preprocessor and Postprocessor

21.6.4.1 Java Sample Code for Preprocessor and Postprocessor

Sample DataPreProcessor

The following is a sample DataPreProcessor file:

```
 import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import javax.swing.text.html.HTMLDocument.Iterator;

import org.apache.log4j.Logger;
```

```
import com.konylabs.middleware.common.DataPreProcessor;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.controller.DataControllerResponse;
import com.konylabs.middleware.dataobject.Result;
import com.konylabs.middleware.session.Session;

public class sampleDataPreProcessor implements DataPreProcessor {

    @Override
    public boolean execute(HashMap arg0, DataControllerRequest arg1,
Result arg2)
    throws Exception {

        //Write application logic here
        // if true is returned then service call and post processor
are invoked. If false is returned, then service call and post
processor are not invoked.
        return true;
    }
}
```

Sample DataPreProcessor2

The following is a sample DataPreProcessor2 file:

```
 package jsoncustjar;
import java.util.HashMap;
import java.util.Map;

import com.konylabs.middleware.common.DataPreProcessor2;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.controller.DataControllerResponse;
import com.konylabs.middleware.dataobject.Result;
```

```
public class DigitePreProcessor implements DataPreProcessor2 {

    @Override
    public boolean execute(HashMap arg0, DataControllerRequest arg1,
        DataControllerResponse arg2, Result arg3) throws Exception {

        System.out.println("Pre-Processor started");

        // Below sample code is to append input params to the request

        if (arg0.get("DigiteLoginPwd") == null) {
            arg0.put("DigiteLoginPwd", "mypassword");
            System.out.println("Added DigiteLoginPwd to Map");

        }

        String selectQuery = (String) arg0.get("$select");

        //some

        arg0.put("select", "$select=" + selectQuery);

        System.out.println("Pre-Processor ended");

        return true;
    }
}
```

Sample DataPostProcessor

The following is a sample DataPostProcessor file:

```
 import com.konylabs.middleware.common.DataPostProcessor;
import com.konylabs.middleware.controller.DataControllerRequest;
```

```
import com.konylabs.middleware.controller.DataControllerResponse;
import com.konylabs.middleware.dataobject.Param;
import com.konylabs.middleware.dataobject.Result;

public class sampleDataPostProcessor implements DataPostProcessor {

    @Override
    public Object execute(Result arg0, DataControllerRequest arg1)
    throws Exception {

     // Write application logic here to modify the results returned
from the service

     // return Result object here
        return arg0;
    }

}
```

Sample DataPostProcessor2

The following is a sample DataPostProcessor2 file:

```
 package jsoncustjar;

import java.util.HashMap;
import java.util.Map;

import com.konylabs.middleware.common.DataPostProcessor2;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.controller.DataControllerResponse;
import com.konylabs.middleware.dataobject.Result;

public class JSONPostProcessor implements DataPostProcessor2 {
```

```
    @Override
    public Object execute(Result arg0, DataControllerRequest arg1,
        DataControllerResponse arg2) throws Exception {
        Map <String, String>  map = new HashMap <String, String> ();
        map.put("CustHeader", "From-PostProc");
        arg2.setDeviceHeaders(map);
        return arg0;
    }
}
```

Important: When an integration service is used in an orchestration service (sequential,concurrent and looping services), and if the integration service has a postprocessor, then the `opstatus param` field from the **Result** (first argument of the execute method) object should not be removed in the postprocessor.

21.6.4.2 JavaScript Sample Code for Preprocessor and Postprocessor

PreProcessor

- To add an input parameter:

```
 // Sample JavaScript Code to add an input parameter for
preprocessor

function fun1() {
    logger.debug('Tesing put method of HashMap');
    serviceInputParams.put('place', 'London');
    return true;
}
fun1();
```

- **To add a new request parameter:**

```
 // Sample JavaScript Code to add a request parameter for
preprocessor

function fun2() {
    logger.debug('Testing addRequestParam_ method of
DataControllerRequest');
    request.addRequestParam_('newParam', 'newParamValue');
    return true;
}
fun2();
```

- **To add an attribute to the session:**

```
 // Sample JavaScript Code to add an attribute to the session for
preprocessor

function fun3() {
    logger.debug('Tesing getSession method of
DataControllerRequest');
    var varSession = request.getSession();
    varSession.setAttribute('sessionAttributeName',
'sessionAttributeValue');
    return true;
}
fun3();
```

- **To validate the input parameter before Service call:**

```
 // Sample JavaScript Code to validate the input before service
call for preprocessor

function fun4() {
    var varUsername = serviceInputParams.get('username');
```

```
    var varPassword = serviceInputParams.get('password');
    if (varUsername == 'masterUser' && varPassword == 'password')
{
        return true;
    }
    return false;
}
fun4();
```

- **A function which uses most of the request APIs:**

```
 // Sample JavaScript Code for preprocessor

function fun5() {
    logger.debug('Testing addRequestParam_ method of
DataControllerRequest');
    request.addRequestParam_('newParam', 'newParamValue');

    var varParam = request.getParameter('newParam');
    logger.debug('The Param Value is :' + varParam);
    logger.debug('Tesing containsKeyInRequest method of
DataControllerRequest');

    var varContainsKey = request.containsKeyInRequest
('newParam');
    logger.debug('Does it contain newParam :' + varContainsKey);
    logger.debug('Tesing containsKeyInRequest method of
DataControllerRequest');
    request.setAttribute('newAtt', 'newAttValue');

    var varContainsAttribute =
request.containsKeyInRequestContext('newAt');
    logger.debug('Does it contain newAtt :' +
varContainsAttribute);
```

```
    logger.debug('Tesing getAttribute method of
DataControllerRequest');

    var varGetAttribute = request.getAttribute('newAtt');
    logger.debug('The value of attribute newAtt is :' +
varGetAttribute);
    logger.debug('Tesing getHeader method of
DataControllerRequest');

    var varHeader = request.getHeader('Host');
    logger.debug('The value of Host header is :' + varHeader);
    logger.debug('Tesing GetParameterValues method of
DataControllerRequest');

    var varGetParameterValues = request.getParameterValues
('newParam');
    for (var i in varGetParameterValues) {
        logger.debug('The Array Values are :' +
varGetParameterValues[i]);
    }
    logger.debug('Tesing getRemoteAddr method of
DataControllerRequest');

    var varGetRemoteAddr = request.getRemoteAddr();
    logger.debug('The Remote Address is :' + varGetRemoteAddr);
    logger.debug('Tesing put method of HashMap');
    serviceInputParams.put('place', 'London');
    serviceInputParams.put('city', 'Madras');
    serviceInputParams.put('country', 'india');
    logger.debug('Tesing getSession method of
DataControllerRequest');

    var varSession = request.getSession();
```

```
    varSession.setAttribute('sessionAttributeName',
'sessionAttributeValue');
}
fun5();
```

- For jsonToResult API

```
 For jsonToResult APIfunction executePreProcessor() {
    var resultToModify = resultToJSON();
    resultToModify.record = {
      "message" : "Returned from Pre-processor",
      "status" : "rejected"
    };

    result = jsonToResult(resultToModify);
    return false;
}

executePreProcessor();
```

PostProcessor

- To add a new output parameter:

```
 // Sample JavaScript Code to add a new output parameter for
postprocessor

function fun5() {
    logger.debug('Tesing Adding a new Output Parameter');
    var newOutputParam = new
com.konylabs.middleware.dataobject.Param();
    newOutputParam.setName('outputParamName');
    newOutputParam.setValue('outputParamValue');
```

```
    result.setParam(newOutputParam);
}
fun5();
```

- **To retrieve an attribute added to the session (from preprocessor):**

```
 // Sample JavaScript Code to retrieve an attribute added to the
session for postprocessor

function fun6() {
    var varSessionPost = request.getSession();
    var varRequestSessionAttribute = varSessionPost.getAttribute
('sessionAttributeName');
    logger.debug('The Session Attribute Value is :' +
varRequestSessionAttribute);
}
fun6();
```

- **To add a custom device header:**

```
 // Sample JavaScript Code to add a custom device header for
postprocessor

function fun7() {
    logger.debug('Tesing setDeviceHeaders method of
DataControllerResponse');
    var varHashMap = new java.util.HashMap();
    varHashMap.put('newHeader', 'newHeaderValue');
    response.setDeviceHeaders(varHashMap);
}
fun7();
```

- **To add Datasets to result:**

```
 // Sample JavaScript Code to add a dataset to result for
postprocessor

function fun8() {
    var varDataSet = new
com.konylabs.middleware.dataobject.Dataset();
    var varRecord = new com.konylabs.middleware.dataobject.Record
();
    var varParam = new com.konylabs.middleware.dataobject.Param
();

    varParam.setName('ParamName');
    varParam.setValue('ParamValue');
    varRecord.setParam(varParam);
    varDataSet.setRecord(varRecord);
    varDataSet.setId('DataResponse');
    var varDataSet1 = new
com.konylabs.middleware.dataobject.Dataset();
    var varRecord1 = new
com.konylabs.middleware.dataobject.Record();
    var varParam1 = new com.konylabs.middleware.dataobject.Param
();
    varParam1.setName('ParamName1');
    varParam1.setValue('ParamValue1');
    varRecord1.setParam(varParam1);
    varDataSet1.setRecord(varRecord1);
    varDataSet1.setId('DataResponse1');
    var varList = new java.util.ArrayList();
    varList.add(varDataSet1);
    varList.add(varDataSet);
```

```
    result.setDataSets(varList);
}
fun8();
```

- **A function which uses most of the response APIs:**

```
 // Sample JavaScript Code for postprocessor

function fun8() {
    var newOutputParam = new
com.konylabs.middleware.dataobject.Param();
    newOutputParam.setName('outputParamName');
    newOutputParam.setValue('outputParamValue');
    result.setParam(newOutputParam);
    logger.debug('Tesing getAttribute method of
DataControllerResponse');
    response.setAttribute('responseAttribute',
'responseAttributeValue');

    var varResponseAttribute = response.getAttribute
('responseAttribute');
    logger.debug('The value of Response Attribute is :' +
varResponseAttribute);
    logger.debug('Tesing getCharsetEncoding method of
DataControllerResponse');

    var varCharsetEncoding = response.getCharsetEncoding();
    logger.debug('The value of getCharsetEncoding is :' +
varCharsetEncoding);
    logger.debug('Tesing setDeviceHeaders method of
DataControllerResponse');
```

```
    var varHashMap = new java.util.HashMap();

    varHashMap.put('newHeader', 'newHeaderValue');
    response.setDeviceHeaders(varHashMap);
    logger.debug('Tesing setStatusCode method of
DataControllerResponse');

    varStatusCode = response.getStatusCode();
    logger.debug('The Current Status Code is :' + varStatusCode);
    response.setStatusCode(302);

    varStatusCode1 = response.getStatusCode();
    logger.debug('The Current Status Code is :' +
varStatusCode1);

    var varSessionPost = request.getSession();
    var varRequestSessionAttribute = varSessionPost.getAttribute
('sessionAttributeName');
    logger.debug('The Session Attribute Value is :' +
varRequestSessionAttribute);
}
fun8();
```

- **To set custom httpStatusCode params through JavaScript postprocessor in result:**

```
function changeHttpStatusCode() {
    result.getParamByName("httpStatusCode").setValue(300);
}
changeHttpStatusCode();
```

- To set custom opstatus params through JavaScript postprocessor in result:

```
function changeOpstatus() {
    result.getParamByName("opstatus").setValue(1);
}
changeOpstatus();
```

- For resultToJSON API

```
 function getCertificate_Postprocessor(){
    var resultToModify = resultToJSON();

    if(resultToModify.records &amp;amp;&amp;amp;
resultToModify.records.length &amp;gt; 0){
        var cert = resultToModify.records[0];

        resultToModify.certificate = [{
            "online_course": true,
            "doctor": {
                "id": cert.doctor.doctor_id,
                "email": cert.doctor.doctor_email,
            },
            "entity": {
                "id": cert.entity.entity_id,
                "email": cert.entity.entity_email,
                "doc_num": cert.entity.entity_doc_num,
            }
        }];

        return resultToModify;
    }
}
```

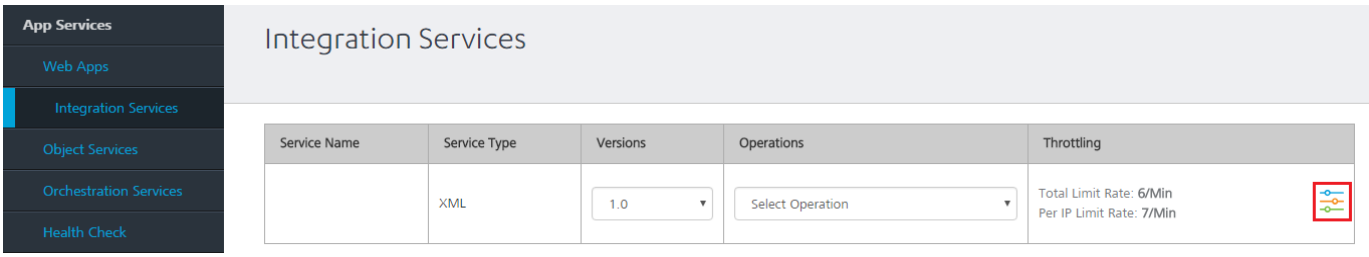
```
getCertificate_Postprocessor();
```

21.6.5 Override API Throttling Configuration

After an app's integration service enabled with API throttling configuration is published, you can override throttle configuration through Admin Console.

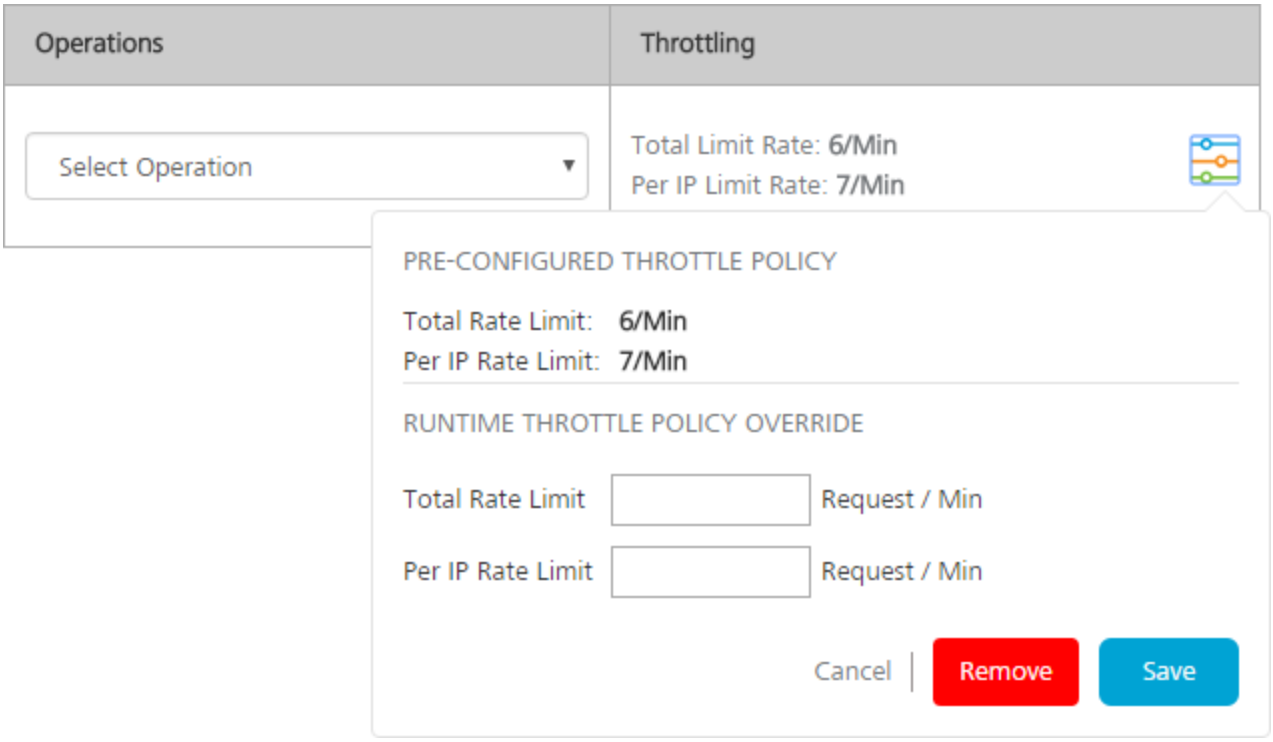
To override API throttling configuration, follow these steps:

1. Go to the Kony Fabric Admin console.
2. In your Kony Fabric Admin console, in the left-pane, click on **Integration Services**. The **Integration Services** page appears. The page displays published app services.
3. If throttling is enabled in a service, the throttling icon will be displayed under the **Throttling** section.

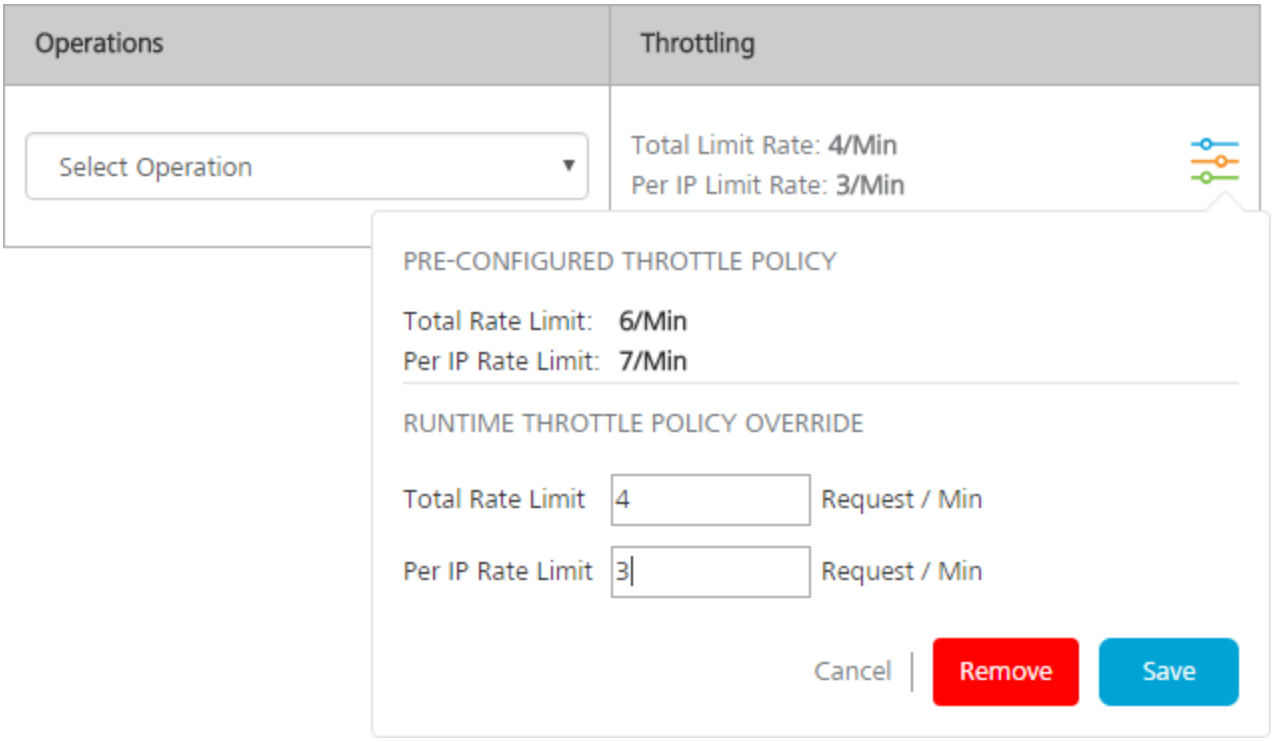


4. Click on **Throttling** icon, a throttling pane appears.

   Pre-Configured throttle values will be displayed under **PRE-CONFIGURED THROTTLE POLICY.**



You can override the throttle configuration by assigning values under **RUNTIME THROTTLE POLICY OVERRIDE.**



a. In the **Total Rate Limit** text box, enter a value. With this you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit.

b. In the **Per IP Rate Limit** text box, enter a value. With this you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit.

c. Click on **Save** to override the throttle configuration.

21.6.6 How to Develop Apps based on a Stubbed Service

The **Kony Fabric Stub back-end response** capability helps app developers to continue to develop apps when the backend services that an app connects to are not ready to be leveraged. There are several instances in an app development life-cycle when back-end systems and app development happen in parallel and only the contract or interface for the app to communicate to a backend is

finalized. In this scenario, the app developer can create a response template to "stub" the response that is expected from the actual backend. The response template can have data from service requests, hard-coded values or use pre-built functions such as concat, firstName, lastName, gender, random, email, and phone, and options to randomize the output within the required criteria.

For example, in the scenario mentioned earlier, a Kony app developer can create the service that points to an endpoint URL along with a stub template. The stub response template can be set for each operation of a service. The app developers can continue to develop apps based on a **sample back-end response from the stub template**. The service can be modified to connect to the configured backend URL or to send a stub response based on a setting.

> ***Note:*** Stubbing is supported only for JSON, XML, and SOAP Integration services.

> ***Important:*** Services built with Stub Backend Response will give mock data and live data as these services can be switch between live backend and mock data.

> ***Note:*** nd response is used by the **Data Panel** feature of Kony Visualizer V8 SP3 GA. For more information on Data Panel, click here.

21.6.6.1 Stub Template

The back-end stubbed response specifies a Stub template for stubbing and returning dynamically-generated mock data, instead of connecting to the back end and getting the data from the back-end URL.

The following table details a sample Stub response template and a back-end response that is generated based on the stub template.

Sample Stub Template	Sample Stub Response

[
'{{repeat(1,2)}}',
{
"locationID": "{{index()}}",
"company": "{{toUpperCase(company())}}",
"phone": "+1 {{phone()}}",
"address": "{{integer(100, 999)}} {{street()}}, {{city()}}, {{state()}}, {{integer(100, 10000)}}",
"latitude": "{{float(-90.000001, 90)}}",
"longitude": "{{float(-180.000001, 180)}}",
"office":"{{random("HR Head Office","Sales Head Office","Marketing Head Office","Development Center")}}"
}
]
	[{
	"locationID": "0",
	"company": "RODEOMAD",
	"phone": "+1 371-222-9269",
	"address": "671 Division Place, Grill, South Dakota, 9220",
	"latitude": "-0.29528046",
	"longitude": "159.72824",
	"office": "Marketing Head Office"
	},
	{
	"locationID": "1",
	"company": "ACME",
	"phone": "+1 311-324-8984",
	"address": "257 Adam Place, McCoy, South Carolina, 21245",
	"latitude": "-0.23528046",
	"longitude": "124.72824",
	"office": "Sales Head Office"
	}]

21.6.6.2 Pre-built Functions Supported in Stub Template

The following list of the **sample pre-built functions** are supported in the Stub response template:

repeat(<lower_value>, <upper_value>)

Description	**repeat** function repeats JSON or XML objects randomly based on the **value range** provided in the syntax. This is typically set at the beginning of a collection to repeat number of times as required. For example, it can be set to get a random number of transactions for an account for a set format by setting repeat function at the head of the collection as shown in the default template above.
Syntax	`'{{repeat (<lower_value>, <upper_value>)}}'`
Sample stub template	`'{{repeat(30,40)}}'`

repeat(<number>)

Description	**repeat function repeats JSON or XML objects randomly based on the fixed number** provided in the syntax. This is typically set at the beginning of a collection to repeat number of times as required. For example, it can be set to get a random number of transactions for an account for a set format by setting repeat function at the head of the collection as shown in the default template above.
Syntax	`'{{repeat (<number>)}}'`
Sample stub template	`'{{repeat(2)}}'`

integer(min,max)

Description	Generates a random integer in the specified range.
Syntax	`"{{integer(min,max)}}"`
Sample stub template	`"productRating":"{{integer(0,5)}}"`

Sample stub response	`"productRating": "4"`

float(min,max)

Description	Generates a random 32-bit floating point number in the specified range.
Syntax	`"{{float(min,max)}}"`
Sample stub template	`"floatrange": "{{float(3,9)}}"`
Sample stub response	`"floatrange": "6.718242"`

float(min,max,"%.2f")

Description	Generates a random 32-bit floating point number in the specified range of floating point numbers, with an option to round of the number of decimal places.
Syntax	`"{{float(min,max,"%.2f")}}"`
Sample stub template	`"floatrange": "{{float(3,9,"%.2f")}}"`
Sample stub response	`"floatrange": "8.87"`

double(min,max)

Description	Generates a random 64-bit double number in the specified range.
Syntax	`"{{double(min,max)}}"`
Sample stub template	`"double": "{{double(2,8)}}"`
Sample stub response	`"double": "7.654668228367652"`

long(min,max)

Description	Generates a random long number in the specified range.

Syntax	`"{{long(min,max)}}"`
Sample stub template	`"network":"{{long(200,500)}}"`
Sample stub response	`"network":"378"`

uuid()

Description	Generates a random GUID.
Syntax	`"{{uuid()}}"`
Sample stub template	`"objects": "{{uuid()}}"`
Sample stub response	`"objects": "fb81ad08-42e3-4b61-9a2c-636f95952766"`

hex()

Description	Generates a random 16 bytes hexadecimal string.
Syntax	`"{{hex()}}"`
Sample stub template	`"color": "{{hex()}}"`
Sample stub response	`"color": "b53ff7fa4b63b18cdf7729e85c39e660"`

hex(size)

Description	Generates a random hexadecimal string according to the specified size in bytes.
Syntax	`"{{hex(size)}}"`
Sample stub template	`"color": "{{hex(2)}}"`
Sample stub response	`"color": "1b41"`

objectId()

Description	Generates a hexadecimal string of size 12 bytes.
Syntax	`"{{objectId()}}"`
Sample stub template	`"id": "{{objectId()}}"`
Sample stub response	`"id": "1c91293a4c777000585e04e6"`

bool()

Description	Generates a random Boolean value, either True or False.
Syntax	`"{{bool()}}"`
Sample stub template	`"stockAvailable": "{{bool()}}"`
Sample stub response	`"stockAvailable": "true"`

bool(<probability>)

Description	Generates a random Boolean value, either True or False as per the given probability.
Syntax	`"{{bool(probability)}}"`
Sample stub template	`"stockAvailable": "{{bool(0.9)}}"`
Sample stub response	`"stockAvailable": "true"`

index()

Description	Generates an incrementing index integer for each record with a specific starting point.
Syntax	`"{{index()}}"`
Sample stub template	`"locationID": "{{index()}}"`

Sample stub response	`"locationID": "0"`

index("index-name")

Description	Generates an incrementing index integer for each record based on the name of the index.
Syntax	`"{{index("index-name")}}"`
Sample stub template	`"index-number": "{{index("abc")}}"`
Sample stub response	`"index-number": "42"`

index(<number>)

Description	Generates an incrementing index integer for each record with a specific starting point.
Syntax	`"{{index(78)}"`
Sample stub template	`"index-number": "{{index(78)}}"`
Sample stub response	`"index-number": "4248"`

index("index-name",<number>)

Description	Generates an incrementing index integer for each record based on both a specific starting point and name of the index.
Syntax	`"{{index("index-name",78)}"`
Sample stub template	`"index-name-number": "{{index("abc",78)}}"`
Sample stub response	`"index-name-number": "626"`

lorem(count,"words")

Description	Generates a random dummy text. User must specify the count of words required.
Syntax	`"{{lorem(count,"words")}}"`
Sample stub template	`"productDescription": "{{lorem(5, "words")}}"`
Sample stub response	`"productDescription": "lorem ipsum porta sit curabitur"`

lorem(count,"paragraphs")

Description	Generates a random dummy paragraph. User must specify the count of paragraphs required.
Syntax	`"{{lorem(count,"paragraphs")}}"`
Sample stub template	`"about": "{{lorem(2, "paragraphs")}}"`

Sample stub response	`"about": " Lorem ipsum eros amet accumsan non quisque ut molestie nullam sagittis tincidunt.Lorem ipsum quis aliquam nostra.`
	
	`Lorem ipsum litora tristique arcu habitant.Lorem ipsum nulla mauris inceptos fusce adipiscing tortor torquent."`

phone()

Description	Generates a random phone number. The phone number is preceded by `+` to indicate a country code. This allows to set phone numbers for different countries. For example, for USA/Canada, the phone number format is `+1 xxx xxx xxxx`

Syntax	`"{{phone()}}"`
Sample stub template	`"phone": "+1 {{phone()}}"`
Sample stub response	`"phone": "+1 371-222-9269"`

gender()

Description	Generates a random gender value, either male or female.
Syntax	`"{{gender()}}"`
Sample stub template	`"gender": "{{gender()}}"`
Sample stub response	`"gender": "female"`

date()

Description	Generates the current date.
Syntax	`"{{date()}}"`
Sample stub template	`"date": "{{date()}}"`
Sample stub response	`"date": "Tue, 11 Sep 2018 10:55:44 GMT"`

date("java-simple-date-format")

Description	Generates the current date in the specified date format.
Syntax	`"{{date("java-simple-date-format")}}"`
Sample stub template	`"date_format": "{{date("dd-MM-yyyy HH:mm:ss")}}"`
Sample stub response	`"date_format": "11-09-2018 12:00:00"`

date("begin-date","end-date","java-simple-date-format")

Description	Generates a random date in the specified range and specified format. Your date range input must be in this format: `dd-MM-yyyy HH:mm:ss`
Syntax	`"{{date("begin-date","end-date","java-simple-date-format"}}"`
Sample stub template	`"{{date("01-01-2014 12:00:00", "01-01-2018 12:00:00", "yyyy-MM-dd'T'HH:mm:ss Z")}}"`
Sample stub response	`"date_of_joining": "2015-12-22T22:00:33 +0000"`

date("begin-date","end-date")

Description	Generates a random date in the specified range of dates with default format. Your input must be in this format `EEE, d MMM yyyy HH:mm:ss z`
Syntax	`"{{date("begin-date","end-date")}"`
Sample stub template	`"date_default_format": "{{date("01-01-2014 12:00:00", "01-01-2018 12:00:00", "EEE, d MMM yyyy-MM-dd'T'HH:mm:ss Z")}}"`
Sample stub response	`"date_default_format": "Tue, 21 Jun 2016-06-21T19:26:18 +0000"`

timestamp()

Description	Generates the current timestamp (milliseconds, between the current time and midnight, January 1, 1970 UTC):
Syntax	`"{{timestamp()}}"`
Sample stub template	`"time": "{{timestamp()}}"`

Sample stub response	`"time": "1536662962710"`

timestamp("begin-date","end-date")

Description	Generates the current timestamp (milliseconds, between the current time and midnight, January 1, 1970 UTC) between two dates with default format. Your input must be in this format `EEE, d MMM yyyy HH:mm:ss z`
Syntax	`"{{timestamp("begin-date","end-date"}}"`

country()

Description	Generates a random country name.
Syntax	`"{{country()}}"`
Sample stub template	`"country": "{{country()}}"`
Sample stub response	`"country": "Montenegro"`

countryList()

Description	Generates a JSON mapping with all country codes and country name.
Syntax	`"{{countryList()}}"`

countryList("country_code_1", "country_code_2")

Description	Generates a JSON mapping with given country codes and country name.
Syntax	`"{{countryList("IN", "US", "UK")}}"`

city()

Description	Generates a random city.
Syntax	`"{{city()}}"`

Sample stub template	`"city": "{{city()}}"`
Sample stub response	`"city": "Belva"`

state()

Description	Generates a random state name.
Syntax	`"{{state()}}"`
Sample stub template	`"state": "{{state()}}"`
Sample stub response	`"state": "Pennsylvania"`

company()

Description	Generates a random company name.
Syntax	`"{{company()}}"`
Sample stub template	`"company": "{{company()}}"`
Sample stub response	`"company": "Gorganic"`

lastName()

Description	Generates a random last name.
Syntax	`"{{lastName()}}"`
Sample stub template	`"lastname": "{{lastName()}}"`
Sample stub response	`"lastname": "Randolph"`

firstName()

Description	Generates a random first name.
Syntax	`"{{firstName()}}"`

Sample stub template	`"firstname": "{{firstName()}}"`
Sample stub response	`"firstname": "Mays"`

username()

Description	Generates a random username based on the first initial of your random first name and random last name in lowercase.
Syntax	`"{{username()}}"`
Sample stub template	`"username": "{{username()}}"`
Sample stub response	`"username": "nbarr"`

email()

Description	Generate a random email address in the standard format. For example: the email standard format is `<firstName>.<lastName>@<domain>.<com>`
Syntax	`"{{email()}}"`
Sample stub template	`"email": "{{email()}}"`
Sample stub response	`"email": "irma.england@mazuda.com"`

email("mydomain.com")

Description	Generates a random email address with the specified domain name in the standard format. For example: the email standard format is `<firstName>.<lastName>@<"mydomain.com">`
Syntax	`"{{email("mydomain.com")}}"`
Sample stub template	`"email": "{{email("mydomain.com")}}"`

Sample stub response	`"email": "wilson.allison@mydomain.com"`

ssn()

Description	Generates a random social security number.
Syntax	`"{{ssn()}}"`
Sample stub template	`"ssn": "{{ssn()}}"`
Sample stub response	`"ssn": "285-59-5039"`

ipv4()

Description	Generates a random ipv4 address.
Syntax	`"{{ipv4()}}"`
Sample stub template	`"ipv4": "{{ipv4()}}"`
Sample stub response	`"ipv4": "218.110.1.153"`

ipv6()

Description	Generates a random ipv6 address.
Syntax	`"{{ipv6()}}"`
Sample stub template	`"ipv6": "{{ipv6()}}"`
Sample stub response	`"ipv6": "ipv6": "e801:bde0:c898:3a0e:2401:1b23:2199:4d3f"`

ipv6("upper")

Description	Generates a random ipv6 address with all the alphabetical characters in uppercase.
Syntax	`"{{ipv6("upper")}}"`

ipv6("lower")

Description	Generates a random ipv6 address with all the alphabetical characters in lowercase.
Syntax	`"{{ipv6("lower")}}"`

concat(var arg)

Description	Generates a string by concatenating all the strings given as inputs.
Syntax	`"{concat("A","B","C","D")}}"`
Sample stub template	`"id": "{{concat("EMP",index())}}"`
Sample stub response	`"id": "EMP2942"`

substring("word",3)

Description	Generates a substring from the given string and the starting position.
Syntax	`"{{substring("word",3)}}"`
Sample stub template	`"substring":"{{substring("SampleApps", 3)}}"`
Sample stub response	`"substring":"pleApps"`

substring("long word", 1, 6)

Description	Generates a substring from the given string, the starting position, and the end position.
Syntax	`"{{substring("long word", 1, 6)}}"`

Sample stub template	`"substring":"{{substring("Sampleword" , 1,6)}}"`
Sample stub response	`"substring":"ample"`

random("SampleValue1","SampleValue2","SampleValue3","SampleValue4")

Description	**random** function provides string values to each record randomly based on the predefined sample string values provided while invoking the function.
Syntax	`"{{random ("SampleValue1","SampleValue2","SampleValue3","SampleValue4")}}"`
Sample stub template	`"office":"{{random("HR Head Office","Sales Head Office","Marketing Head Office","Development Center")}}"`
Sample stub response	`"office": "Marketing Head Office"`

alpha()

Description	Generates a random string with alphabetic characters of length between 10 to 20 characters.
Syntax	`"{{alpha()}}"`
Sample stub template	`"alpha": "{{alpha()}}"`
Sample stub response	`"alpha": "jFulYDTuFQBk"`

alpha(min,max)

Description	Generates a random string with alphabetic characters and length in the given range.
Syntax	`"{{alpha(min,max)}}"`
Sample stub template	`"alphaRange": "{{alpha(15,20)}}"`
Sample stub response	`"alphaRange": "hTyLpLYMHGJkYrEB"`

alpha(length)

Description	Generates a random string with alphabetic characters of given length.
Syntax	`"{{alpha(length)}}"`
Sample stub template	`"alphaLength": "{{alpha(7)}}"`
Sample stub response	`"alphaLength": "ZKziDST"`

alphaNumeric()

Description	Generates a random string with alpha-numeric characters of length between 10 to 20 characters.
Syntax	`"{{alphaNumeric()}}"`
Sample stub template	`"alphaNumeric": "{{alphaNumeric()}}"`
Sample stub response	`"alphaNumeric": "hIrUkxDzdH4VIR86g6G"`

alphaNumeric(min,max)

Description	Generates a random string with alpha-numeric characters of length in the specified range.
Syntax	`"{{alphaNumeric(min,max)}}"`

Sample stub template	`"alphaNumericRange": "{{alphaNumeric (15,20)}}"`
Sample stub response	`"alphaNumericRange": "k5BBh4U45o19vhaO4LQ"`

alphaNumeric(length)

Description	Generates a random string with alpha-numeric characters of the given length.
Syntax	`"{{alphaNumeric(length)}}"`
Sample stub template	`"alphaNumericLength": "{{alphaNumeric(7)}}"`
Sample stub response	`"alphaNumericLength": "Zphdrag"`

toLowerCase("text")

Description	Converts the string value to lowercase letters.
Syntax	`"{{toLowerCase(company())}}"`
Sample stub template	`"company": "{{toLowerCase(company())}}"`
Sample stub response	`"company": "rodeomad"`

toUpperCase("text")

Description	Converts the string value to uppercase letters.
Syntax	`"{{toUpperCase(company())}}"`
Sample stub template	`"company": "{{toUpperCase(company())}}"`
Sample stub response	`"company": "RODEOMAD"`

##Escape braces

Description	If you want to escape braces from within a function use a single escape character as seen in the example below:
Syntax	`"{{concat("\{", "test", "\}")}}`

##XML support

Description	Stub template supports XML format.
Syntax	<?xml version="1.0" encoding="UTF-8"?> <root> '{{repeat(2)}}', <element> <id>{{guid()}}</id> <name>{{firstName()}}</name> <index>{{lastName()}}</index> </element> <tags>'{{repeat(7)}}', {{lorem(1, "words")}}</tags> <friends> '{{repeat(3)}}', <friend> <id>{{index()}}</id> <name>{{firstName()}} {{surname()}} </name> </friend> </friends> </root>

##Nesting functions

Description	Stub service supports nesting functions as well. For example, if you wanted to create results that looked like dollar amounts, you can do the following:

Syntax	`"{{concat("$",float(0.90310, 5.3421, "%.2f"))}}`
	
	or something like this if you wanted a capitalized F or M:
	
	`{{toUpperCase(substring(gender(),0,1))}}`

21.6.6.3 How to Enable Stub Back-end Response

1. In the **Integration > Operations > Advance** section, select the **Stub Backend Response** check box. The text box is enabled with a sample Stub response template.

2. Configure your stub template in the provided field.

3. Click **Test**. Kony Fabric generates the backend response based on the Stub template.

4. Publish the app. After you publish the app, you can test the stub response by either using Admin Console, a client application, or Kony Fabric.

21.6.6.4 How to Test a Stub Response from Admin Console

1. Publish your app to a runtime server.
2. Go to the runtime server in **Admin Console**.
3. Go to the **Integration Services** tab.
4. For the stub service that you created, select the stubbed operation from the **Operations** list.
5. Click **Get Response**.

> ***Note:*** The `X-Kony-Stub-Response header` as `true` in the back-end response indicates that the response is generated from the Stub template, and not from the actual back end.

21.6.6.5 Advanced Parameters in Stub Response

How to Configure a Request Input and Request Header Parameters in Stub Template

You can access input parameters in a Stub template by using the `{{requestBody("<request_param_name>")}}` function. Additionally, you can access headers by using the `{{requestHeader("<header_name>")}}` function.

For example, you want to send the `testUser` request input parameter to the Stub template and the value of the parameter is defined in the Input Parameters section of Console. You can access the input parameter in the stub template as `"inputtest": "{{requestBody("testUser")}}"`.

The following sample Stub template has been configured with the testUser request input parameter.

```
[
 '{{repeat(30,40)}}',
 {
   "locationID": "{{index()}}",
```

```
    "company": "{{toUpperCase(company())}}",
    "phone": "+1 {{phone()}}",
    "address": "{{integer(100, 999)}} {{street()}}, {{city()}},
{{state()}}, {{integer(100, 10000)}}",
   "inputtest": "{{requestBody("testUser")}}",
    "latitude": "{{float(-90.000001, 90)}}",
    "longitude": "{{float(-180.000001, 180)}}",
    "office":"{{random("HR Head Office","Sales Head Office","Marketing
Head Office","Development Center")}}"
  }
]
```

> ***Note:*** Kony functions for request input and header are as follows:
>
> - To access any header with name, `requestHeader`: Syntax: `{{requestHeader ("<header_name>")}}`
> - To access a request parameter, `requestBody`: Syntax: `{{requestBody("<request_param_name>")}}`

How to Configure X-Kony-Stub-Request Header

At runtime, when the actual back-end service is available, and you still want to test the Stub response, you can enable the Stub response feature by sending the `X-Kony-Stub-Request header` parameter as request input.

> ***Important:*** For example, You have created a Stub response, but have not selected the **Stub Backend Response** check box, and published the back-end app. When you send the `X-Kony-Stub-Request header` parameter in the request input, even if the stub back response is set to false, you still get the Stub response.

> ***Note:*** You can send headers with values in the SDKs for service level. For example, `var headers = {"your-header-keys" : "your-header-values"};`
>
> For more information on how to configure headers in SDKs, refer Kony Visualizer SDKs > Invoking an Integration Service

How to Configure a Global Request Parameter to enable Stubbing from an Application

If you want to enable stubbed response from all services in an app, even if the service is not published with stubbing enabled, you can do so by using the `setGlobalRequestParam` feature from Kony Fabric SDK.

Consider a scenario where you want to switch between live and stubbed services during app development and you have an option from the app to switch between services. Here, this feature helps you to effectively troubleshoot in case there is any issue with the back-end service data or with the application code.

Syntax:

```
setGlobalRequestParam(paramName, paramValue, aramType);
```

For more information on Global Request Parameter - SDKs, refer Kony Visualizer SDKs > Initializing the Kony Client SDK > setGlobalRequestParam, and search for `setGlobalRequestParam`.

How to Disable Stubbing in an Environment

When you want to use a back-end service at all times and want to disable the Stub response feature at the environment-level (such as production environment) to avoid Stub response in a non-development environment, you can set the `KONY_SERVER_DISABLE_ALL_STUB_RESPONSE` configuration property with the value as `true` (default value is false) as `-D param` or in the `server_configuration` table of the Admin database.

21.6.7 Enhanced Identity Filters

Identity filters are an enhanced data filtering mechanism that you can use to filter data for a mobile app based on dynamic fields returned from an identity provider.

A backend response returned from an identity service has two kinds of attributes **profile attributes** and **security attributes:**

- **Profile attributes** refers to the attributes such as First Name, Email, Phone Number, which identify a user.
- **Security attributes** are associated with a session such as tokens required to make API calls to a backend, for example, `access_token`.

Identity filters are supported for the following service types:

- Enhanced Identity Filters - Integration services
- Enhanced Identity Filters in Objects > Storage Services and Service-Driven Objects

21.6.7.1 Enhanced Identity Filters - Integration Services

For identity filters to work, you must protect an integration service with an identity provider. For example, if an identity provider responds with a profile that has `userid` as one of its attributes after a successful login, use `profile.userid` as a value to pass it as request parameter to the backend. The response is filtered based on the value mentioned in `userid`.

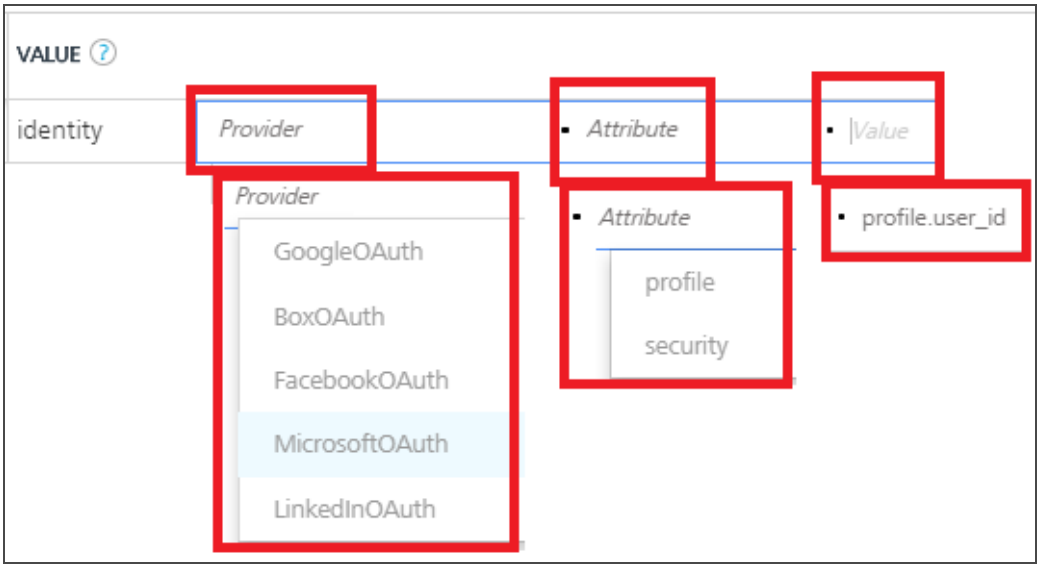
When an integration service is protected with the identity provider, the operations of the integration service are enabled with an **identity** filter.

To configure identity filters to an integration service, follow these steps:

1. Create an integration service.
2. Create an operation for the integration service. For example, **IntIdt**

3. In the operation's **Request Input** > **Body** tab, do the following:
    a. In the **NAME** field, enter the name for the request input parameter - for example, **identityuserid**.
    b. Click the drop-down menu in the **Value** column, and select **identity**.

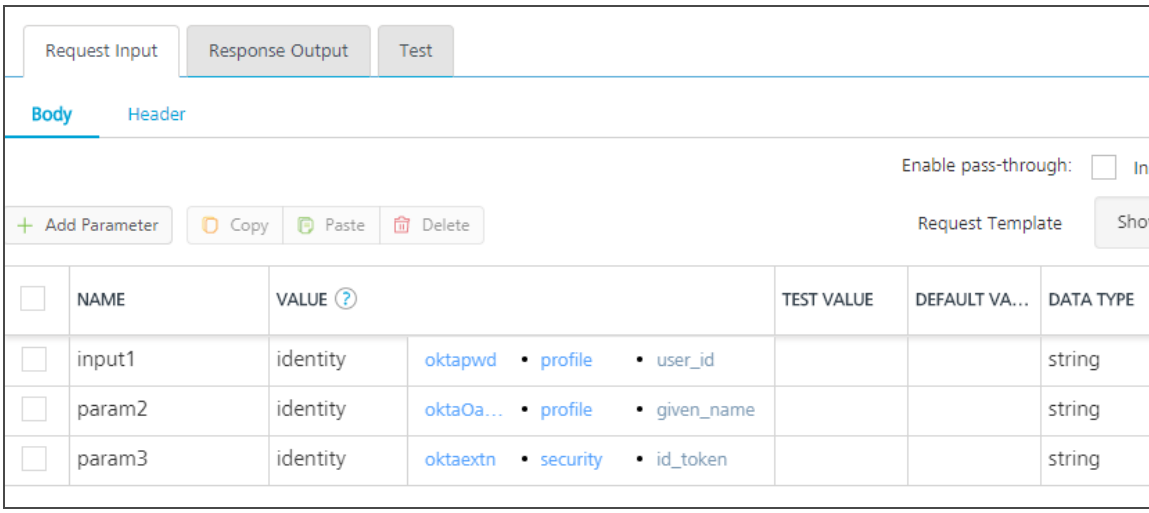
       An identity value indicates that Kony Fabric will retrieve the value specified from the user's security profile in the identity service that is linked to the object service.
    c. From the drop-down list next to the **identity** value, do the following:
        i. Select the type of the identity provider from the drop-down list. For example, `GoogleOAuth` or `FacebookOAuth`.



        ii. Select the type of the attribute from the drop-down list. For example, `profile` or `security`,
        iii. Next to the attribute, enter the attribute key as obtained from a backend, for example, `profile.user.id`.

- Add the following details.

NAME	VALUE	
identityuserid	identity	profile.user_id
sessionparam	identity	security.access_token
identityname	identity	profile.name



d. Click **SAVE** to save the operation.

4. In the operation's **Request Input > Header** tab, do the following:
   a. In the NAME field, enter the name for the request input parameter -for example, **identityuseridheader**.
   b. Click the drop-down menu in the **Value** column, and select **identity**.
   c. From the drop-down list next to the **identity** value, select **profile/security**, and then enter **profile.user.id**.

Add the following details.

NAME	VALUE	
identityuseridheader	identity	profile.user_id
sessiontokenheader	identity	security.access_token
profilenameheader	identity	profile.name

d. Click **SAVE** to save the operation.

> ***Note:*** In case you want to use identity filters with a custom code, you can define JavaScript **preprocessor** and **postprocessor**.

21.6.8 Collection Support

Collection Support in Request Input for SOAP

You can now map collections within your SOAP request to your application. If you want your collection to be displayed for certain iterations, you can also add a `for loop` to your SOAP contract. IDE now supports n level nested collections as a part of the request template.

An example of a single level collection is as follows:

```
 <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:ZFmmmGoodsReceiptKony>
         <!--Optional:-->
         <IGrCreate>
```

```
            <GRHeader>
            <HId>$HId</HId>
            <HTime>$HTime</HTime>
            <HUser>$HUser</HUser>
            <!--Add the for loop for Zero or more repetitions:-->
#foreach $items
            <item>
                                    <Bldat>$Bldat</Bldat>
                                    <Budat>$Budat</Budat>
                                    <Mtsnr>$Mtsnr</Mtsnr>
                                    <Usnam>$Usnam</Usnam>
                                    <PoNumber>$PoNumber</PoNumber>
                                    <PoItem>$PoItem</PoItem>
                                    <PoUnit>$PoUnit</PoUnit>
                                    <QuantityRec>$QuantityRec</QuantityRec>
                                    <UnloadPoint>$UnloadPoint</UnloadPoint>
                                    <Xsaut>$Xsaut</Xsaut>

                        </item>
           #end

<!--end of loop:-->
          </GRHeader>
        </IGrCreate>
     </urn:ZFmmmGoodsReceiptKony>
   </soapenv:Body>
</soapenv:Envelope>
```

You can similarly have multiple nested collections and you can define a for loop for each of the nested collections. Ensure that each of the *for loop* has a unique identifier name. For example `#foreach $<uniqueidentifier>`. Since Kony Studio parses the items and creates a service definition template using end of line logic for each *for statement* till for *end of loop*.

```
 <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:ZFmmmGoodsReceiptKony>
         <!--Optional:-->
         <IGrCreate>
            <GRHeader>
            <HId>$HId</HId>
            <HTime>$HTime</HTime>
            <HUser>$HUser</HUser>
            <!--Zero or more repetitions:-->
          #foreach $items
            <item>
                                 <Bldat>$Bldat</Bldat>
                                 <Budat>$Budat</Budat>

                                 <Serialno>
                                        <!--Zero or more repetitions:-->
                                     #foreach $SerialNo
                                        <item>
                                              <Gernr>$Gernr</Gernr>
                                              <UnloadPoint>$UnloadPoint</Unload

                                              #foreach $test
                                                    <XXX>$a</XXX>
                                              #end
                                        </item>
                                     #end
                                 </Serialno>

                        </item>
```

```
           #end
          </GRHeader>
        </IGrCreate>
     </urn:ZFmmmGoodsReceiptKony>
   </soapenv:Body>
</soapenv:Envelope>
```

Scenario 1: If the input value for an element is not provided, you can include an exclamation mark "!" after the "$" (for example, `<Bldat>$!Bldat</Bldat>`) to generate a collection that includes null value. For example: `<Bldat></Bldat>`

To avoid this empty <Bldat></Bladat> tag, you can use the following if condition statement:

```
 #if ($Bldat)
<Bldat>$!Bldat</Bldat>
#end
```

Scenario 2: If a Key is missing in the input parameter, write the template in the following manner and past it in the request tab:

```
 <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:ZFmmmGoodsReceiptKony>
         <!--Optional:-->
         <IGrCreate>
            <GRHeader>
            <HId>$HId</HId>
            <HTime>$HTime</HTime>
```

```
            <HUser>$HUser</HUser>
            <!--Zero or more repetitions:-->
           #foreach ($rec in $items.records)  #set( $Budat =
$rec.getParam("Budat").getEscapeXMLValue())  #set( $Bldat =
$rec.getParam("Bldat").getEscapeXMLValue())  #set( $SerialNo =
$rec.getDatasetById("SerialNo"))
            <item>
                                  <Bldat>$Bldat</Bldat>
                                  <Budat>$Budat</Budat>

                                  <Serialno>
                                        <!--Zero or more repetitions:-->
                                     #foreach ($rec in $SerialNo.records)  #set
$rec.getParam("Gernr").getEscapeXMLValue())  #set( $UnloadPoint =
$rec.getParam("UnloadPoint").getEscapeXMLValue())
                                        <item>
                                              <Gernr>$Gernr</Gernr>
                                              <UnloadPoint>$UnloadPoint</Unload
                                        </item>
                                     #end
                                  </Serialno>

                        </item>
           #end
          </GRHeader>
        </IGrCreate>
     </urn:ZFmmmGoodsReceiptKony>
   </soapenv:Body>
</soapenv:Envelope>
```

After you have structured your xml, do the following:

1. Copy and paste the xml in the **Request** tab.
2. Click **Add**. A row gets added to the table.
3. Enter the value for **ID**. The ID should be the same as that of loop of the collection. For example if the loop is *#foreach $items* the corresponding ID should be items
4. Enter a **Test Value**.

   Test Value is the value of the parameter that is used for testing the service. The test value for a collection could be of the form:

   ```
   [{"Bldat":"123","Budat":"adf","SerialNo":
   [{"Gernr":"AAAA","UnloadPoint":"adadad","test":[{a:abc},{a:def}]}
   ,{"Gernr":"BBBB","UnloadPoint":"adadad","test":[{a:abc},
   {a:def}]}]}]
   ```

   For an item whose value whose value is empty specify it as [].
5. Select a value for **Scope** from the drop-down list. **Scope** denotes the scope of the parameter, if it is limited to *request* or a *session*.
6. Set the value of the **Datatype** as collection.
7. Select True or False for **Encode**. Specifies if the URL needs to be encoded or not
8. If you need to delete a parameter row, select the parameter row you want to delete and click **Delete**. This action removes the parameter row from the list of service parameters.
9. Navigate to the response pane and fetch the response.

Collection Support in Request Input for JSON

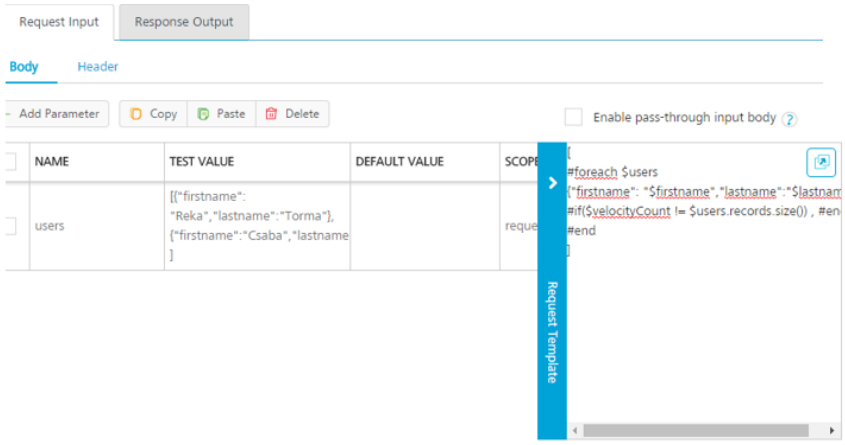
You can now map collections within your JSON request to your application.

If you want your collection to be displayed for certain iterations, you can also add a `for loop` to your JSON input.

Kony Fabric Console supports multilevel (N level) nested collections as part of the request template.

Example of a simple single level collection is as follows:

```
 {
"users": [ #foreach $users
#if($velocityCount!=1)
,
#end
{"firstname": "$firstname","lastname":"$lastname"} #end ]
}
```



Note that you have to change the datatype to collection in Kony Fabric Console for the added input parameter. To test, pass a simple single level JSON.

For example:

```
 [{ "firstname": "Sam", "lastname": "Tim"  }, { "firstname":
"Jim","lastname": "Jerry" }]
```

> ***Note:*** For nested collections, please go through the SOAP documentation for collection.

Collection Support in Request Input for XML

You can now map collections within your XML request to your application.

If you want your collection to be displayed for certain iterations, you can also add a `for loop` to your XML input.

Kony Fabric Console supports multilevel (N level) nested collections as part of the request template.

Example of a simple single level collection is as follows:

```
#foreach $items
            <item>
                                    <Bldat>$Bldat</Bldat>
                                    <Budat>$Budat</Budat>
                                    <Mtsnr>$Mtsnr</Mtsnr>
                        </item>
           #end
```

And you can pass an input collection with a test value:

```
 [{"Bldat":"123","Budat":"adf","Mtsnr":"dsd"},
{"Bldat":"234","Budat":"ffg","Mtsnr":"aff"},
{"Bldat":"677","Budat":"fdf","Mtsnr":"jft"}]
```

21.6.9 Custom Front End URL

21.6.9.1 Resource URL/Front End URL

From Kony Fabric V8 SP3 onwards, using Kony Fabric, you can map your endpoint/back-end URL of an operation to a front-end URL. The front-end URL of an operation contains custom details, which does not expose in the original details of the back-end URL. So, you can use the front-end URL to send request to the back-end securely, instead of the original back-end URL.

For example:

- The following is a sample back-end URL of an operation to send a request:

  ```
  BaseURL/Services/<Integration_Service_Name>/<Operation_Name>
  ```

- For example,

```
http://news.google.com/Services/<Integration_Service_Name>/<Operation_Name>
```

- The following is a sample **front end URL** of an operation to send a request:

```
/Services/<Integration_Service_Name>/<Operation_Path>
```

> ***Note:*** To walk-through defining front-end URLs for service APIs with Kony Fabric, take a look at our hands-on tutorial for API Management - Front End URLs.

Advantages:

- A front-end URL contains a custom data, which does not expose the original data of the back-end operation.
- You can map more than one front-end URLs to one base back-end URL and perform different operations. For example, you can configure four of the front-end URLs to perform CRUD operations based on the same back-end URL.

> ***Note:*** By default, the **Front End URL** check box is not selected. And the URL path (populated with an operation name by default) in the **Resource Path** field is disabled from editing.

To enable a front end URL, do the following:

1. Create an operation for an in integration service.
2. In the Operation configuration page, in the **Advanced** section, select the **Front End URL** check box. The **Resource Path** field is editable now.

3. Configure the front-end URL in the **Resource Path** field.

   By default, the **Resource Path** field is configured with the `/Services/<Integration_Service_Name>/<Operation_path>`.

   The `<Operation_Path>` is the operation name. You can add request input parameters to the path as verbs. This is optional.

4. Configure `request input parameters` as `verbs` in a in curly braces ({ }) to the **Operation_path**, if required.

   For example:

   ```
   /Services/<Integration_Service_Name/<Operation_Name>/{Request_Input_Param}
   ```

5. Save the operation.

> ***Important:*** If you clear the **Front End URL** check box, the Console resets the default value in the URL path. And you cannot access the endpoint by using the **front-end URL** that you configured earlier.

The Front-end URL is divided into two parts.

/Services/<Integration_Service_Name>/	<Operation_Path>
• You cannot edit the value in the `/Services/`. • The `/<Integration_Service_Name>/` is the auto-generated name as per the name that you specify for the integration service.	• You can enter the appropriate name for the front-end URL of the operation. ***Note:*** An `Operation_Path` can be `/<Operation_Name>` or `/Operation_name>/{Request_Input_param}`

21.6.9.2 Resource Method/Front End HTTP Method

You can select which HTTP method to invoke on the integration server. Select the Front End HTTP Method field. By default, the field is set to Post method.

> ***Note:*** The front-end HTTP methods are used for all non-SDK clients such as API Management users. Invoking a service from an SDK will continue to use the POST method for operations.

21.6.10 XPath in Kony Fabric

XPath (XML Path Language) is a query language for selecting nodes from an XML document. Kony Fabric supports XPath expressions to compute/filter (for example, strings, numbers, or Boolean values) a back-end response that is in XML format. Kony supports XPath for XML, SOAP, and JSON integration services.

For more information about XPath, refer https://www.w3schools.com/xml/xml_xpath.asp.

21.6.10.1 XPath supported operators in Kony Fabric

The following table details the supported operators for XPath expressions in Kony Fabric.

Operators	Description
AND	Boolean and
OR	Boolean or
+	Plus
-	Minus
*****	Multiply
Div	Division

Operators	Description
Mod	Modulus (division remainder)
Sum	This converts the value of each node in the node-set to a number and totals the result.
Round	This returns the closest integer to the argument. The rounding rules follow Java conventions which are not quite the same as the XSL rules.

- While configuring XPath for a response, for arithmetic XPath expressions, operators and operands should be separated by at least one space ( ).

  For example,

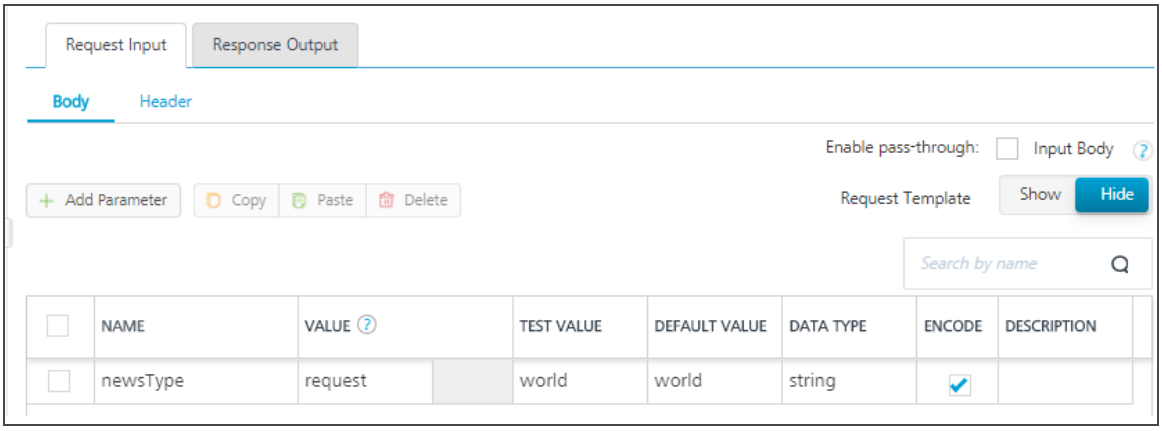
```
 <? xml version="1.0" encoding="UTF-8”?>
<CATALOG>
<PLANT>
<ZONE>4</ZONE>
<PRICE>2.44</PRICE>
</PLANT>
</CATALOG>

*Sample arithmetic operations with at least one space:*

//CATALOG/PLANT/ZONE + //CATALOG/PLANT/PRICE
//CATALOG/PLANT/ZONE * //CATALOG/PLANT/PRICE
//CATALOG/PLANT/ZONE div //CATALOG/PLANT/PRICE
```

21.6.10.2 How to use XPath in Kony Fabric

1. Create an integration **service** of an app for XML, SOAP, or JSON.
2. Create an operation.

3. Configure the input parameters in the **Request Input**.

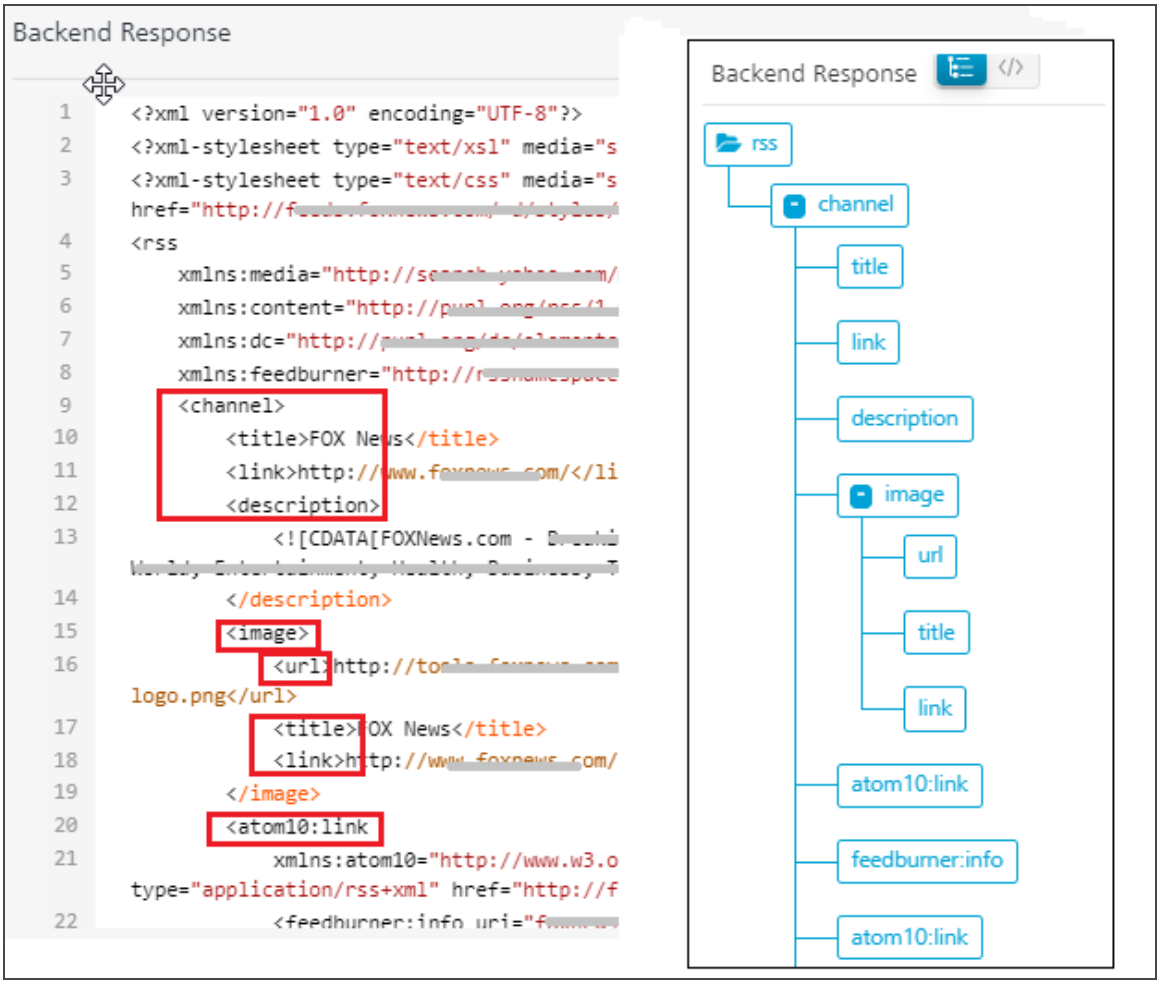
   For example:



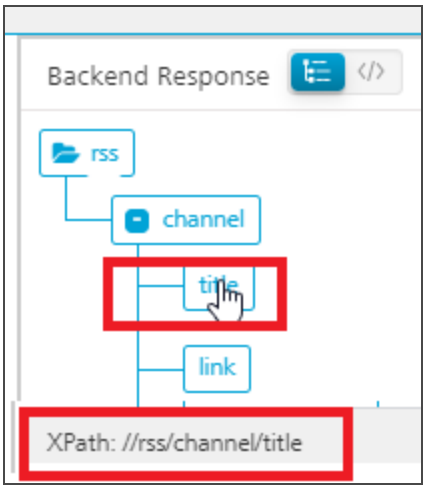
4. Select the environment from the **Select an Environment** list.

5. Click **Save and Fetch Response** to view the results of the operation.

   The back-end response is displayed in the XML format, in the **Test Result** section.
   In **Backend Response** window, you can view the raw response as either raw data or in a tree format.

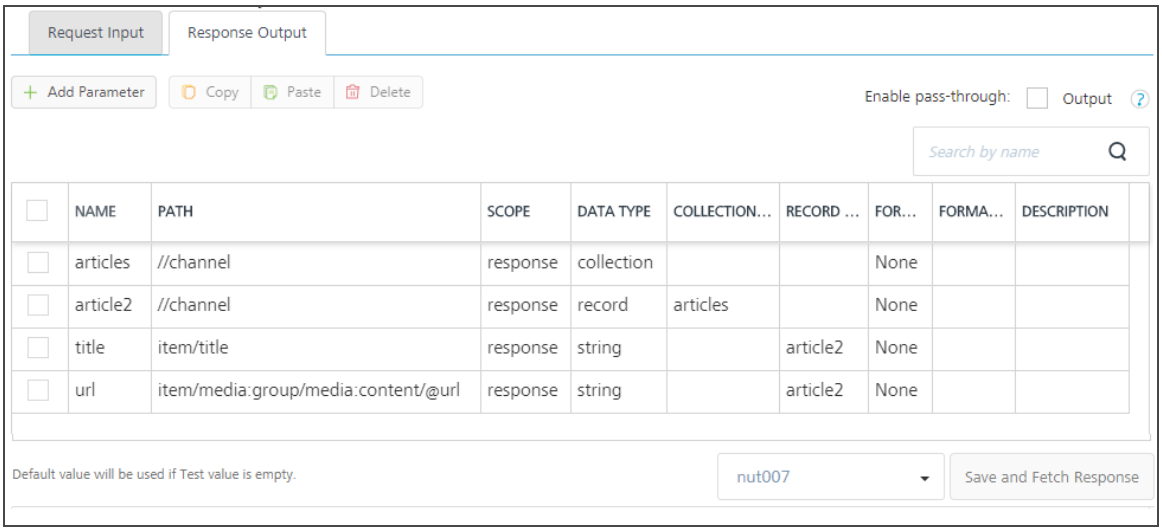


Clicking on an element in the tree to display the XPath of that tag.



6. Now apply XPath expressions for extracting the required elements from the back-end response of the service call.

   For example, the following is a sample XPath expressions



Name/ ID	XPath	Scope	Data Type	Collection ID	Record ID
articles	//channel	response	Collection		
articles2	//channel	response	Record	articles	
title	item/title	response	Sting string		articles2

Name/ ID	XPath	Scope	Data Type	Collection ID	Record ID
url	item/media:group/media:content/@url	response	Sting string		articles2

- **Collection** - A group of data, also referred to as data set. A collection contains only records, and a record contains a string, boolean, or number values.
- **Record** - A group data elements under the specified parameter. A record can also be part of a collection. Typically, a record provides metadata to a segment.

> ***Note:*** For JSON integration service, the back-end response will be in JSON format. JSONPath format should contain `$` in the expression.
>
> For example: `$` the root object/element

7. Click **Save and Fetch Response** again.

   Now the back-end response is filtered based on the XPath expressions and displays the output in JSON format.

XPath Example Reference

The following example details a sample backend response, and XPath configurations and Output Result based on the XPath configurations

- **The following is a sample Backend Response, in XML, from the Target URL of an integration service.**
  Sample Target URL link: http://www.mocky.io/v2/5a9fa4e92e0000630074d133

```
<!-- This is a Sample backend response -->

<catalog>
    <book id="bk101">
        <author>Gambardella, Matthew</author>
        <title>XML Developer's Guide</title>
        <genre>Computer</genre>
        <price>44.95</price>
```

```
        <publish_date>2000-10-01</publish_date>
    </book>
    <book id="bk102">
        <author>Ralls, Kim</author>
        <title>Midnight Rain</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <publish_date>2000-12-16</publish_date>
    </book>
    <book id="bk103">
        <author>Corets, Eva</author>
        <title>Maeve Ascendant</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <publish_date>2000-11-17</publish_date>
    </book>
    <book id="bk104">
        <author>Corets, Eva</author>
        <title>Oberon's Legacy</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <publish_date>2001-03-10</publish_date>
    </book>
    <book id="bk105">
        <author>Corets, Eva</author>
        <title>The Sundered Grail</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <publish_date>2001-09-10</publish_date>
    </book>
    <book id="bk106">
        <author>Randall, Cynthia</author>
        <title>Lover Birds</title>
```

```
        <genre>Romance</genre>
        <price>4.95</price>
        <publish_date>2000-09-02</publish_date>
    </book>
    <book id="bk107">
        <author>Thurman, Paula</author>
        <title>Splish Splash</title>
        <genre>Romance</genre>
        <price>4.95</price>
        <publish_date>2000-11-02</publish_date>
    </book>
    <book id="bk108">
        <author>Knorr, Stefan</author>
        <title>Creepy Crawlies</title>
        <genre>Horror</genre>
        <price>4.95</price>
        <publish_date>2000-12-06</publish_date>
    </book>
    <book id="bk109">
        <author>Kress, Peter</author>
        <title>Paradox Lost</title>
        <genre>Science Fiction</genre>
        <price>6.95</price>
        <publish_date>2000-11-02</publish_date>
    </book>
    <book id="bk110">
        <author>O'Brien, Tim</author>
        <title>Microsoft .NET: The Programming Bible</title>
        <genre>Computer</genre>
        <price>36.95</price>
        <publish_date>2000-12-09</publish_date>
    </book>
    <book id="bk111">
```

```
        <author>O'Brien, Tim</author>
        <title>MSXML3: A Comprehensive Guide</title>
        <genre>Computer</genre>
        <price>36.95</price>
        <publish_date>2000-12-01</publish_date>
    </book>
    <book id="bk112">
        <author>Galos, Mike</author>
        <title>Visual Studio 7: A Comprehensive Guide</title>
        <genre>Computer</genre>
        <price>49.95</price>
        <publish_date>2001-04-16</publish_date>
    </book>
</catalog>
```

- The following table details the XPath for the AND operator and Output Result based on the sample response.

XPath for AND operator	Output Result

`computerBooks: book [genre = 'Computer' and author = "O'Brien, Tim"]`	{
	"httpStatusCode": 200,
	"catalog": [
	{
	"computerBooks": {
	"computerAuthor": "O'Brien, Tim",
	"computerTitle": "Microsoft .NET: The Programming Bible"
	}
	},
	{
	"computerBooks": {
	"computerAuthor": "O'Brien, Tim",
	"computerTitle": "MSXML3: A Comprehensive Guide"
	}
	}
	],
	"opstatus": 0
	}

- The following table details the XPath for the OR operator and Output Result based on the sample response.

XPath for OR operator	Output Result

`fictionBooks: book [genre = 'Fantasy' or genre = 'Science Fiction']`	{
	"httpStatusCode": 200,
	"catalog": [
	{
	"fictionBooks": {
	"fictionTitle": "Midnight Rain",
	"fictionGenre": "Fantasy"
	}
	},
	{
	"fictionBooks": {
	"fictionTitle": "Maeve Ascendant",
	"fictionGenre": "Fantasy"
	}
	},
	{
	"fictionBooks": {
	"fictionTitle": "Oberon's Legacy",
	"fictionGenre": "Fantasy"
	}
	},
	{
	"fictionBooks": {
	"fictionTitle": "The Sundered Grail",
	"fictionGenre": "Fantasy"
	}
	},

- The following table details the XPaths for arithmetic operators and Output Result based on the sample response.

XPath	Output Result

• add: `count(//book[genre = 'Fantasy']) + count(//book[genre = 'Romance'])`
• romanceCount: `count(//book[genre = 'Romance'])`
• fantasyCount: `count(//book[genre = 'Fantasy'])`
• computerAuthorCount: `count(//book[genre = 'Computer' and author = "O'Brien, Tim"])`
• fictionCount: `count(//book[genre = 'Fantasy' or genre = 'Science Fiction'])`
• subtract: `count(//book[genre = 'Fantasy']) - count(//book[genre = 'Romance'])`
• divide: `count(//book[genre = 'Fantasy' or genre = 'Science Fiction']) div count(//book[genre = 'Romance'])`
• multiply: `count(//book[genre = 'Fantasy']) * count(//book[genre = 'Romance'])`
• modulus: `count(//book`
	{
	"add": 6,
	"romanceCount": 2,
	"fantasyCount": 4,
	"computerAuthorCount": 2,
	"fictionCount": 5,
	"subtract": 2,
	"divide": 2.5,
	"multiply": 8,
	"modulus": "1",
	"httpStatusCode": 200
	}

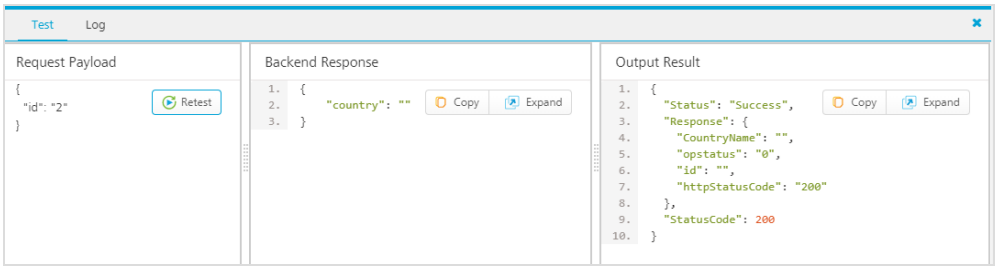
21.6.11 Test a Service Operation

To test a service operation, do the following:

1. From the **Select Environment** drop-down list, select an environment from the listed run-time environments configured for the Kony Fabric account.
2. Click **SAVE AND FETCH RESPONSE**. The operation gets saved and then the **Output Result** dialog shows the operation test results.

   The output result dialog displays two tabs **Test** and **Log**.

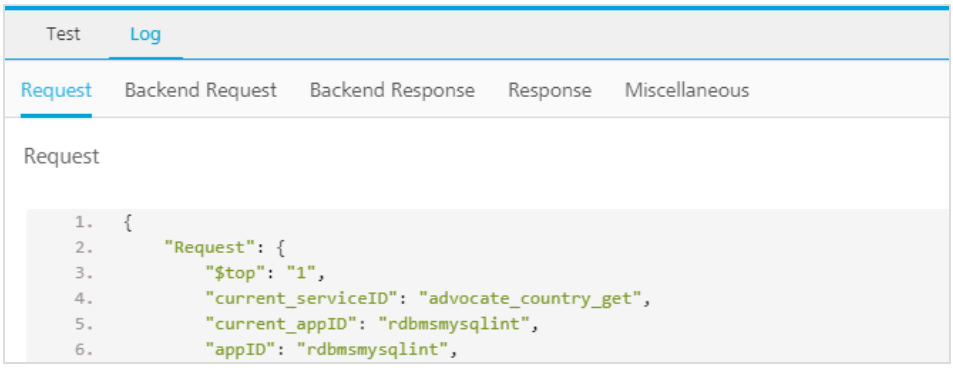
   a. The **Test** tab displays three sections Request Payload, Backend Response and Output Result. You can perform a retest by varying the request payload and clicking **Retest**.



   b. The **Log** tab details the flow of a complete service execution.

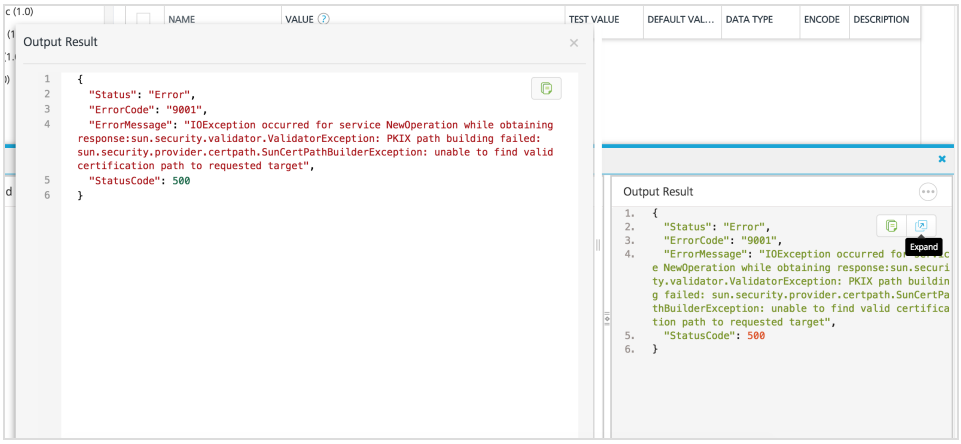
   The log tab displays the following details:

- **Request** displays the data related to request such as service ID, app ID and so on.
- **Pre-Processor Output** displays the pre-processor details added to the service.
- **Integration Service Input** displays the data sent to Integration service such as headers, payload and so on.
- **Integration Service Output** displays the response received from Integration service such as headers, response and so on.
- **PostProcessor Output** displays the postprocessor details added to the service.
- **Response** displays the final output of a service and the status showing success or failure of a test call.
- **Miscellaneous** displays the payload information before and after the pre-processor execution, performance data (time spent in executing different stages of pipeline), logs, and so on.
- **Backend Request** displays the data sent to the backend.
- **Backend Response** displays the response received from the backend.



c. You can perform following actions in this window:

- Click **Copy** to copy the code.
- Click **Expand** to pop over the section as a separate pop-up.



- **Choose number of sections** and content to view in the pop-over.
- **Increase/decrease** the height of the Output Result dialog.
- **Dock** the sections to different edges by clicking on the section partitions.

21.6.12 How to Use Custom Servlets, Filters, and Listeners

Kony Fabric lets you further customize your Kony Fabric apps by adding custom servlets, filters and listeners. This topic provides examples and describes how to add these to your Kony Fabric apps.

21.6.12.1 How to Add a Custom Servlet

You can add custom servlets with custom logic, and publish it by adding it to Kony Fabric application and it works as per the Java Servlet Specification.

For example, you can

- Create a servlet with the desired URL pattern where you can then write your custom code.
- Use custom servlets to integrate other third-party servlets.

The following sample code shows a simple custom servlet that redirects a request to the provided URL.

```
@IntegrationCustomServlet(urlPatterns = {"SampleCustomServlet"})
public class SampleCustomServlet extends HttpServlet {

  private static final Logger LOGGER = Logger.getLogger(SampleCustomServlet.class);

  private static final long serialVersionUID = 1L;
    /**
     * Default constructor.
     */
    public SampleCustomServlet() {
        // TODO Auto-generated constructor stub
    }
  /**
   * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
   */
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExceptio
    // TODO Auto-generated method stub'
    LOGGER.error("This Logs are from the CustomServlet : GET SampleCustomServlet");
    response.setContentType("text/html");
    PrintWriter pw=response.getWriter();

    response.sendRedirect("http://10.10.25.71/staticjson/redirecttothis.json");

    pw.close();
  }
  /**
   * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
   */
  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExcepti
    // TODO Auto-generated method stub
    LOGGER.error("This Logs are from the CustomServlet : POST SampleCustomServlet");
  }
}
```

21.6.12.2 How to Add a Custom Filter

You can add custom filters to add custom logic alongside the existing filters already provided by Kony Fabric.

The following steps describe how to add security checks to a request.

1. Create a filter class following the Java Servlet Specification, and then annotate it with the Kony **@IntegrationCustomFilter** annotation.
2. Write the logic in the **doFilter()** method, and then publish the custom JAR by adding it as a transitive dependency to your Kony Fabric App.

```
@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    throws IOException, ServletException {

  doBeforeProcessing(request, response);
  chain.doFilter(request, response);
  doAfterProcessing(request, response);

}
```

3. Configure **filterOrder** between the custom filters for your custom filter. Refer to the example below. Kony recommends that you start the **filterOrder** from the number 101 as 1 - 100 are reserved for Kony.
4. Configure a custom filter to a request pattern

```
@IntegrationCustomFilter(filterOrder=10,urlPatterns=MWConstants.ANY)
public class AddAdditionalResponseHeaderAttribute implements Filter {
  private static final Logger LOGGER = Logger.getLogger(AddAdditionalResponseHeaderAttribute.class
```

or

Configure a custom filter to one of the following Kony-provided servlets

- **MWServlet** - Used for integration and orchestration services.
- **AppServices** - Used for object services runtime request.

- **AppMetadataServices** - Used for object services metadata request.

```
@IntegrationCustomFilter(filterOrder=100,servletNames={"MWServlet","AppServices"})
public class MiddlewareMemCacheDCFilter extends DCFilter {}
```

5. Request pattern can be given to match for a particular service URL pattern.

21.6.12.3 How to Add a Custom Listener

You can also use custom listeners to add custom logic for application events, such as application start-up, or session create and destroy.

> ***Note:*** To create a custom session listener, use the **@IntegrationHttpSessionListener** annotation.

The following code sample shows a **SimpleSessionListener** class which implements an**HttpSessionListener** that is invoked every time the session is created or destroyed.

```
@IntegrationHttpSessionListener
public class SimpleSessionListener implements HttpSessionListener {

  private static final Logger LOGGER = Logger.getLogger(SimpleSessionListener.class);
  private static String sessionCheck = "null";

  @Override
  public void sessionCreated(HttpSessionEvent arg0) {
    // TODO Auto-generated method stub
    LOGGER.error("Session Created : From the Custom Listener");
    arg0.getSession().setAttribute("CustomListenerCreateAttribute",
        "CustomListenerCreateAttributeValue");
    arg0.getSession().setAttribute("isSessionActive", sessionCheck);
  }

  @Override
  public void sessionDestroyed(HttpSessionEvent arg0) {
    // TODO Auto-generated method stub
    LOGGER.error("Session Destroyed : From the Custom Listener");
    if("CustomListenerCreateAttributeValue".equals(arg0.getSession().getAttribute("CustomListenerCreateA
      sessionCheck = "true";
    }else {
      sessionCheck = "false";
    }
    arg0.getSession().setAttribute("isSessionActive", sessionCheck);
  }
}
```

21.6.12.4 How to Create a Custom Servlet Context Listener

Servlet Context Listener is used to listen to startup and shutdown event of context. Every method in listener interface takes Event object as input.

To create a custom servlet context listener, use **@IntegrationServletContextListener** annotation.

The following sample for Custom `ServletContextListener` is invoked on application life cycle events.

- `contextInitialized` - When the application boots up or during startup.
- `contextDestroyed` - When the application is stopped or during shutdown.

```
 package com.kony.customlistener;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.log4j.Logger;

import
com.konylabs.middleware.servlet.listeners.IntegrationServletContextLi
stener;

@IntegrationServletContextListener
public class MWServletContextListener implements
ServletContextListener {

  private static final Logger LOGGER = Logger.getLogger
(MWServletContextListener.class);

  @Override
  public void contextInitialized(ServletContextEvent sce) {
    LOGGER.error("Initializing custom resources");
    CustomResourceInitializer.initialize();
```

```
  }

  @Override
  public void contextDestroyed(ServletContextEvent sce) {
    LOGGER.error("Destroying custom resources");
    CustomResourceInitializer.destroy();
  }

}
```

Follow these steps to publish a custom ServletContextListener -

1. Create a servlet context listener with **@IntegrationServletContextListener** annotation.
2. Build a jar file for the class.
3. Upload the jar file as a dependency to a service in **Kony Fabric Console**.
4. **Publish** the app.

> ***Note:*** As the **ServletContextListener** is an application life cycle, the events will be invoked only once during startup and shutdown. After publish, the server must restart to invoke the `contextInitialized` event.

21.6.13 How to Save or Use a Version of a Service

You can save an integration service to a new version. Saving a new version of an integration service unlinks the current version of the service from the Kony Fabric application, and links the new version. A Kony Fabric app can be associated with only one version of an integration service.

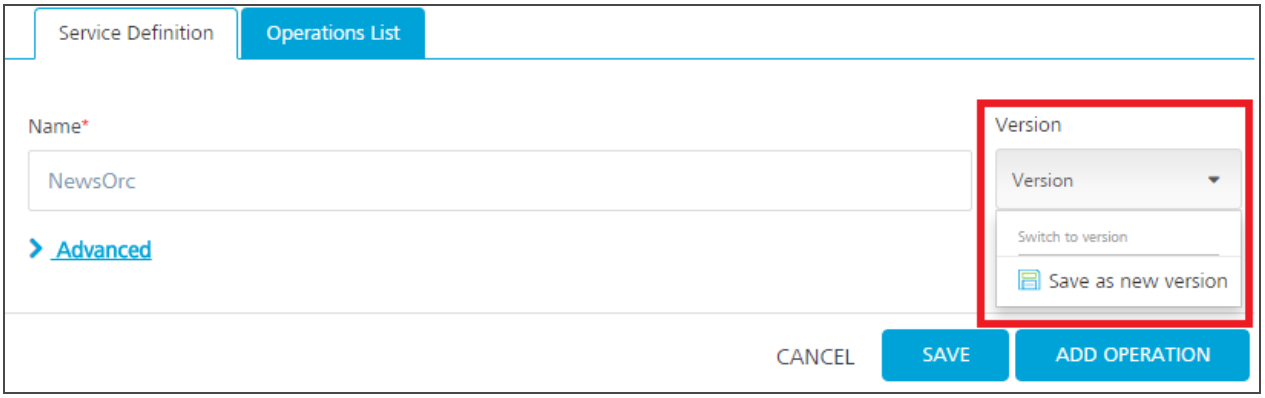
> ***Note:*** For more details on API Versioning Use Cases, refer to Kony Fabric API Versioning

To create a new version of an Integration service, follow these steps:

1. On the **Integration** tab, click the name of a service in the list of existing services. You can also select Edit from the Operations menu for a service.

   The **Service Definition** tab appears.

2. Click the **Version** menu.



3. Select **Save as new version**.

   The **Save as** dialog appears.

4. Type the version number of the new version of the service.

   You can use *major.minor* numbering for versions. Kony Fabric supports major versions from 1 through 999, and minor versions from 0 through 99. Kony Fabric supports version numbers from 1.0 to 999.99. Note that the dot '.' is a separator for version numbers and does not function as a decimal.

5. Add a description of the new version.

6. Click **OK**.

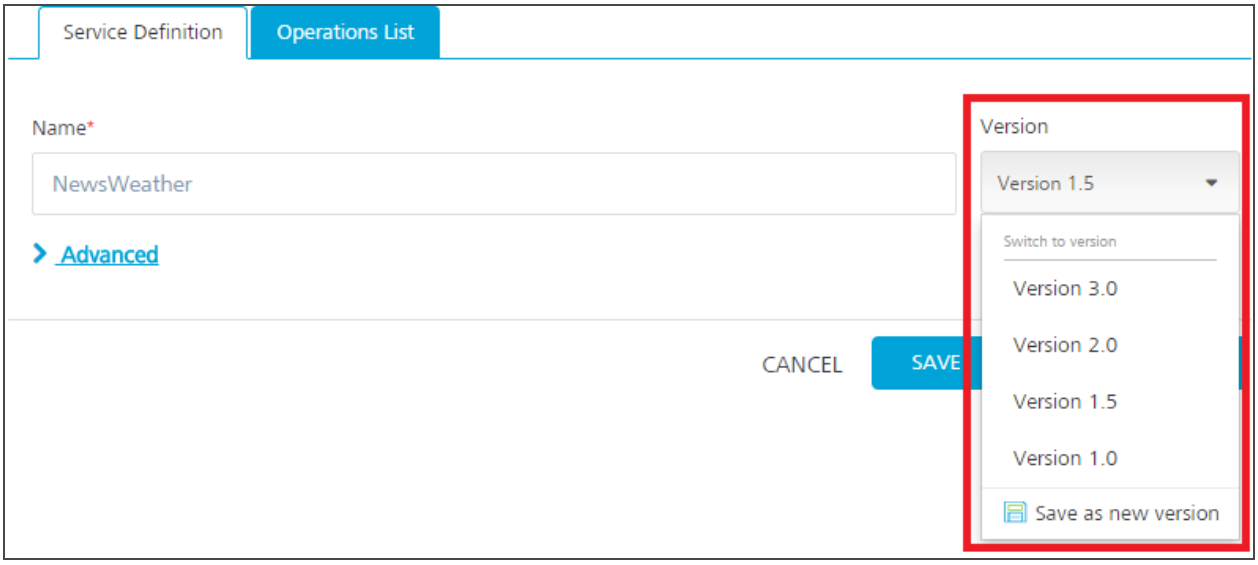
   Kony Fabric links the new version of the service to the Kony Fabric app, and unlinks the previous version.

To use a different version of an integration service, follow these steps:

1. On the **Integration** tab, click the name of a service in the list of existing services. You can also select Edit from the Operations menu for a service.

   The **Service Definition** tab appears.

2. Click the **Version** menu.



3. Select a version of the service.
4. The **Alert** dialog appears. The Alert warns you that you are unlinking the current associated version of the service from the application.
5. Click **OK**.

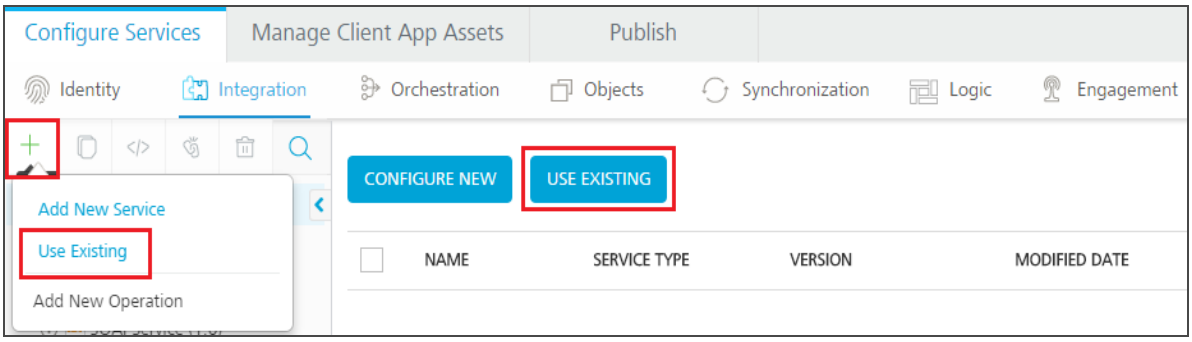
   Kony Fabric links the version that you selected to the Kony Fabric application.

21.6.14 How to Use an Existing Integration Service

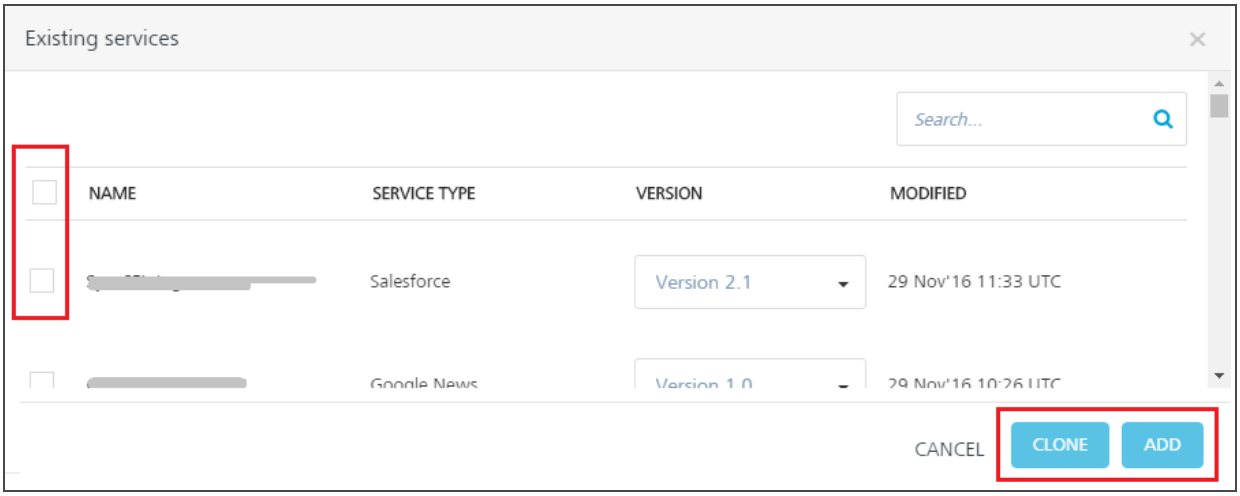
Kony Fabric allows you to use an existing integration service. You can add or clone more than one service at a time from existing services.

To use an existing integration service, follow these steps:

1. Go to the **Integration** tab. The page lists the existing services (if any).
2. In the **Integration**, click **USE EXISTING** from the tree in the left-pane or right pane.



The Existing services dialog appears with a list of existing services.



3. To use a different version of an integration service, click the **Version** menu of the required service and select a version of the service.
4. Select the check box for the desired services. If you want add or clone more than one service, select the required check boxes from the existing services.
5. Click **CLONE** or **ADD** button.
   - Click **ADD** to reuse (link) an existing service. If any changes made to this service, the changes will affect in all the apps using this service.

> ***Note:*** If a service is a part of a published app, you can rename that service only after the app is unpublished.

- Click **CLONE** to duplicate an existing service. If any changes made to this service will have no impact on the original service.

6. After a service is added or cloned successfully, click **CLOSE** to close the process dialog.

21.6.15 How to Clone, Unlink, or Delete Multiple Existing Integration Services

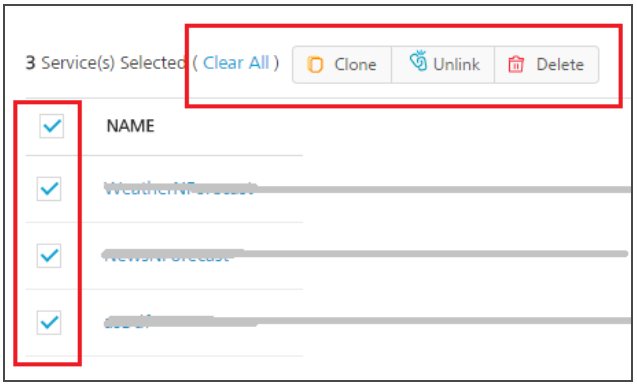
Kony Fabric allows you to clone, unlink, or delete one or more existing integration services from the **Integration** list page. You can add or clone one or more services at a time from existing services.

To clone, unlink, or delete multiple existing integration services, follow these steps:

1. Go to the **Integration** tab. The page lists the existing services (if any).

   By default, the check box is cleared for each service in the services list page.

2. Select one or more the check boxes for services. The quick access bar for the selected services appears with actions such as **Clear All**, **Clone**, **Unlink**, **Delete**.



- **Clear All**: Allows you to clear the one or more check boxes for the services.
- **Clone**: Allows you to duplicate an existing service. Changes made to a cloned service will not impact the original service.
- **Unlink**: Allows you remove the service from the **Integration** tab of an app. When a service is unlinked, it is disassociated from a particular app.

  > ***Note:*** If you want to use an unlinked service, select the service from the Use Existing Integration Service dialog.

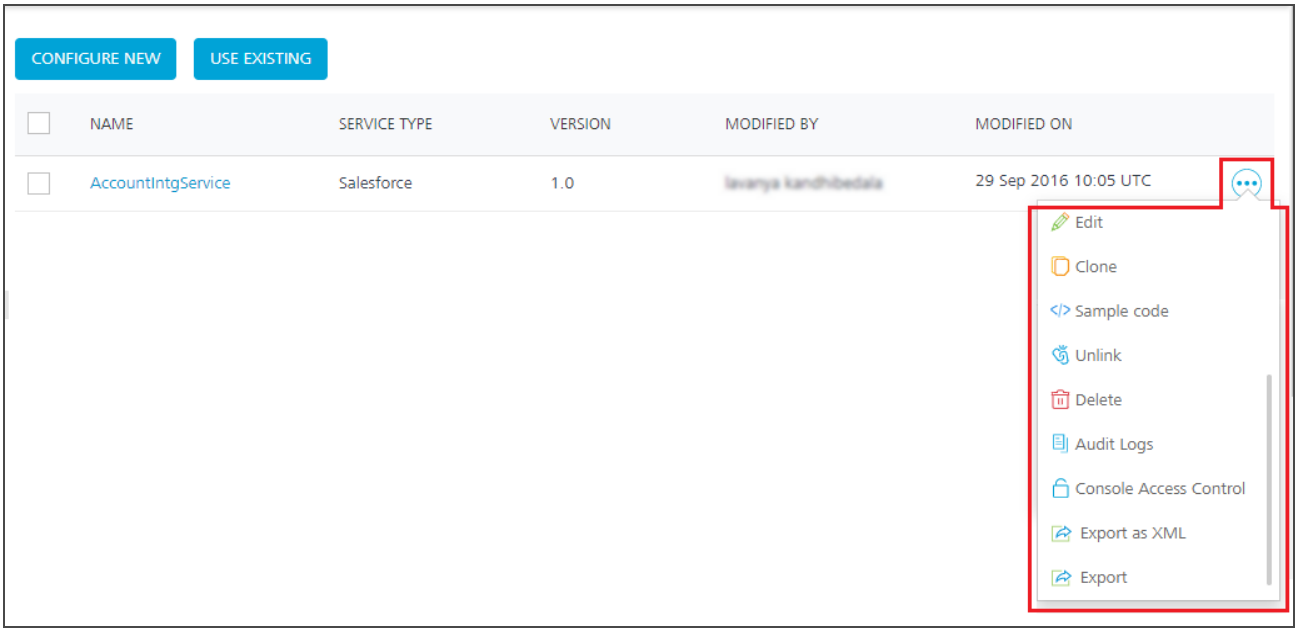
- **Delete**: Allows you to delete a service.

  > ***Note:*** If a service is a part of a published app, you can delete that service only after you unlink the service from all the published app.

3. Click the desired (**Clone**, **Unlink**, **Delete**) button.

21.6.16 Context Based Options

To perform various actions on an existing service, click the contextual menu of the required service.



The contextual menu contains the following options:

- **Edit**: Allows you to edit a service. After you edit a service, you have to republish all the apps that are using the service to apply the changes.

  > ***Note:*** To know more about publishing an app, refer to Publish an app.

  > ***Note:*** If a service is part of a published app, you can rename that service only after the app is unpublished.

- **Clone**: Allows you to duplicate an existing service. Changes made to a cloned service will not impact the original service.
- **Sample Code**: A dynamic code is generated based on the configuration of a service. You can use this code in your SDK.
- **Unlink**: Allows you remove the service from the **Integration** tab of an app. When a service is unlinked, it is disassociated from a particular app.

  > ***Note:*** If you want to use an unlinked service, select the service from the Use Existing Integration Service dialog.

- **Delete**: Allows you to delete a service.

  > ***Note:*** If a service is a part of a published app, you can delete that service only after you unlink the service from all the published app.

- **Audit Logs** helps you to capture all the user activities performed in a service. **Object Name**, **Object Type** and **Modified On** fields are prepopulated with the **Service Name**, **Services**, and **Last 7 Days** respectively.

  For more information on Audit Logs, refer to Audit Logs documentation.

- **Console Access Control**: Controls the access to the applications and services of apps.

- **Export as XML**: Exports an existing version of a service to an XML file.
- **Export**: Allows you to export the integration service into your local system. The exported file is an .xml file.

21.6.17 Test the Login for an OAuth 2.0 Identity Service

When you are defining an operation for an integration service, you can request that Kony Fabric fetch the response to the operation. If the integration service uses an OAuth 2.0 identity service for authorization, you must log in to the identity service before Kony Fabric can fetch the response from the back end.

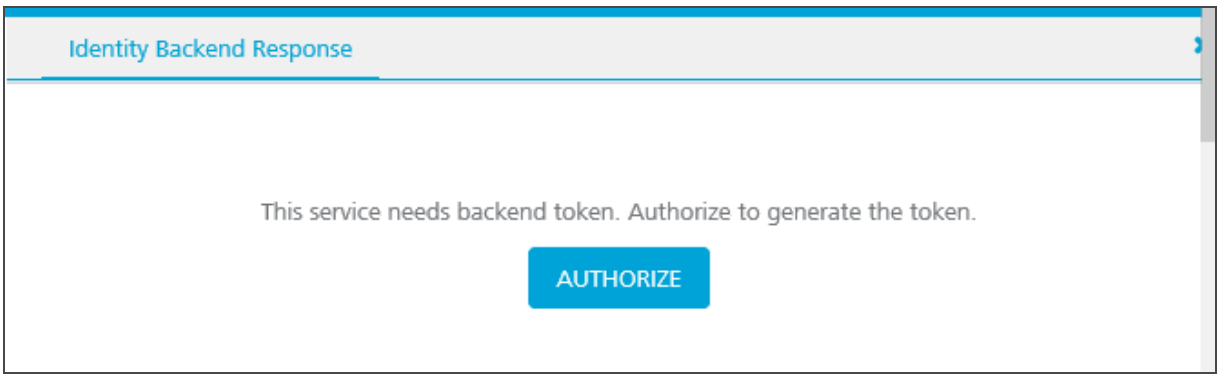
When you first request that Kony Fabric fetch the response, the service does not yet have the required token. You can then authorize Kony Fabric to generate the token. Kony Fabric then pops up the sign in for the OAuth 2.0 identity provider. After you successfully sign in to the identity service, Kony Fabric sends the request with the identity to the OAuth 2.0 back end. Kony Fabric gets the token and stores it in a cookie for Kony Fabric on the browser. Kony Fabric sends the token to the back end, and because the back end has the valid OAuth 2.0 token, the back end sends its response. The back end populates the identity backend response, the backend response, and the output result.

For the lifetime of the session, you can test an operation and you are not required to sign in again. In the identity backend response, you can view the token and clear the token if you want to test the login again.

To test the log in for an identity service, do the following:

1. Click **Fetch Response**.

   The **Identity Backend Response** tab appears. Kony Fabric identifies that the integration service is linked to an OAuth identity service. A message informs you that the service needs a backend token. If the message does not appear, the integration service has the backend token.



2. Click **Authorize**.

   The sign in screen for the identity service appears.

3. Enter your credentials and click **Sign In**.

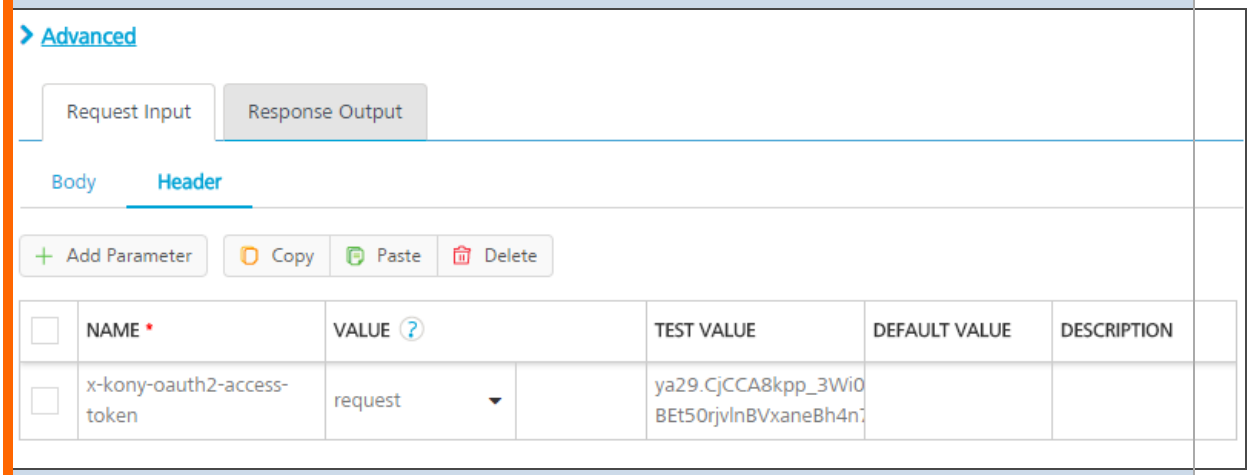
   The backend response is populated in the response pane.

4. To clear the cached token, in the Identity Backend Response tab, click **Clear Token**.

   In the Identity Backend Response tab, you can click **Copy** to copy the results of the response to the clipboard. Click **Expand** to open the Identity Backend Response in its own window.

> ***Important:*** If a custom integration service (for example, MongoDB or RAML) is linked to an OAuth2 identity service, while testing an operation of the integration service from Kony Fabric Console, you must pass the `x-kony-oauth2-access-token` as header and `access_token` as header value.
>
> Also, If a custom integration service (for example, MongoDB or RAML) is linked to an OAuth2 identity service, while testing an operation of the integration service from Admin Console, you must pass the `x-kony-oauth2-access-token` as header and `access_token` as header value.

For example:



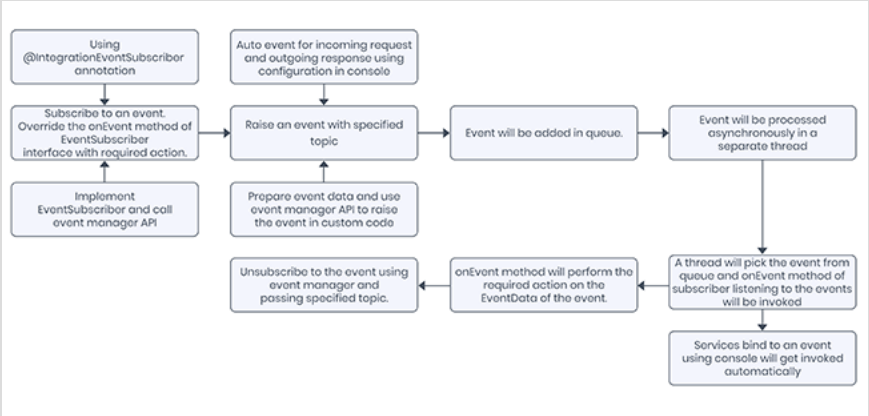
21.6.18 Server Events

Server-side events is a capability of the Kony Fabric runtime server that enables backend services or custom business logic to fire events and handle events. Server events are helpful when certain activities can be executed asynchronously such as processing a submitted order or invoking a slow backend API where the client app doesn't need to wait on the response.

21.6.18.1 Server Events Processing

To process an event, you have to create a Subscriber. You can do it multiple ways like by using an Annotation or through custom java API or by binding a service using Kony Fabric Console to act as a subscriber. When an event is raised, the Subscriber listening to it will get the event data. There are two ways to raise an event which are through Auto Event or through Java API.

This event will be added to the memory queue in Kony Fabric. A separate thread runs in Kony Fabric which will pick the event from the queue and notify the subscribers listening to the specified topic.



21.6.18.2 Terminology

Publisher - A service/custom code acts as a Publisher when it raises an event, when invoked. It sends the event data to the Event Bus for processing.

Subscriber - A service/custom code which is listening to a Topic is called a Subscriber. When an event is raised with that Topic, the Subscriber gets the related event data from Event Bus and processes the data.

Topic - Topics are classes or named logical channels to which Publisher can fire events. The Subscribers listening to a Topic will receive the related event data whenever an event is raised with the subscribed Topic.

Event - These are messages that contain event data. A Publisher raises an event and a Subscriber consumes the event data of an event.

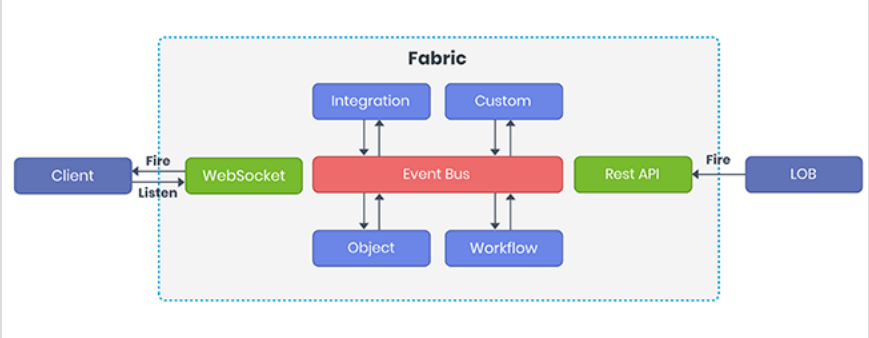
Event Bus - It acts as a message broker which stores and routes the event data from Publishers to Subscribers.

21.6.18.3 Create a Server-Side Event

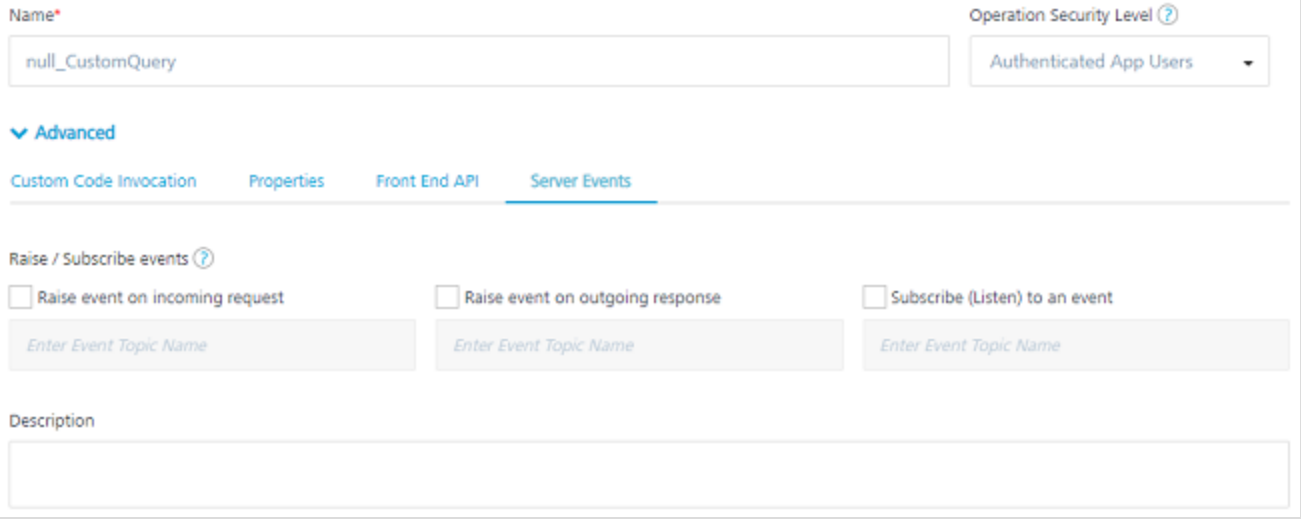
Events can be created (i.e. “raised” or “fired”) as part of a service invocation or from custom business logic. Kony Fabric support auto creation and fire of events for incoming request and outgoing response of a service. User can also bind a service to one or more events with specified topics. Custom business logic can subscribe/unsubscribe/raise an event using exposed APIs.

Kony Fabric exposes a WebSocket end point to subscribe/unsubscribe/raise an event from an external client. Kony Fabric also exposed a HTTP end point to fire an event from any external client. . Once the event is created, the service or business logic will finish executing it. Another service or custom business logic should be configured to handle that event and perform the desired action and this work will be executed in a different process on the server asynchronously.

> ***Note:*** HTTP end point doesn't support subscribe and unsubscribe.



Create an Event from any Service



Create event on incoming request: To create the server-side event on the inbound request, in the Advanced settings tab of an operation, click the **Server Events** tab and then select the **Raise event on incoming request** checkbox. Provide the name of the event that describes the data or purpose of the event in the related **Enter Event Topic Name** box. The event will include the payload of the incoming request and default parameters like service id, operation id, object id, service version, headers, query parameters and more. You can provide semi-colon separated multiple event topic name.

Create event on outgoing response: To create the server-side event on the outbound response, in the Advanced settings tab of an operation, click the **Server Events** tab and then select the **Create event on outgoing response** checkbox. Provide the name of the event that describes the data or purpose of the event in the **Enter Event Topic Name** box. The event will include the payload of the outgoing response payload and default parameters like service id, operation id, object id, service version, headers, query parameters and more. You can provide semi-colon separated multiple event topic name.

Other services can listen to this event with the help of the given Topic name. Whenever this service is invoked, an Event is raised and subscribers to this Topic will get the relevant event data. You can also provide the Topic name in Parent topic and Child topic name format. When a service subscribes to the parent topic, the event data of all the child topics is sent to the subscriber. If you subscribe to any child topic, the event data related only to the subscribed child topic is sent to the subscriber. Use “/” as a separator for parent and child topic name.

For example, Parent service name can be “transaction” and its child can be “transaction/credit” and “transaction/debit”.

Create Event from Business Logic

APIs

- **EventData**: This class is used to create events with details. The subscriber will get the event details if the conditions in the event are met. Event details can contain id, topic, type (type of event data in an event like transaction, audit, and more), data, additional metadata, producer, timestamp, app id and user profile from the identity of the event. Following are the constructors supported to create EventData:

- **EventData(String topic, Object data)**: It takes only topic and data to create EventData. Identity related attributes like app id, user profile will not be generated in this case.
- **EventData(String topic, Object data, ServicesManager servicesManager)**: It takes topic, data and services manager to create event data. Services Manager will be used to populate identity related attributes like app id, user profile.

- **EventManager**: This class is used to subscribe/unsubscribe to an event of a specified topic. You can access Event Manager using Services Manager API in their custom code.

```
request.getServicesManager().getEventManager()
```

This class contains the following APIs:

- **subscribe**: Subscribes the given event subscriber to all events with the specified topic.

```
request.getServicesManager().getEventManager().subscribe
(<topic>, <subscriber>);
```

- **unsubscribe**: Unsubscribes the given event subscriber to all events with the specified topic.

```
request.getServicesManager().getEventManager().unsubscribe
(<topic>, <subscriber>);
```

- **EventNotifier**: This class is used to notify subscribers and check the status of events. You can access Event Notifier using Services Manager API in their custom code.

```
request.getServicesManager().getEventNotifier();
```

This class contains the following APIs:

- **notify (with event data)**: Notify all subscribers of the given event based on the topic.

```
request.getServicesManager().getEventNotifier().notify
(<event_data>);
```

- **notify (with topic and data)**: Creates event data internally and notifies all subscribers of the given event based on the topic.

```
request.getServicesManager().getEventNotifier().notify
(<topic>, <data>);
```

21.6.18.4 Handling Server Side Events

Once an event is created, a service or custom business logic is configured to handle that event. The execution of the event handler occurs asynchronously in a separate thread. If the event is handled by a service, the event payload is passed as the request to the service similar to a client invoking the service as a REST API. If the event is handled by custom code, the event payload is passed into the custom logic as part of the event data.

Handle Events from Service

Subscribe (Listen) to an event: To bind the service to an event of specified topic, in the Advanced settings tab of an operation, click the **Server Events** tab and then select the **Subscribe (Listen) to an event** checkbox. The service will be invoked whenever the subscribed event is triggered with the specified topic. Type the name of the event in the related **Enter Event Topic Name** box. You can provide semi-colon separated multiple event topic name.

Handle Event from Custom Logic:

APIs

- **EventSubscriber**: This class provides mechanism for receiving notification from notifier and perform the required actions. User can implement this class in their custom code. This class contains the following API:
  - **onEvent**: Receives notification when any event of the subscribed topic is triggered. Make sure that this API is thread-safe as multiple threads can call this API simultaneously.

> ***Note:*** Refer to the Fabric Queue Service java document for more information on Fabric Queue Service APIs.

21.6.18.5 Create Subscriber from Business Logic

User can create subscriber in their custom code by following ways:

Implementing EventSubscriber Interface:

Implement **EventSubscriber** interface and provide specific implementation of **onEvent** method to create custom subscriber.

```
public class CustomEventSubscriber implements EventSubscriber {

    @Override
    public void onEvent(EventData eventData) {
        // user specific implementation
    }
}
```

Using Annotation

Use **@IntegrationEventSubscriber** to create custom subscriber. While creating subscriber using annotation, user can specify the topic name that he wants to subscribe, and the framework will subscribe the user to the specified topic.

```
@IntegrationEventSubscriber(topics = {"app/test/events",
"app/test/events1"}
public class TestEventSubscriber implements EventSubscriber {

    @Override
    public void onEvent(EventData eventData) {
        // user specific implementation
    }
}
```

21.6.18.6 Use Case

An online shopping website can use server events feature to send the order status emails automatically to its customers who have placed orders on their website. The developer can write the custom code to trigger events when an order is either successful or failed. The concerned customer will get an email depending upon the status of the order.

Custom Code to trigger an Event

```
import com.konylabs.middleware.api.events.EventData;
import com.konylabs.middleware.common.DataPostProcessor2;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.controller.DataControllerResponse;
import com.konylabs.middleware.dataobject.Result;

public class PostProcessorToTriggerEvents implements
DataPostProcessor2 {

  @Override
  public Object execute(Result result, DataControllerRequest request,
      DataControllerResponse response) throws Exception {
    EventData eventData;
    if (response.getStatusCode() == 200) {
      eventData = new EventData("apps/order/success",
response.getResponse());
      eventData.addAdditionalMetadata("orderId",
result.getParamValueByName("orderId"));
    } else {
      eventData = new EventData("apps/order/fail",
response.getResponse());
      eventData.addAdditionalMetadata("trackingId",
result.getParamValueByName("trackingId"));
    }
```

```
    eventData.addAdditionalMetadata("userId",
result.getParamValueByName("userId"));
    eventData.addAdditionalMetadata("product",
result.getParamValueByName("product"));
    request.getServicesManager().getEventNotifier().notify(eventData);
    return result;
  }
}
```

Event on successful order

```
import java.util.Map;

import com.konylabs.middleware.api.events.EventData;
import com.konylabs.middleware.api.events.EventSubscriber;
import com.konylabs.middleware.api.events.IntegrationEventSubscriber;

@IntegrationEventSubscriber(topics = {"apps/order/success"})
public class OrderSuccessEvent implements EventSubscriber {
  private static final String SUCCESS_MESSAGE = "Dear %s, Your order
for %s is successful. "
      + "Your order id is%s. Order details:%s.";

  @Override
  public void onEvent(EventData eventData) {
    Map<String, Object> additionalMetadata =
eventData.getAdditionalMetadata();
    Object userId = additionalMetadata.get("userId");
    Object product = additionalMetadata.get("product");
    Object orderId = additionalMetadata.get("orderId");
    String message = String.format(SUCCESS_MESSAGE, userId, product,
orderId, eventData.getData());
    sendSuccessMail(String.valueOf(userId), message);
  }
```

```
  private static void sendSuccessMail(String userId, String message) {
  }
}
```

Event on failed order

```
import java.util.Map;

import com.konylabs.middleware.api.events.EventData;
import com.konylabs.middleware.api.events.EventSubscriber;
import com.konylabs.middleware.api.events.IntegrationEventSubscriber;

@IntegrationEventSubscriber(topics = {"apps/order/fail"})
public class OrderFailedEvent implements EventSubscriber {
  private static final String FAILURE_MESSAGE = "Dear %s, Your order
for %s is failed. "
      + "Your tracking id is %s. Details: %s.";

  @Override
  public void onEvent(EventData eventData) {
    Map<String, Object> additionalMetadata =
eventData.getAdditionalMetadata();
    Object userId = additionalMetadata.get("userId");
    Object product = additionalMetadata.get("product");
    Object trackingId = additionalMetadata.get("trackingId");
    String message = String.format(FAILURE_MESSAGE, userId, product,
trackingId,
        eventData.getData());

    sendFailureMail(String.valueOf(userId), message);
  }

  private static void sendFailureMail(String userId, String message) {
```

```
   }
}
```

21.6.18.7 Raise an Event from HTTP end point

In Kony Fabric, you can also raise an event from HTTP end point from external client. You have to invoke a POST method on /ServerEvents endpoint (sample URL: <kony_app_services_ url>/services/ServerEvents) with X-Kony-Authorization token in header, JSON array body with events as key, topic name, and data for each event.

Sample Request:

Method: POST

Header: X-Kony-Authorization : <Authorization_token>

Body: JSON array with events as key.

```
{
    "events": [
        {
            "topic": "transaction/deposit",
            "data": "988",
            "additionalMetadata": {
                "amount": "5678",
                "user": "kony"
            }
        }
    ]
}
```

21.6.18.8 WebSocket endpoint for events (/ServerEvents/Stream)

If you want to support a bi-directional communication between the client and server, you can use WebSocket to subscribe to an event, unsubscribe from an event, raise an event, and listen to an event from client. WebSocket endpoint is protected with **X-Kony-Authorizatio**n token. By default, WebSocket is disabled. Set the **KONY_SERVER_CLIENT_EVENTS_ENABLED** to **true** in **server_**

configuration table of admin DB to enable WebSocket.

Sample end point URL: <kony_fabric_app_services_ul>/services/ServerEvents/Stream

Protocol: ws

Header: X-Kony-Authorization : <Authorization_token>

Body:

- **Subscribe to an event**: JSON array with topic of events to subscribe and subscribe as key.

```
{
    "subscribe": [
        "transaction/deposit",
        "transaction/withdraw"
    ]
}
```

- **Raise an event**: JSON array with events as key and topic name and data in body.

```
{
    "events": [
        {
            "topic": "transaction/deposit",
            "data": "988",
            "additionalMetadata": {
                "amount": "5678",
                "user": "kony"
            }
        },
        {
            "topic": "transaction/withdraw",
            "data": "1500",
            "additionalMetadata": {
```

```
                "amount": "12345",
                "user": "fabric"
            }
        }
    ]
}
```

- **Unsubscribe to an event**: JSON array with topic of events to subscribe and **unsubscribe** as key.

```
{
    "unsubscribe": [
        "transaction/deposit",
        "transaction/withdraw"
    ]
}
```

> ***Note:*** You can also subscribe, unsubscribe and raise an event with JSON array as body with required keys and data.

21.7 Manage Existing Integration Services

You can perform the following actions to manage the existing Integration Services.

- Save or Use a Version of a Service
- Use an Existing Integration Service
- Clone, Unlink, or Delete Multiple Existing Integration Services
- Context based Options

22. Orchestration

22.1 Overview

Orchestration Services leverage the concept of combining multiple integration services, object services or orchestration services into a single service to simplify business logic in client apps and reduce the number of service invocations.

22.2 Use Case

An HR executive wants to fill a vacancy in the company available for a developer with 3-6 years of experience. The HR uses an HR tool to search for the suitable candidates. The tool sends the initial request to an Orchestration service which then invokes other services that connect to various data sources, for example, Naukri, LinkedIn, Monster, Indeed and get the list of suitable candidates.

Finally, the orchestration service gets the list of eligible candidates suitable for the vacancy and the data is displayed in the HR tool.

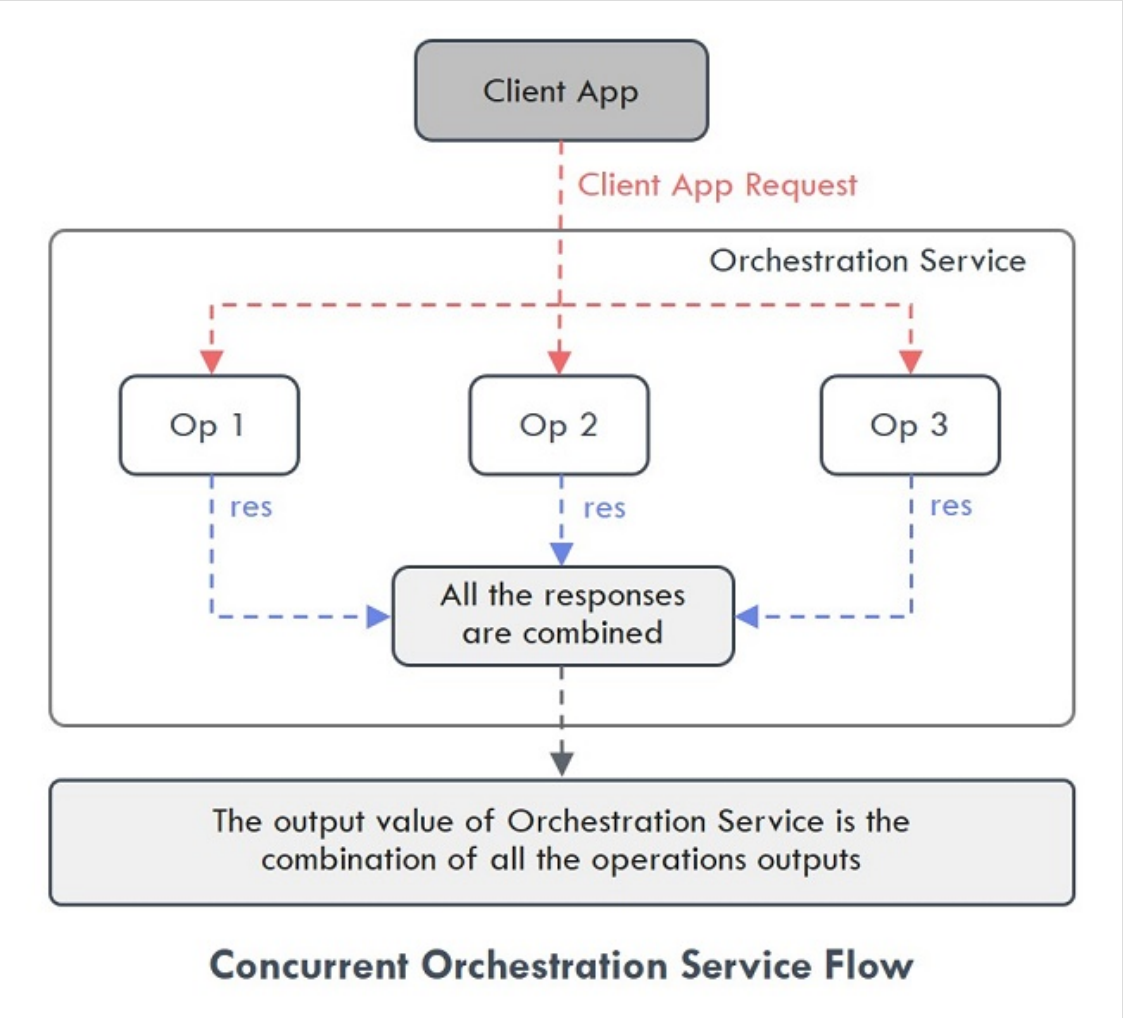
22.3 Orchestration Service Operations

Operations are the functional entities that are invoked from the client-side core functions that define the capabilities of any service. Operations in an orchestration service can be defined to invoke operations from multiple services. You can create Orchestration service operations from the existing Integration service operations, Object service operations, and other Orchestration service operations; and assign an execution strategy to these operations so that they are executed in the defined order.

There are two major execution types for Orchestration service Operations:

- **Composite** - The Composite execution type allows you to combine multiple operations and execute the operations added to the Orchestration service. There are two execution modes for this execution type:
    - **Composite Concurrent** - This executes the operations added in a service simultaneously, and there is no order in the way the operations are executed. For

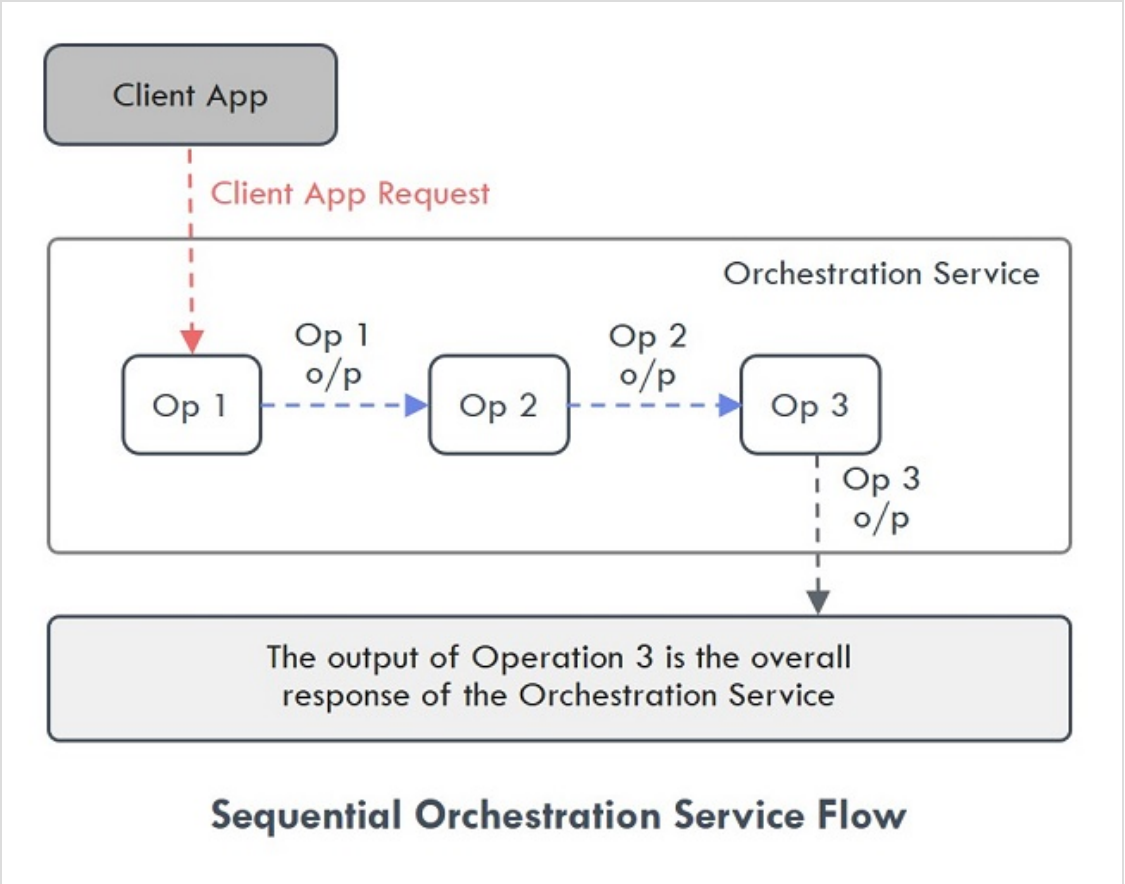
example, in the provided use case, the service calls made to different data sources are executed simultaneously.



Concurrent Orchestration Service Flow

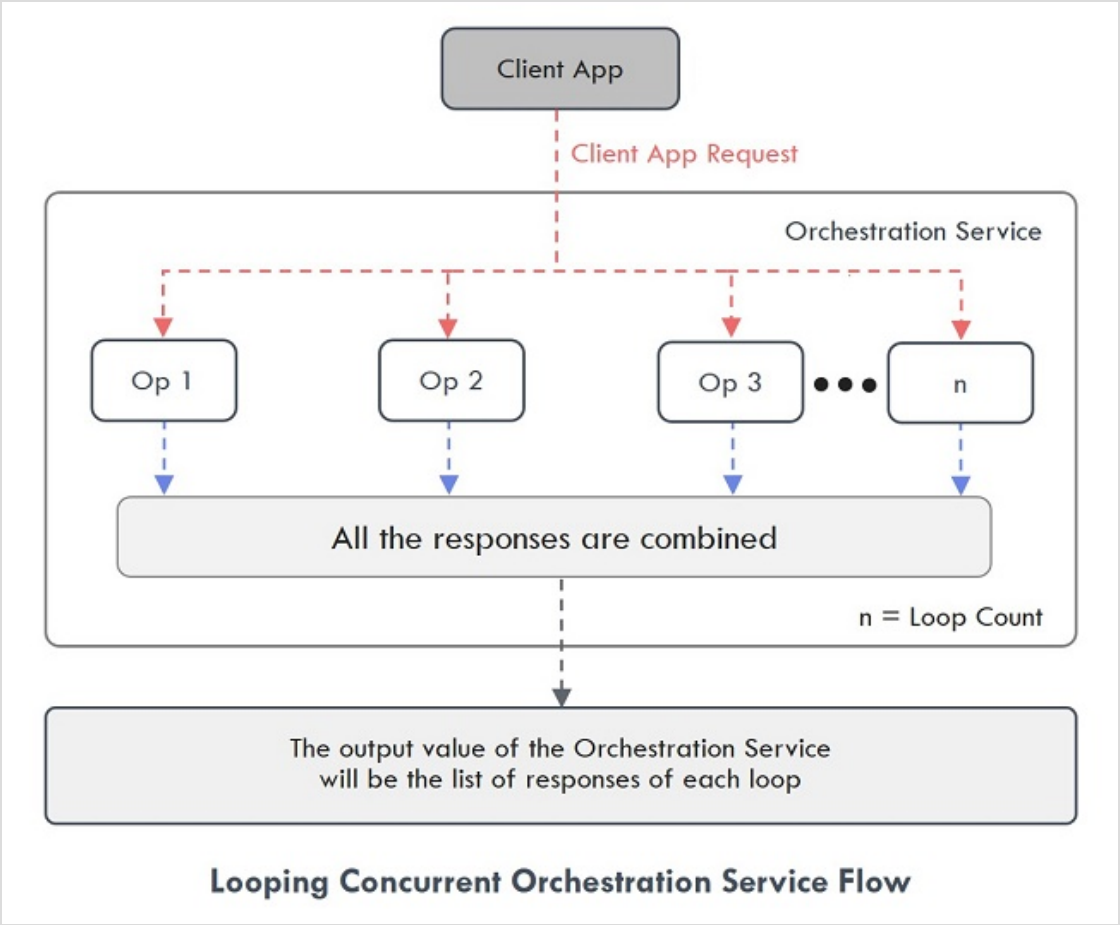
- **Composite Sequential** - This mode executes the operations added in a service in a specified order, and you can also use the output response of one service as input request for the next service.
  For example, a weather app with Orchestration service gets the Geolocation of the device first, then gets the Pin code based on the input from the Geolocation, and then calls a

weather service to get the temperature based on Pin code.



Sequential Orchestration Service Flow

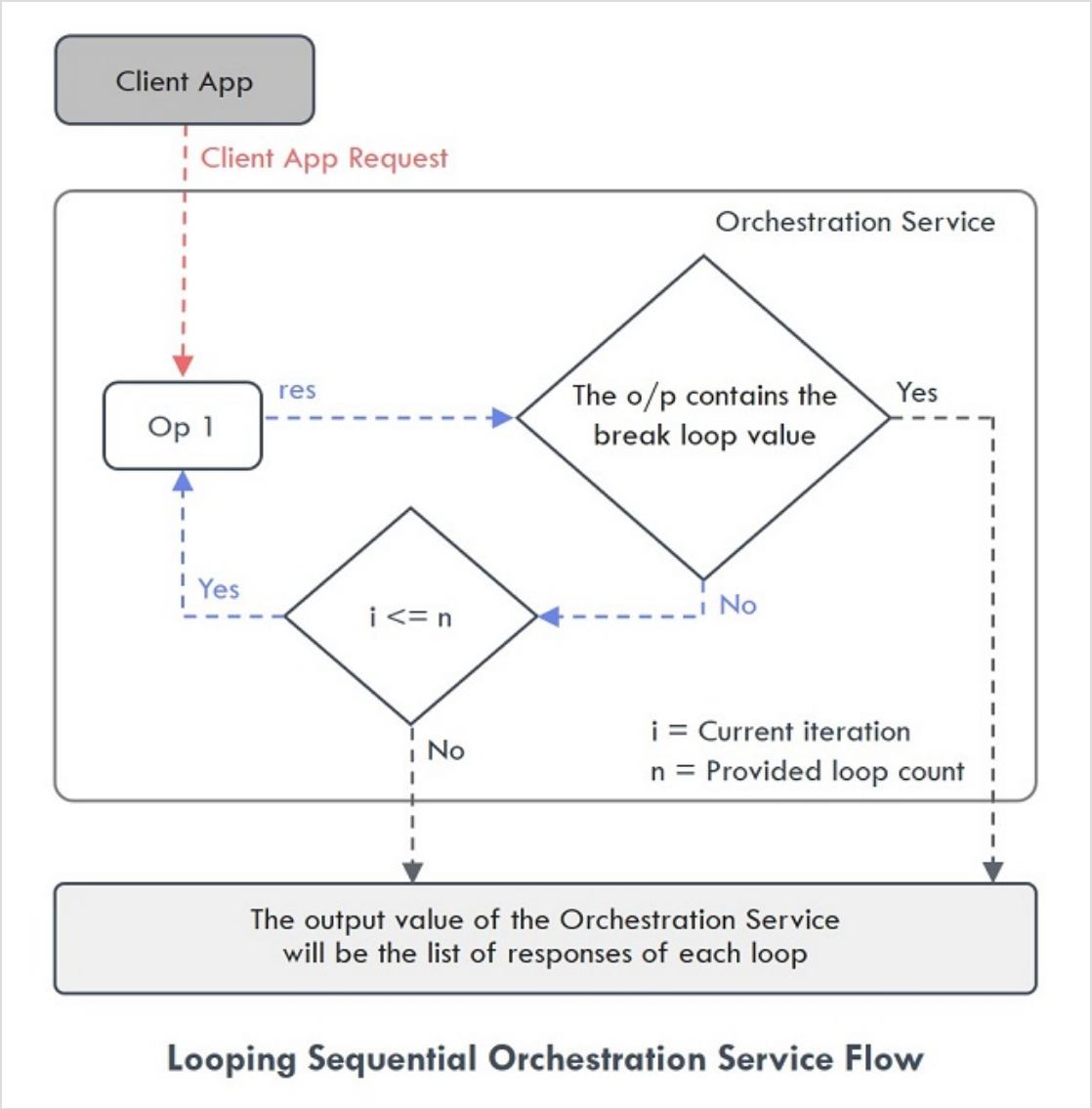
- **Looping** - You can add only one operation in the orchestration service if you want to select looping execution type. It allows you to execute the operation repeatedly in a loop using a delimited set of inputs or until a specified condition is met. You have to define the following parameters if you want to execute the operation in a loop:
    - **Looping Concurrent** - When you select this mode, the operation is executed simultaneously “n” number of times based on the loop count value. The output value of the Orchestration Service will be the list of responses of each loop.



Looping Concurrent Orchestration Service Flow

- **Looping Sequential** - When you select this mode, the operation is executed sequentially in a loop till the break loop parameter or loop count is met. The output value of the

Orchestration Service will be the list of responses of each loop.



Looping Sequential Orchestration Service Flow

> *Important:* You cannot add any service to an Orchestration Service in which the **Enable Pass-through** feature is enabled.

22.4 Orchestration Service Tab

To go to the Orchestration tab from the Kony Fabric Console dashboard, click **Add New** or select any existing app > **Configure Services** > **Orchestration**.

If you have not configured any Orchestration service before, the Kony Fabric displays a message that there are no Orchestration services configured on the Orchestration tab landing page.

You can do the following from the Orchestration service landing page:

- Create a new Orchestration service
- Use existing Orchestration services in the account to create a new Orchestration service
- Manage the available Orchestration services

22.5 Create a New Orchestration Service

> ***Note:*** If you want to create an Orchestration service without associating it with any Kony Fabric app, from the side menu, click API Management > APIs > Orchestration > Configure New. For more information on API Management, refer API Management.

To configure a new Orchestration service, you must first create a service definition and add operations to that service definition.

22.5.1 Create a Service Definition

To create a new Service Definition, in the Orchestration page click **Configure New** or in the left pane click the "+" icon and select **Add New Service**. The Kony Fabric displays the **Service Definition** screen. You can do the following on the **Service Definition** screen:

- In the **Name** box, type a unique name for the new Orchestration service. The name of the service must not match with any of the existing service names.

  > ***Note:*** The service name should not contain special characters, must begin with a letter, and should be between 4 and 30 characters. The service name can include underscore (_) and dash (-) characters.

- The version of the Orchestration service is mentioned by default in the **Version** list. To know more about service versions, click here.

- In the **Description** box, type the required definition for the new service. This field is optional.

In Kony Fabric, you can upload a custom code library in the form of a .jar file. These uploaded files can be used across all the services.

If you want to associate this new service with a custom code library or apply a throttling limit, expand the **Advanced** section. This section contains two tabs, which are **Custom Code** and **Throttling**. By default, the **Custom Code** tab is selected.

Custom Code

From the **Custom Code** tab, you can select the .jar file you want to associate with your new service. You can do the following to select the .jar file:

> *Important:* If you have access to multiple accounts in Kony Fabric Console, the uploaded .jar files are available only in that account where they were uploaded. The .jar files uploaded in one account will not be available in another account. These files will be accessible to anyone who has access to that accounts.

- If you do not have the required .jar file available in the account, click **Upload New** and select the required .jar file from your local machine.
- If you have the required .jar file available in the accounts, click **Select existing JAR** list and select the required file.

Throttling

From the **Throttling** tab, you can provide the number of times you want to invoke an API per minute. If the API exceeds its throttling limit, it will not return the service responses. You can do the following to provide the throttling limit:

- In the **Total Rate Limit** box, type a required value. This value limits the total number of requests processed by the API.
- In the **Rate Limit per IP** box, type a required value. The value limits the number of requests

made from an IP address. If the device exceeds the limit set, the API returns an appropriate error message.

> ***Note:*** If both the values are given, the throttling limit which is met first will take precedence.

Once all the required parameters are configured in the **Service Definition** page, click **Save** if you want to save the new service and add operations to it later. Click **Save and Add Operation** if you want to add operations to that service immediately.

The new service is added to the All Services list and to the Orchestration tab landing page.

22.5.2 Create an Operation

If you have clicked **Save** on the **Service Definition** page, Kony Fabric displays an **Operations List** tab when you open the service next time. Select the **Operations List** tab and click **Add Operation** or in the left pane, click the "+" icon and select **Add New Operation**. Kony Fabric displays the **NewOperation** tab.

If you have clicked **Save and Add Operation** in the **Service Definition** page, it directly opens the **NewOperation** tab.

You can do the following to add an operation to the service:

- In the **Name** box, type a unique name for the new operation. The name of the operation must not match with any of the existing operation names.
- From the **Operation Execution Type** list, select the required execution type. The available execution types are as follows:
    - Composite Concurrent
    - Composite Sequential

- Looping Concurrent
- Looping Concurrent

- If you select a looping operation, the following fields appear:
  - **Loop Count**: It denotes the number of times the service included in the looping service must get executed.
  - **Loop Separator**: Specifies the character that separates the input parameters of a service. For example: When the input parameter for the operations you add in the Orchestration Service are different, then the character provided here will act as a separator for the input parameters like ",".
  - **Break Loop Parameter Name**: Specifies if there are any breakout parameters for the execution of the looping service. The breakout parameter is a part of the result returned from the service call within a loop.
    For example: If you want a particular Key Value pair in the response as the break loop parameter, then mention the name of the Key here.
  - **Break Loop Parameter Value**: Specifies the value for the breakout parameter of the looping service. If the value of the breakout parameter returned from the service matches with the given value, the operation exits the loop.
    For example: If you want a particular Key Value pair in the response as the break loop parameter, then mention the Value here.
- From the **Operation Security Level** list, select the required authentication type. It defines the way a client must authenticate to access this operation. The available execution types are as follows:
  - **Authenticated App User** – It restricts the access to clients who have successfully authenticated using an Identity Service associated with the app.
  - **Anonymous App User** – It allows the access from trusted clients that have the required App Key and App Secret. Authentication through an Identity Service is not required.

- **Public** - It allows any client to invoke this operation without any authentication. This setting does not provide any security to invoke this operation and you should avoid this authentication type if possible.
- **Private** - It blocks the access to this operation from any external client. It allows invocation either from an Orchestration/Object Service, or from the custom code in the same run-time environment.

- From the **Front End HTTP Method** list, select the required request method you want to use to invoke the service operation from a HTTP Client. The available request methods are as follows:
  - GET
  - POST
  - PUT
  - DELETE

  For example, Visualizer and Other SDKs use POST method for communicating with Kony Fabric.
- In the **Description** box, type the required definition for the new service which you are creating now. This field is optional.
- The **Operation Mapping > Available Services** section lists all the services from Integration and Object services. These services contain all the operations associated with them, and you can map the operations you want to use for the new Orchestration service. Do the following to map the operations:
  - Expand the required services and select the required operations of that service.
  - Drag and drop them into the right pane in the **Operation Mapping** section.
- In the **Enable pass-through** section, select the required parameter type. This field is optional. If you select any parameter type, the related parameter variables are shown as is from the client server. The types of parameter available are: Input, Header, and Output.

For example, if you have used multiple Integration service operations to create an Orchestration service, and selected the **Input** check box in **Enable pass-through** section. The **Request Input** parameter variables configured for those operations will be ignored, and they will be shown as is from the client server.

> ***Note:*** This filter will be applicable to all the operations which you have dropped in the right pane of Operation Mapping.

- If you want to check the service functionality, from the **Environments** list, select the environment to which you want to publish the app, and then click **Save and Fetch Response**. The responses from the client server are displayed. For more information, refer to Test an Orchestration Service Operation.
- Click **Save Operations**. The new operation is added in the **Operations** tab of that Orchestration service.

22.6 Use Existing Service to Create a New Orchestration Service

This feature helps you to use an existing Orchestration service in a Kony Fabric account and use it to configure a new Orchestration service. You can either clone or add existing services and make changes to them accordingly.

To create a new Orchestration service from the available services, perform the following steps:

- In the **Orchestration** service tab, click **Use Existing** or in the left pane click the "+" icon and select **Use Existing**. The **Existing services** screen is displayed.
- Select the required services from the **Existing services** screen and click **Clone** or **Add**. The Clone Service or Add Service status screen appears.
    - **Clone**: It creates a duplicate of the selected service. The changes made to the duplicate service will not affect the original service.

- **Add**: It adds the selected service to the new Kony Fabric app. The changes made to the service will affect all the apps using the service.
  If the service is part of any published app, you must unpublish the service to rename it.

> ***Note:*** If the list is long, you can search for the required service with **Search** option.

- After the **Clone** or **Add** process is complete, the service is added to the Orchestration services list.
- Click the newly added service or open the Contextual menu and click **Edit** to configure the details of the service. For more information on configuring the details, refer to Create a Service Definition and Create an Operation.

22.7 Manage Services

All the services related to a service type are listed on the landing page of the respective service type. You can manage the details of a service from the Contextual menu available adjacent to each service. The following options are available in the Contextual Menu:

- **Edit** - Click to edit the details of a selected service. After you edit a service, republish all the apps that use this service to apply the changes.
- **Clone** - Duplicates an existing service. Clone a service to create a different version of the same service. Changes made to a cloned service will not affect the original service. The name of a cloned service indicates that it is a copy of an existing service.
- **Sample Code** - Generates dynamic code for each SDK type based on the configuration of a service. You can use the code in your mobile app. For example, generate the sample code for an orchestration service from Kony Fabric. Then use that code in the mobile app to invoke the orchestration service instance.
- **Unlink** - Removes a service from the Orchestration tab of an app. Unlink a service when you don't want to associate the service with the Kony Fabric app.

- **Delete** - Deletes a selected service from Kony Fabric Console. You cannot delete a service if the service is in use. A service in use is a service that is referenced by a Kony Fabric app or another service or a Sync scope.

  > ***Note:*** When you delete a service that has multiple versions, only the active version is deleted.

- **Console Access Control** - You can manage the users who can access this service from here. To know more, refer to Console Access Control.
- **Export as XML** - Exports the current version of a service in the form of an XML file.
- **Export** - Exports the service details in the form of a zip file. You can import this zip file to another Kony Fabric app and use it. For more information, refer to Export and Import an Application.

22.8 Service Versions

You can use Service Versioning if you want to save the changes made to a service as another version of the same service. When you create a new version for a service, the Kony Fabric app associated with the service will use the latest version of the service automatically.

> ***Note:*** A Kony Fabric app can be associated with only one version of an orchestration service.

You can do the following in the Service Definition (link) tab to create a new version for a service:

1. In the **Version** list, select **Save as New Version**. The **Save** as screen is displayed.
2. In the **Version** box, type the required version number. This field is mandatory.

   > ***Note:*** The format for version number is major.minor. The value of major can be between 1 to 999, and the value of minor can be between 0 to 99. For example, 1.0 or 999.99.

3. In the **Description** box, type the required notes for the new version. This field is optional.
4. Click **OK**. The version of the service is updated to the latest version number.

If you want to select any previous versions, in the **Version** list, select the required version and then click **OK** on the **Alert** dialog box. The version of the service is updated.

22.9 Importing and Exporting Services

Kony Fabric lets you import and export services, along with any associated identity and services, JAR files and data adapters. By using the export function, Kony Fabric bundles the services, JAR files, and data adapters your service uses together in a ZIP file, which can then be imported by other users into their Kony Fabric environments.

The import process will show you any potential conflicts by comparing your current files, JAR files and data adapters with the files in the import package. Any potential conflict will display as an icon on the right side of the service, JAR, or adapter listing in the Import Services window. You can get more information about the potential conflict and how to resolve it by hovering your mouse cursor over the icon.

Some of the possible warnings you might see during the import process are:

- The service, JAR, or adapter already exists in your account and will be overwritten during the import process. Cancel the import to keep your current resource or continue the import to overwrite the existing resource.
- A service with the same name, but a different type exists in the account. Rename or delete the existing resource before importing.
- A read-only service with the same name already exists in the account. Change the service permissions before importing.

22.9.1 How to Import Services

To import services, do the following:

1. In API Management, click the **Identity**, **Integration**, or **Orchestration** tab.
2. Click the **Import Service(s)** button.
3. In the Import Service(s) window, either drag and drop the zip file into the window, or click **Browse** to locate the file on your system.
4. Click the **Services**, **JARs**, and **Adapters** tabs to review the resources that will be imported. Any potential resource conflicts have an icon to the right of the resource. Hover your cursor over the icon for more information.
5. When you have resolved the potential conflicts, Click **Import**.
6. After the resources have imported, click **Close**.

22.9.2 How to Export Services

To export services, do the following:

1. In API Management, click the **Identity**, **Integration**, or **Orchestration** tab.
2. In the Services pane, click the cogwheel button next to the service that you want to export, and then select **Export**.
3. In the Export Service window, a list of the service and all associated services, JARs and Adapters are displayed. Verify the list, and then click **Export**. A zip file is compiled and saved to your local system.

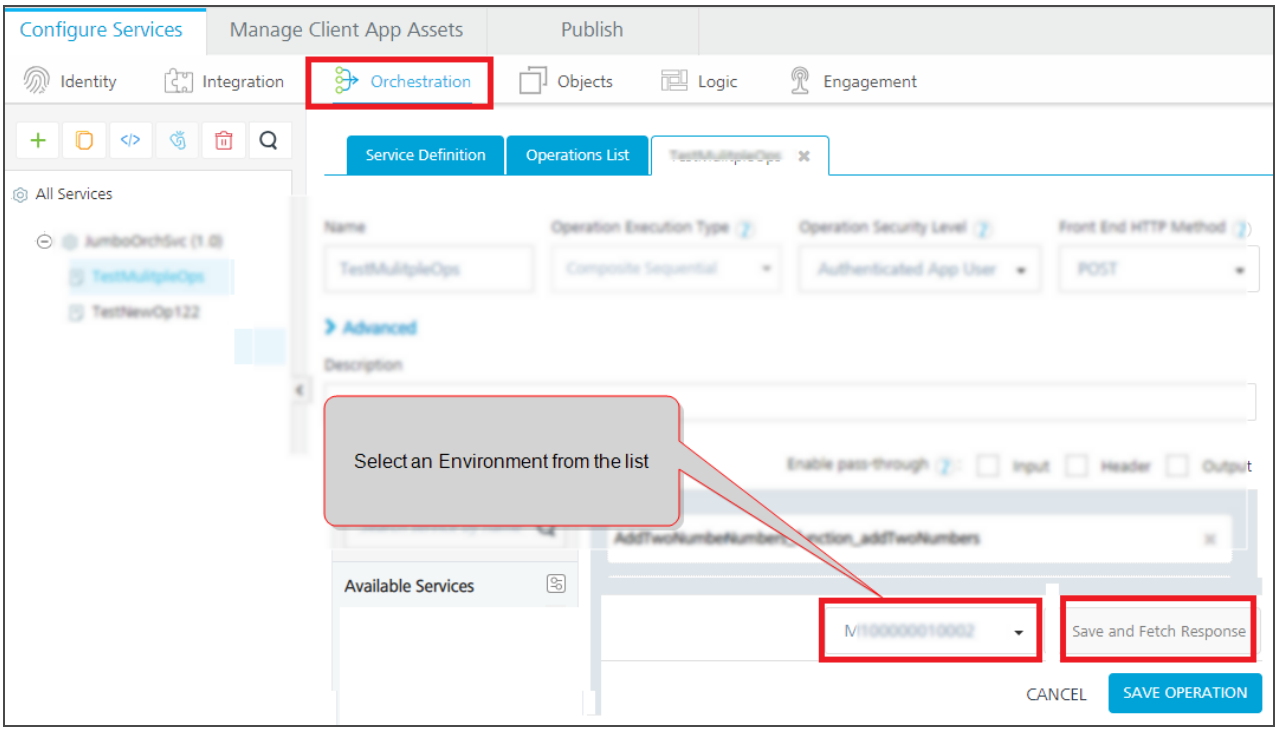
22.10 Test an Orchestration Service Operation

You can test an Orchestration service operation to view the services details of various stages of the service execution for better debugging. The test results are displayed in the **Test results** window.

In the **Log** tab, you can navigate to an individual integration or object service and view the input payloads and response output for each. You can also pass input values in the Request payload under the **Test** tab to each of these services types.

To test an Orchestration service operation, do the following:

1. From the **Select Environment** drop-down list, select an environment from the listed run-time environments configured for the Kony Fabric account.

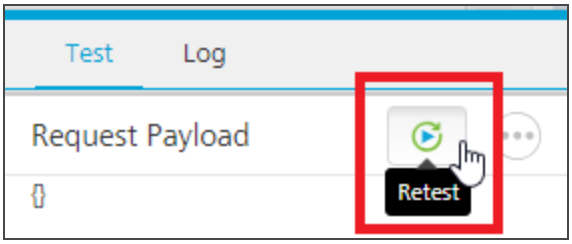


2. Click **SAVE AND FETCH RESPONSE**. The operation gets saved and then the **Output Result** window displays the operation test results.

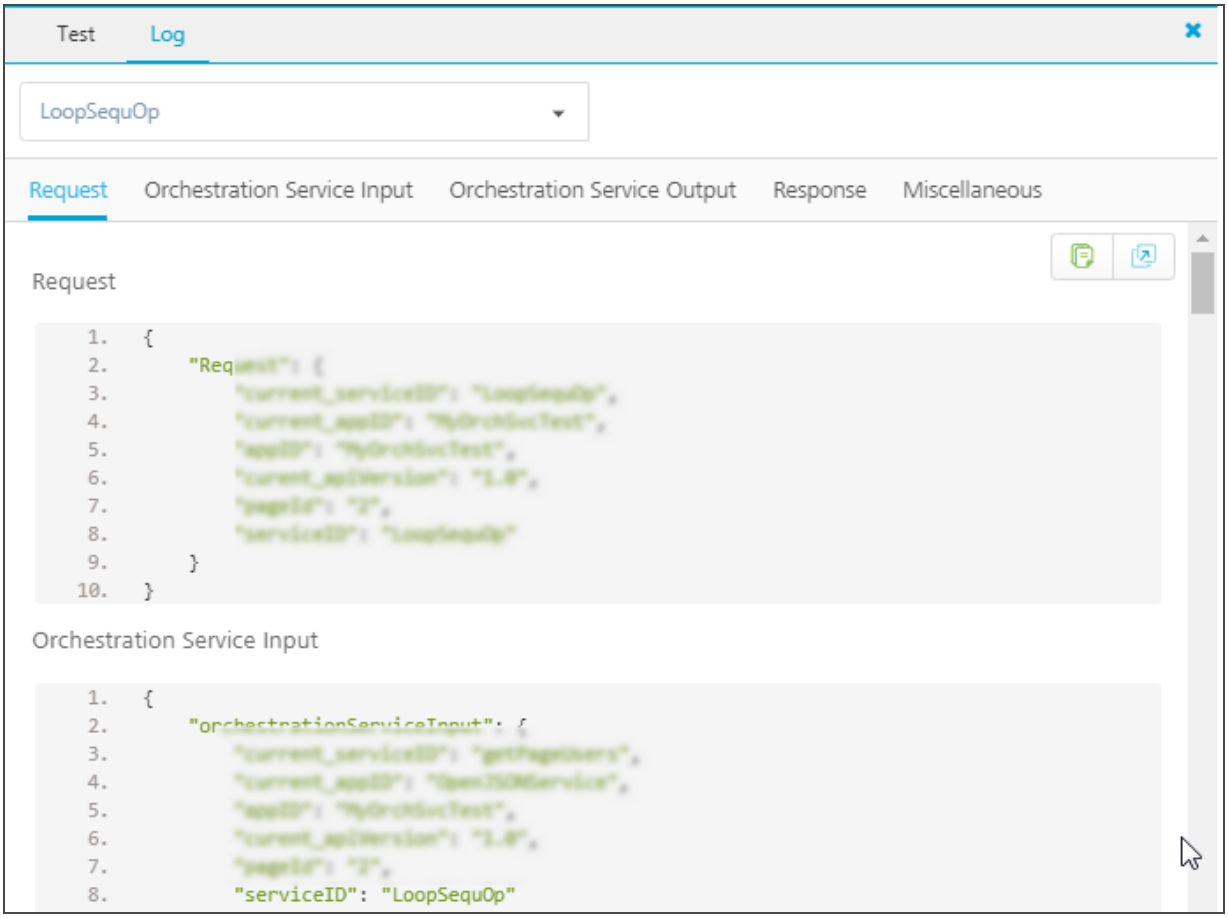
   The **Test Results** window displays two tabs the **Test** and the **Log**.

   The **Test** tab displays two sections such as the **Request Payload** and the **Output Result**.

3. To perform a retest by varying the request payload, click **Retest**.



4. Click the **Log** tab to view details of an entire orchestration operation execution such as request, orchestration service input, orchestration service output, response, and miscellaneous of a service.



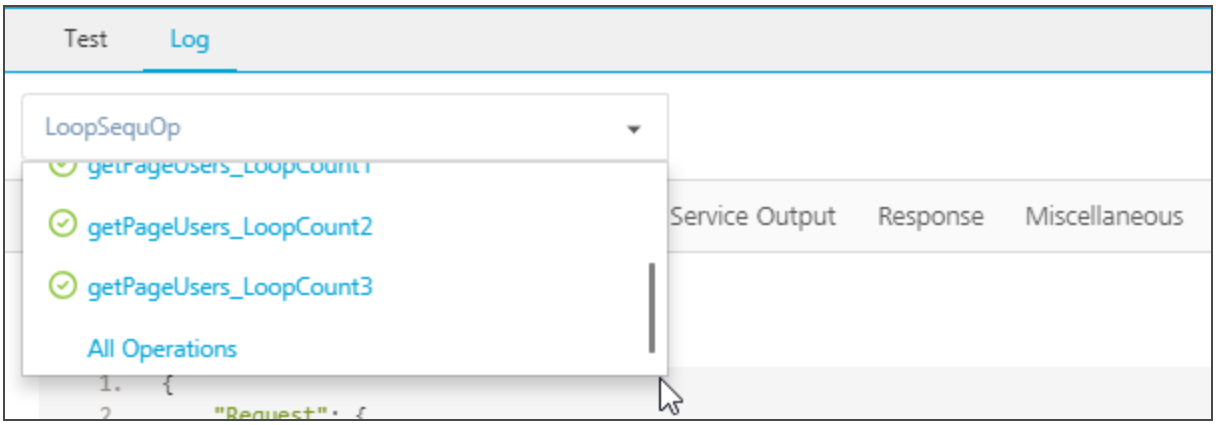
5. In the **Log** tab, you can view the individual operation details of an orchestration operation. Select the required operation:

   - The Orchestration operation shows the request and response parameters of the entire orchestration. This option is selected by default.
   - **Orchestration Service Input** displays the input of Orchestration service.
   - **Orchestration Service Output** displays the output of Orchestration service.
   - Select an **individual operation** from the list to view the request and response parameters of that particular operation.

- Select all **operations** from the list to view the request and response parameters of all the operations called in the orchestration service.

For example, if you have selected an integration service from the list, the following details are displayed:

- Integration Service Input
- Backend Request
- Backend Response
- Integration Service Output



You can perform following actions in this window:

- Click **Copy** to copy the code.
- Click **Expand** to pop over the section as a separate pop-up.
- **Choose number of sections** and content to view in the pop-over.
- **Increase/decrease** the height of the Output Result window.
- **Dock** the sections to different edges by clicking on the section partitions.

23. Workflow

23.1 Overview

Using Workflow, you can visualize and design a backend process. It justifies the low code concept and you can design the backend workflow simply by dragging and dropping different types of nodes and connecting them as per the logic you need.

You can link a backend Workflow with an object either while creating a workflow or through object service. It is linked to a “state” field in the object, and whenever the state field is changed with a create or update on the object (either by using PUT or POST call, or any custom verb based on PUT/POST), the workflow is triggered. The workflow will execute all the subsequent tasks that were defined with in it and completes the entire backend process that is related to the linked object.

There are different kind of nodes in Workflow and each node represents a specific task or an event. These nodes are connected with each other using **Sequence Flows**. When you add these nodes to a workflow and connect them as per the required logic, the complete workflow is implemented automatically whenever it is invoked.

23.2 Working with Workflows

There are different kind of nodes in Workflow and each node represents a specific task or an event. These nodes are connected with each other using **Sequence Flows**. When you add these nodes to a workflow and connect them as per the required logic, the complete workflow is implemented automatically whenever it is invoked.

Different types of Workflow Nodes are as follows:

- **Start** - It is an event that represents where the workflow starts. The Start event has one outgoing flow.
- **End** - It is an event that represents the end of a workflow or a branch of workflow. The End event has one incoming flow.

- **User Task** – It is used to represent that a user action is required in a Workflow. For example: Submitting a loan application, manager approving expense.
- **Service Task** – A Service task is used to invoke a preconfigured service available in the Fabric App.
- **Message Task** – It represents an intermediate event through which you can send notifications to the required recipients. The recipient can be an end user or the manager of the concerned department. For example, if the Workflow reaches a User Task, you can send an email notification to the respective end user stating that there is a pending user action.
- **Exclusive Gateway** – Exclusive Gateways are used to model decision in a process by making exclusive (XOR) disjunction. An exclusive gateway can have multiple outgoing sequence flows and each outgoing sequence flow has its own decision condition. It evaluates the decision condition of each outgoing sequence flow and the matched condition is executed.

To go to the **Workflow** tab from the Kony Fabric Console dashboard, click **Add New** or select any existing Kony Fabric app, and click the **Workflow** tab. The Workflow tab landing page appears.

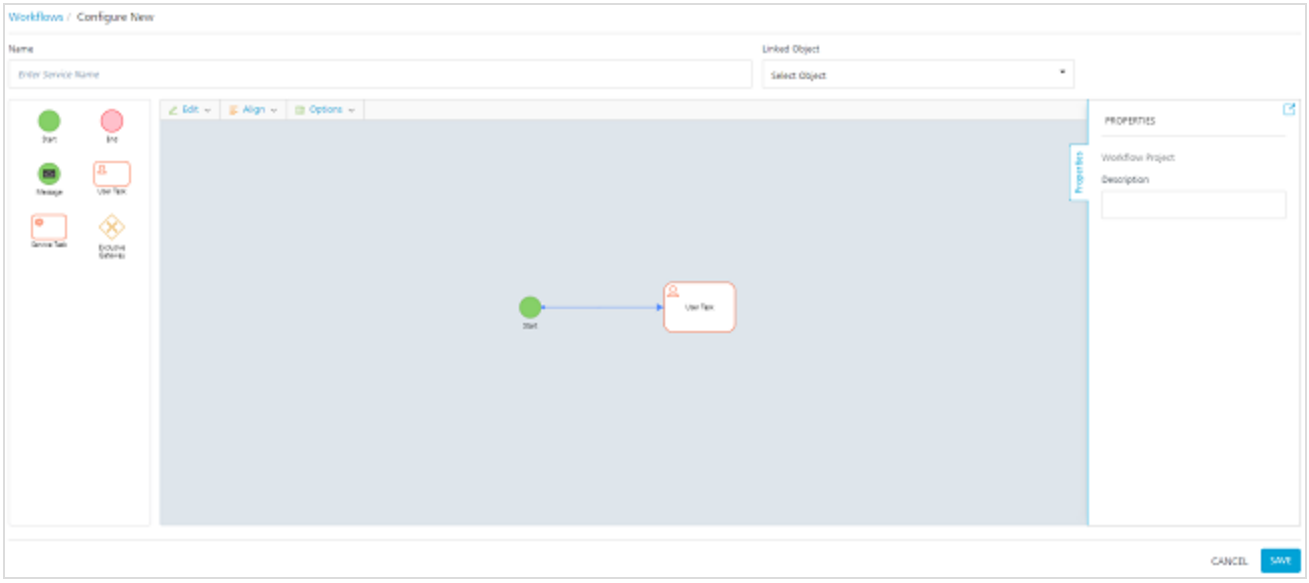
> ***Note:*** In V9-Preview, if the Kony Fabric app is not associated with any workflow, in the address bar, remove the existing service and type “Workflow”. For example, if the address bar has "https://manage.kony.com/console/#/apps/identity" link, then change it to "https://manage.kony.com/console/#/apps/Workflow".

You can do the following from the Workflow landing page:

- Create a new workflow.
- Link an existing workflow in the account to the current Kony Fabric app.
- Manage existing workflows.

23.3 Create a New Workflow

Click **Configure New** from the Workflow landing page, the Configure new screen appears.



You can do the following in the **Configure New** screen:

- In the **Name** field, type a unique name for the new Workflow service.
- From the **Linked Object** list, select the required object to which you want to link the new Workflow service. The list contains two options:
    - **Use Default** - It creates a new object service with a default name as per the new Workflow service name and this workflow is linked to that Object. The new object service created will be a storage object service.
    - **Use Existing** - You can select this option if you want to link the new workflow with any existing object available in the current Fabric app. The linked object service can be a storage object service, RBMS service or an SDO (Integration or Orchestration service) service.
      Click **Use Existing**, and from the **Existing Services** screen select the required Object service, Object, Field, and then click **Add**. The new Workflow is linked to an existing Object service.

> ***Note:*** Make sure that you designate a Field or create a new Field in the required object to store the workflow state if you want to use an existing workflow.

- From the **Nodes** pane, double click on required nodes and drag and drop them on the canvas area. To know more about the activities that you need to perform for each node type, refer to Nodes.

> ***Note:*** A workflow should begin with a **Start** event and finish with an **End** event.

- Link the nodes as per the required flow using **Sequence Flows**. To know more about the activities, you can perform on a Sequence Flow, refer to Sequence Flows.
- The other features available in the canvas area are as follows:
  - **Edit**: Click **Edit** to perform Undo, Redo, Cut, Copy, Paste, Delete, and Select All actions on the nodes that are in the canvas area.
  - **Align**: Click **Align** to arrange the nodes as required. The options in **Align** are Align Left, Align Right, Align Top, Align Bottom, Align Center X, and Align Center Y.
  - **Snap and Grid**: Use the feature to change the canvas layout. Choose **Grid** to get a grid view on the canvas area. If you select **Snapping** and try to place a node on the canvas area, the node would align itself with the closest grid line. This will help you align your nodes properly against each other while creating a workflow.

    > ***Note:*** Grid Lines exist on the canvas area and are visible only when you choose **Grid** from the **Snap and Grid** list.

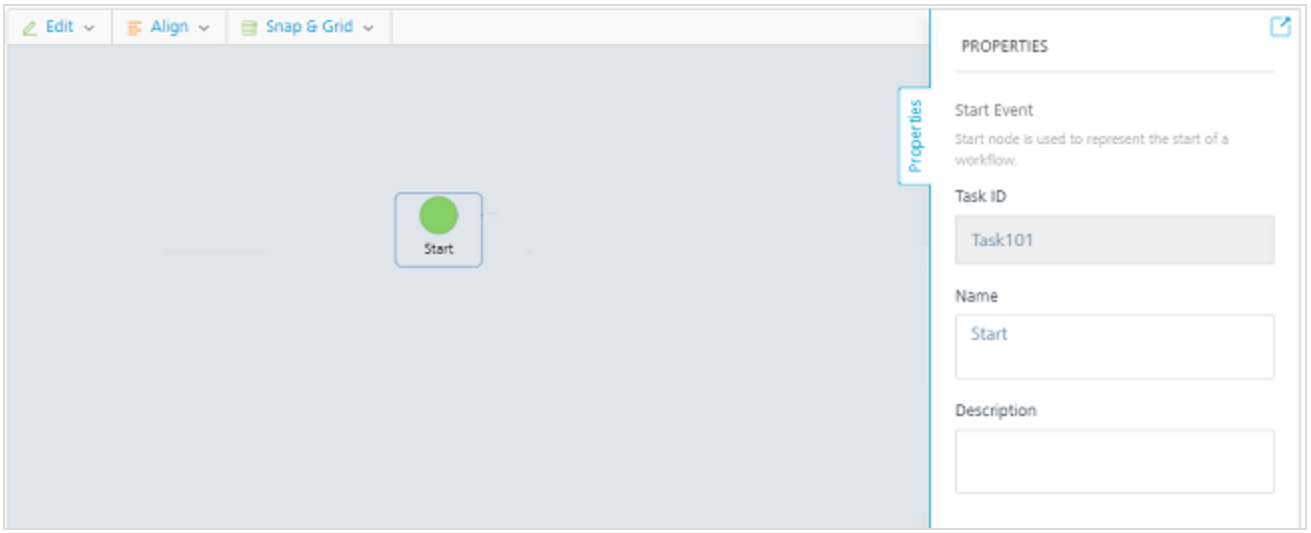
- Click **Save**. A new Workflow service is created and is added to the Workflow's landing page.

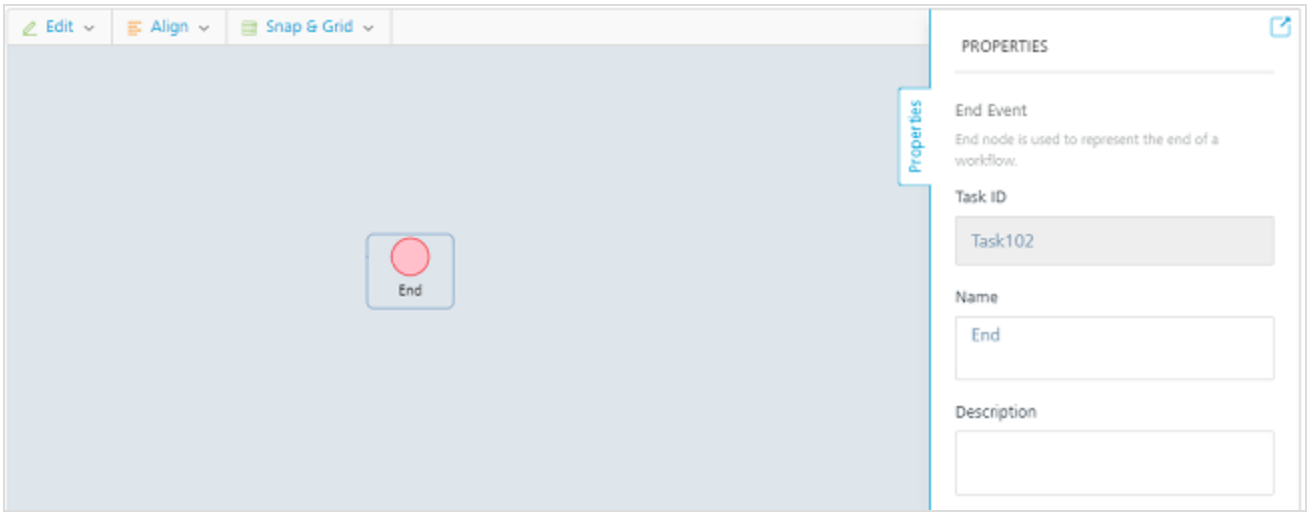
23.4 Nodes

After the nodes are placed in the canvas area, you can manage the properties of a node from the Properties pane. These properties vary for each type of node. Defining these properties is crucial in a workflow as the behavior of each node depends on these properties when a workflow is triggered.

23.4.1 Start and End

The Start/End Event's property pane contains the following fields:

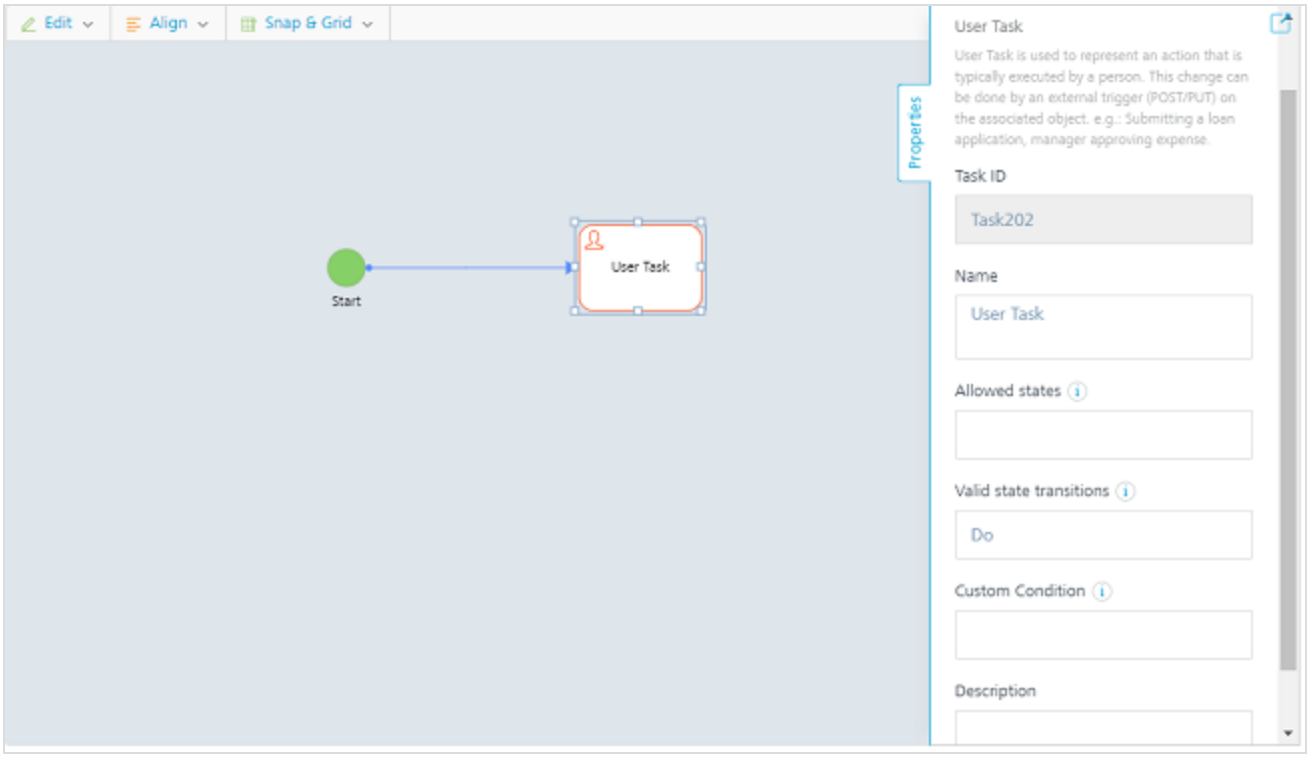




- **Task ID**: A task ID is automatically allotted to the Start and End Event when you drag and drop them in the canvas area. You cannot edit this field.
- **Name**: Displays the name of the node. Modify it as per the activity the node performs.
- **Description**: You can write description for this node.

23.4.2 User Task

The User Task's property pane contains the following fields:



- **Task ID**: A task ID is automatically allotted to the User Task when you drag and drop it in the canvas area. You cannot edit this field.
- **Name**: Displays the name of the node. Modify it as per the activity the node performs.
- **Allowed States**: Mention the required status that represents the state(s) of the Workflow before

the user begins any user task. For example: Loan application has to be in Submitted state for the bank manager to review the application.

> ***Note:*** The **Allowed State** in the first user task can be blank as the Allowed State is not applicable for the first user task.

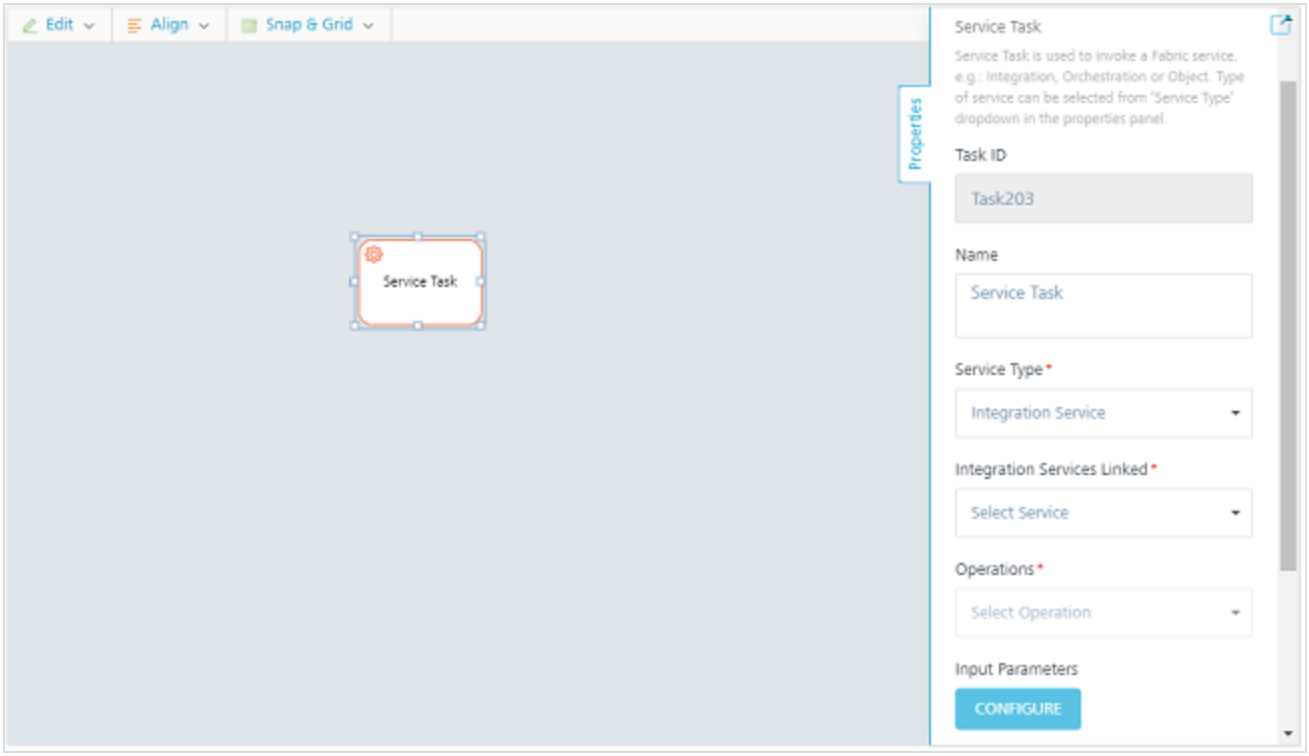
- **Valid State Transitions**: Mention the required status that represents the state(s) of the Workflow after the user has executed the user task. This must match with the status in the respective PUT or POST call. If the state mentioned in **Valid State Transitions** does not match with state available in the respective POST or PUT call, the workflow will not proceed ahead. For Example: After the bank manager has reviewed the loan application it must be either Approved or Rejected. It cannot move to the Draft state.

  > ***Note:*** The state mentioned in **Valid State Transitions** must exactly same as the workflow state received from the client app.

- **Custom Condition**: If you want to provide any additional validation for the User Task, you can provide it here. For example, if you want an additional validation where the exit criteria must contain the relevant manager's name, you can give a logical expression like BACKEND_ RESPONSE.balance = = >10,000.
- **Description**: You can write the description for this node.

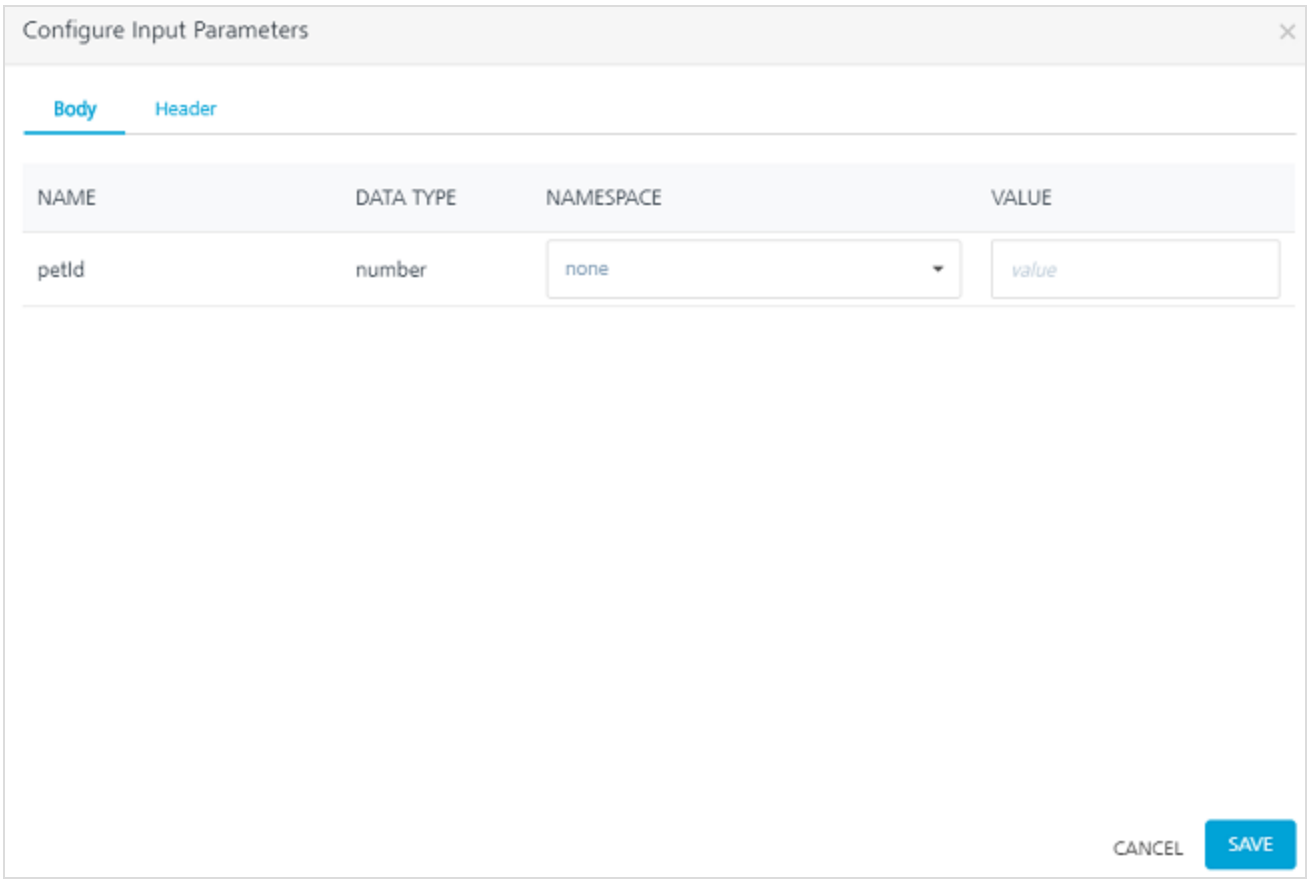
23.4.3 Service Task

The Service Task's property pane contains the following fields:



- **Task ID**: A task ID is automatically allotted to the Service Task when you drag and drop it in the canvas area. You cannot edit this field.
- **Name**: Displays the name of the node. Modify it as per the activity the node performs.
- **Service Type**: It lists all the service types. Select the required service type from the list
- **Services Linked**: As per the selected service type, all the linked services in the current Kony Fabric App are displayed. Select the required service from the list.
- **Operations**: As per the selected service, all the linked operations of the selected services are displayed. Select the required operation from the list.
- If you have selected Object Services, select the required object and verb from the **Objects** and **Verbs** lists respectively.
- Click **Configure** under **Input Parameters** to manage the Request Input parameters. The Configure Input Parameters screen appears.

- This screen contains two tabs, namely, **Body** and **Header**. These tabs display the respective parameters as per the selected service and operations. You can configure the following fields in these tabs:



  - **Namespace**: Select the required Namespace from the list. The list contains different data sources from which the data can be accessed. When you select a namespace for any parameter, the selected namespace will be accessed to retrieve the data related to that parameter.
    The namespaces available for input parameters are as follows:

    - **IDENTITY**: It denotes that the input parameter is from the user or security attributes from the identity service response.
    - **DEVICE_REQUEST**: It is a data source that represents the parameters that are

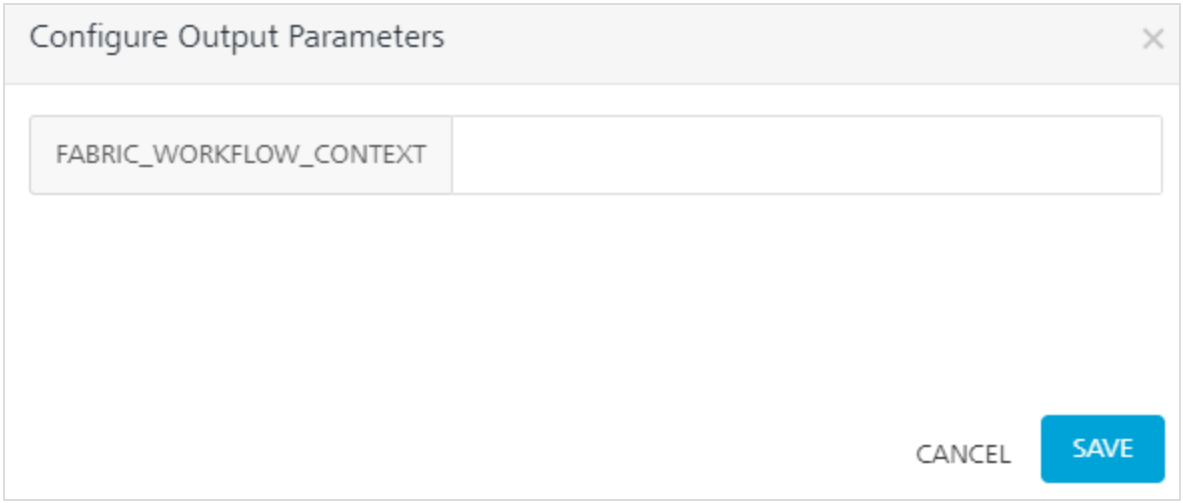
received from client app request.

- **FABRIC_WORKFLOW_CONTEXT**: It is a data source that represents the parameters that are received from the persistent store of the current workflow instance.
- **BACKEND_RESPONSE**: It is a data source that represents the parameters that are received from the actual backend data.
- **SESSION**: It states that the source of the parameter is from the session data.

- **Value**: Type the variable from where the data of the input parameter must be retrieved from the data source.

> ***Note:*** If you have selected None in Namespace list, the data provided in the **Value** column will be considered as the data for the respective parameter.

- Click **Configure** under **Output Parameters** to manage the output response parameters. The Configure Input Parameters screen appears.



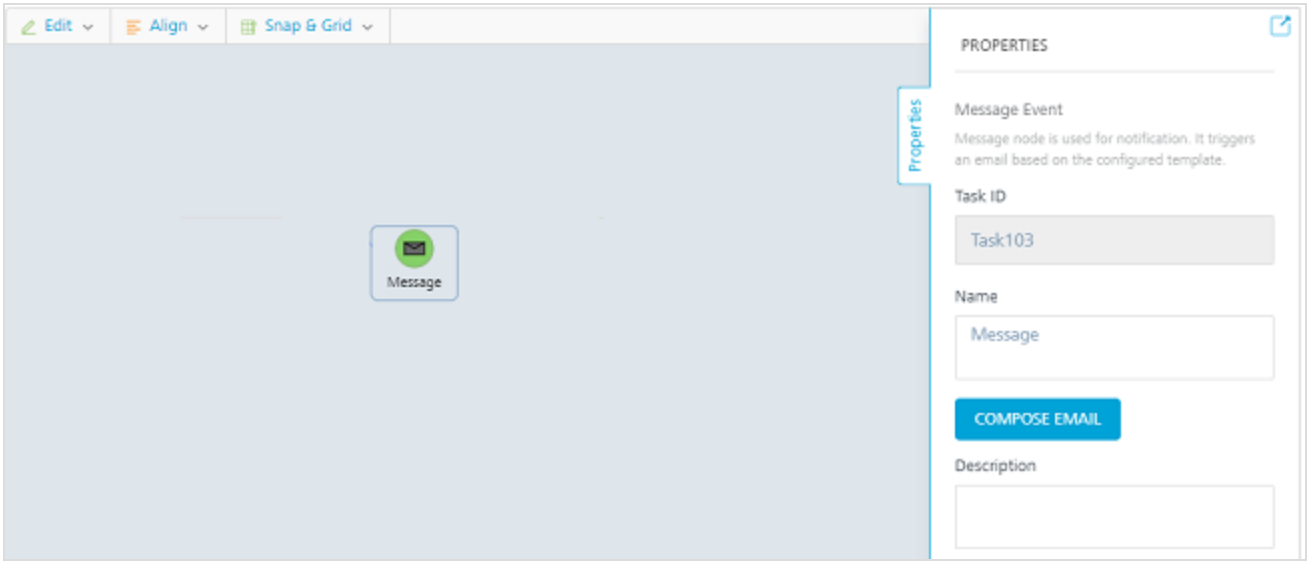
The output parameters are dumped into a single variable under namespace (**FABRIC_WORKFLOW_CONTEXT**) as JSON. Individual elements of the output can be accessed by

using a dot(.) notation with a key. For example: Assume your output `{"name":"John", "age":30, "car":null }` is stored in Variable Profile under FABRIC_WORKFLOW_CONTEXT.Profile, you can access age by using FABRIC_WORKFLOW_CONTEXT.Profile.age.

- **Description**: You can write the description for this node.

23.4.4 Message Task

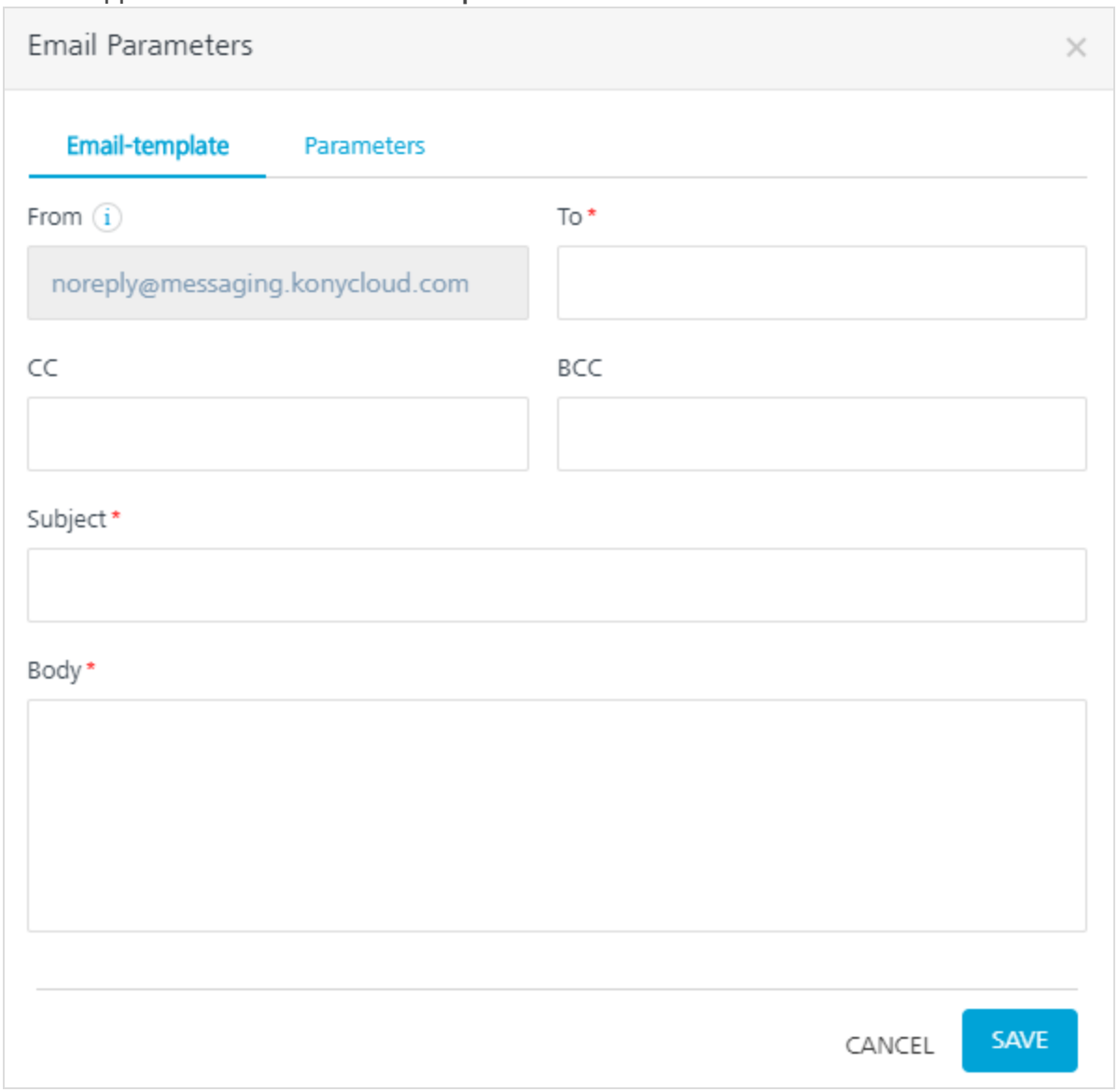
The Message Task's property pane contains the following fields:



- **Task ID**: A task ID is automatically allotted to the Service Task when you drag and drop it in the canvas area. You cannot edit this field.

  Name: Displays the name of the node. Modify it as per the activity the node performs.

- **Compose Email**: Click **Compose Email** to create an email notification. The Email Parameters screen appears. It contains **Email-Template** and **Parameters** tab.



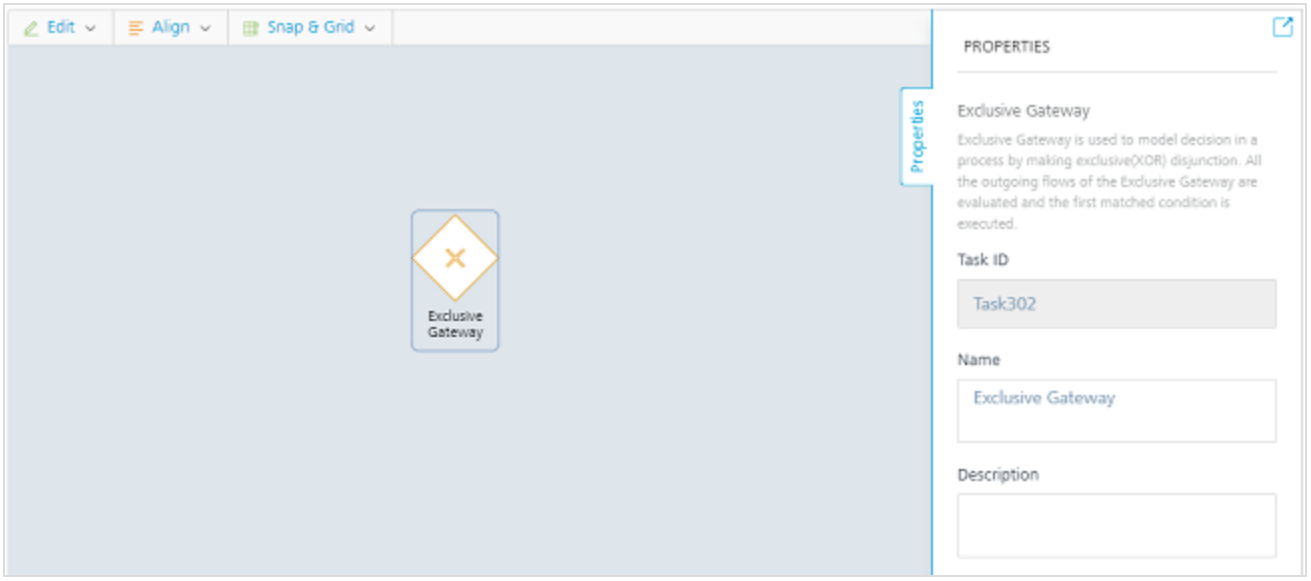
- In the **Email Template** tab, type the details in To, Subject, CC, BCC, and Body fields to configure the required email. You can also set dollar ($) parameters in your email and pass values dynamically at run-time.

> ***Note:*** You can add a maximum five email IDs each in To, CC and BCC. If you add more, only the first five email IDs will get the email notification and the remaining email IDs will be ignored.

- If you have set dollar ($) parameters in the email, in the **Parameters** tab, you can define the namespace and values for those dollar ($) parameters. For example: If you have used a dollar parameter $amount in the email template, from the Parameters tab you can define the **Name**, **Namespace**, and **Value** as amount, BACKEND_RESPONSE, and NetAmount respectively.
- **Description**: You can write the description for this node.

23.4.5 Exclusive Gateways

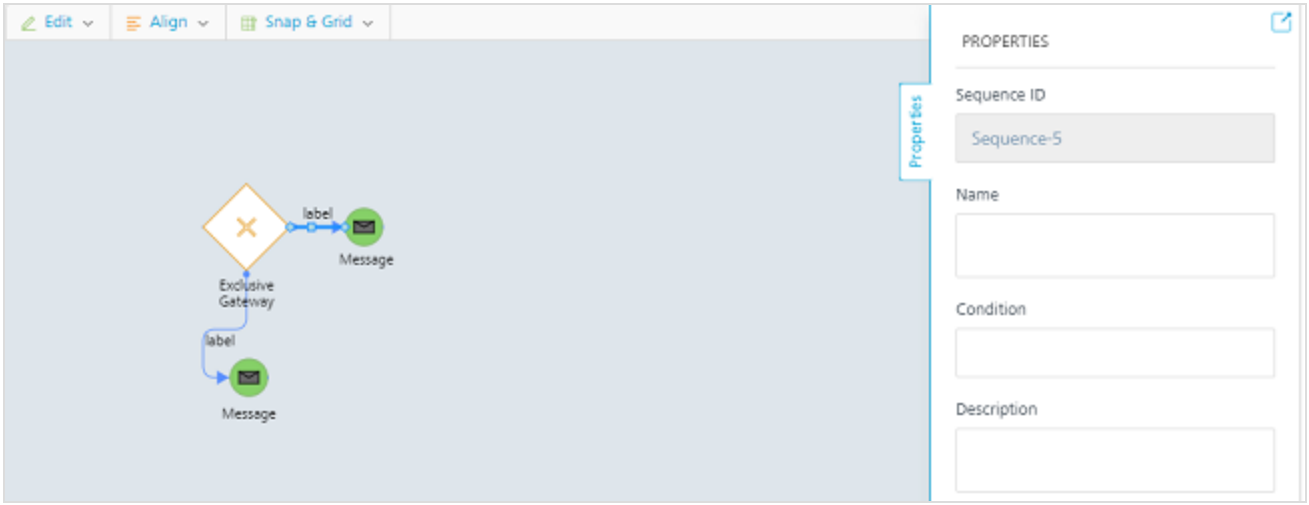
The Exclusive Gateway's property pane contains the following fields:



- **Task ID**: A task ID is automatically allotted to the Service Task when you drag and drop it in the canvas area. You cannot edit this field.
- **Name**: Displays the name of the node. Modify it as per the activity the node performs.
- **Description**: You can write the description for this node.

23.5 Sequence Flows

Sequence Flows act as connectors between the nodes. Whenever you place a **Sequence Flow**, a sequence ID is allotted to it. You can view this sequence ID from the **Properties** pane and add description for the flow if required.



The Sequence Flows coming from Exclusive Gateways also contains an additional **Condition** field in the Properties pane. You can define the output criteria of the selected **Sequence Flow** here. From the **Exclusive Gateway**, the workflow will be redirected towards the first **Sequence Flow** that meets the output criteria defined in the **Condition** field.

23.6 Use Existing Service to Create a New Workflow

This feature helps you to use an existing Workflow in a Kony Fabric account and use it to configure a new Workflow. You can either clone or add existing workflow and make changes to them accordingly.

To create a new Orchestration service from the available services, perform the following steps:

- In the **Workflow** service tab, click **Use Existing** or in the left pane click the “+” icon and select **Use Existing**. The **Existing services** screen is displayed.
- Select the required services from the **Existing services** screen and click **Clone** or **Add**. The

Clone Service or Add Service status screen appears.

- **Clone**: It creates a duplicate of the selected service. The changes made to the duplicate service will not affect the original service.
- **Add**: It adds the selected service to the new Kony Fabric app. The changes made to the service will affect all the apps using the service.
  If the service is part of any published app, you must unpublish the service to rename it.

> ***Note:*** If the list is long, you can search for the required service with **Search** option.

- After the **Clone** or **Add** process is complete, the service is added to the Orchestration services list.
- Click the newly added service or open the Contextual menu and click **Edit** to configure the details of the service. For more information on configuring the details, refer to Create a Workflow.

23.7 Manage Workflows

All the services related to a service type are listed on the landing page of the respective service type. You can manage the details of a service from the Contextual menu available adjacent to each service. The following options are available in the Contextual Menu:

- **Edit** - Click to edit the details of a selected service. After you edit a service, republish all the apps that use this service to apply the changes.
- **Clone** - Duplicates an existing service. Clone a service to create a different version of the same service. Changes made to a cloned service will not affect the original service. The name of a cloned service indicates that it is a copy of an existing service.
- **Sample Code** - Generates dynamic code for each SDK type based on the configuration of a service. You can use the code in your mobile app. For example, generate the sample code for an orchestration service from Kony Fabric. Then use that code in the mobile app to invoke the orchestration service instance.

- **Delete** - Deletes a selected Workflow from Kony Fabric Console. You cannot delete a service if the service is in use. A service in use is a service that is referenced by a Kony Fabric app or another service or a Sync scope.

  > ***Note:*** When you delete a service that has multiple versions, only the active version is deleted.

- **Console Access Control** - You can manage the users who can access this service from here. To know more, refer to Console Access Control.
- **Export as XML** - Exports the current version of a service in the form of an XML file.
- **Export** - Exports the service details in the form of a zip file. You can import this zip file to another Kony Fabric app and use it. For more information, refer to Export and Import an Application.

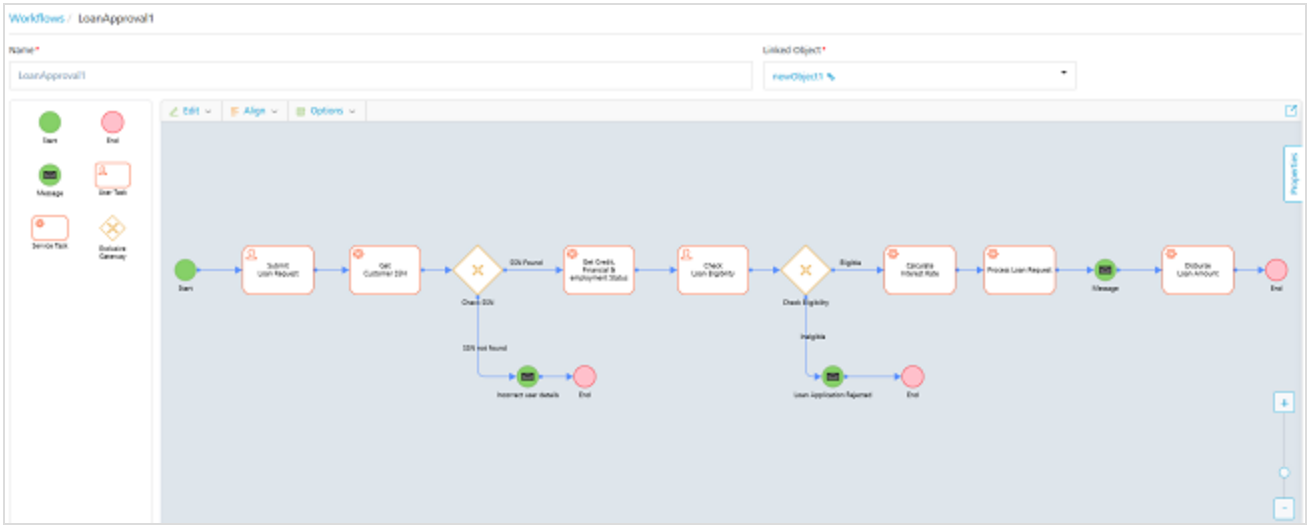
> ***Note:*** To view the Usecase related to Workflow and the implementation of the Usecase, refer to Workflow Implementation.

> ***Note:*** To view the execution status of a workflow service by using Kony App Services Console, refer to Kony App Services Console > Workflow Services section.

23.8 Workflow Implementation

23.8.1 Usecase

Let us create a basic Workflow that is invoked whenever a new loan application is submitted. Using Workflow, we can automate the different phases through which the loan application passes before approval or rejection. Whenever a manual input is required, the Workflow will pause at that phase and the process will resume automatically after an appropriate action is taken by the applicant.



In this example, the loan application process will have the following steps:

- When a loan application is submitted, the SSN information is retrieved automatically based on the information provided in the application.
- If the SSN is invalid, an email will be sent to the applicant stating that the details in the application are incorrect.
- If the SSN is valid, the credit score is retrieved automatically, and the financial and employment status is verified by the bank.
- After the financial details are verified, the applicant's loan eligibility is calculated.
- If the applicant is not eligible, an email will be sent to the applicant stating that the loan application was rejected as the applicant is not eligible.
- If the applicant is eligible, the interest rate is calculated automatically.
- The loan request is processed based on the calculated interest rate and a message is sent to the applicant stating that the loan request is processed.
- The loan amount is disbursed to the applicant's bank account.

23.8.2 Loan Application Workflow

23.8.2.1 Pre-requisites

To create a workflow for the steps that were discussed in this example, you must have the necessary Integration Services configured in your Kony Fabric console.

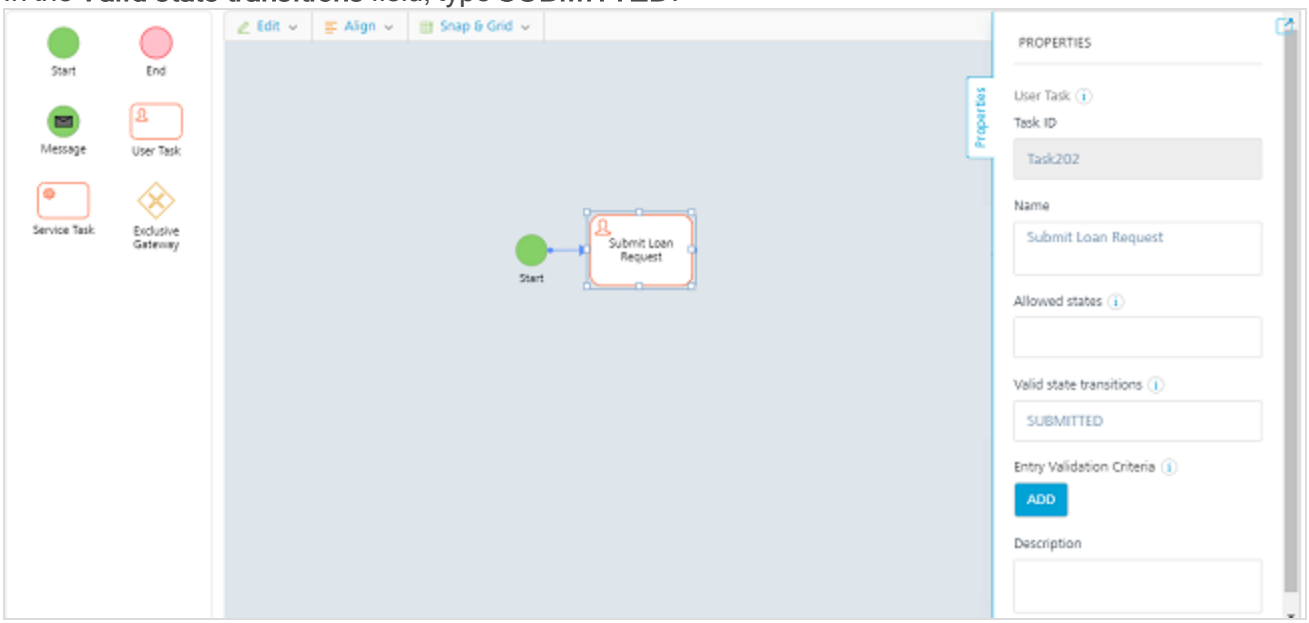
Click here to download the Integration Services required to create the workflow and import the same into your Kony Fabric console.

23.8.2.2 Create Workflow

To create the Loan Application workflow, do the following:

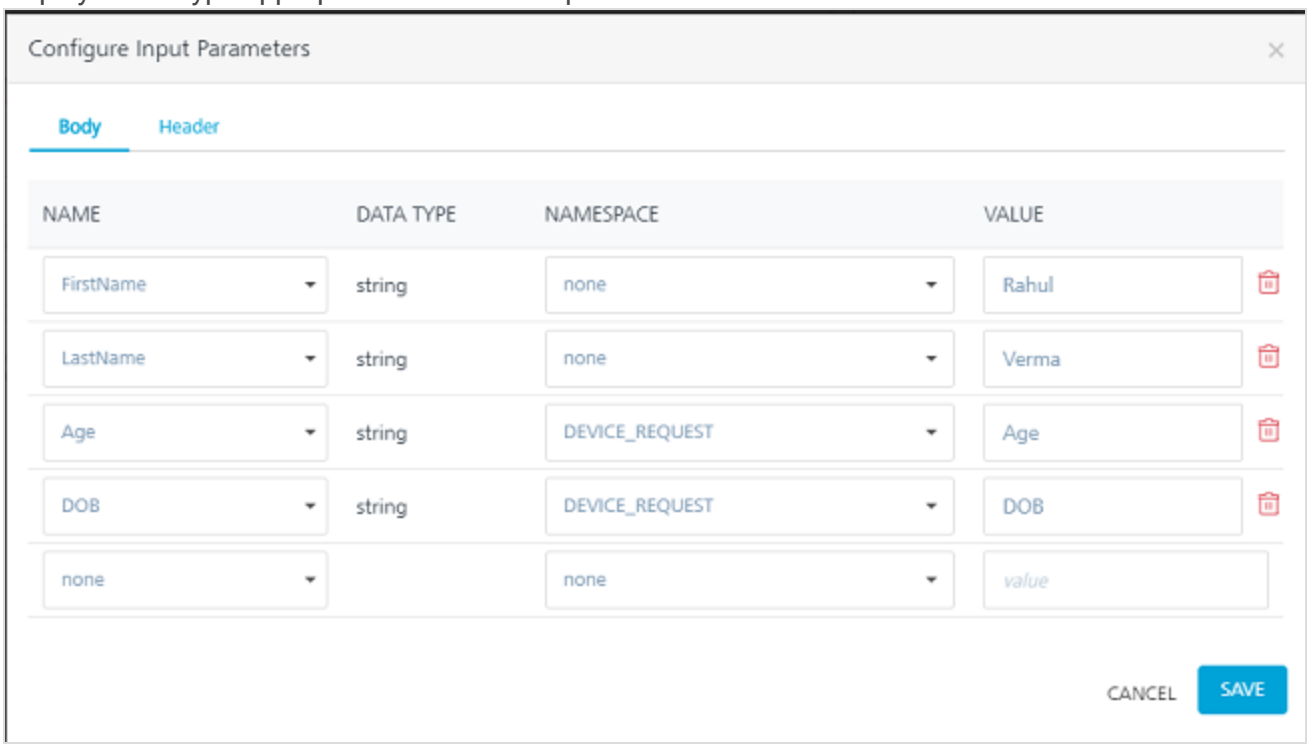
- From the dashboard in Kony Fabric console, click the Fabric app where you have imported the Integration Services related to this use case, and click the Workflows tab. The Workflow's landing page appears.
- Click **Configure New** from the landing page, the Configure New screen appears with a Start and User Task node placed in the canvas area by default.
- In the **Name** field, type a unique name for the new Workflow service. For example: **LoanApplication**.
- From the **Linked Object** dropdown, click Use Default. It creates a new object service. The name of the new object service will depend on the name of the new Workflow service name and this workflow will be linked to said Object service.
- Select the User Task in the Workflow canvas area and click the Properties pane. You can do the following in the Properties pane:
    - In the **Name** field, type the required name of the user task based on the activity it handles. For example: **Submit Loan Request**.

- In the **Valid state transitions** field, type **SUBMITTED**.

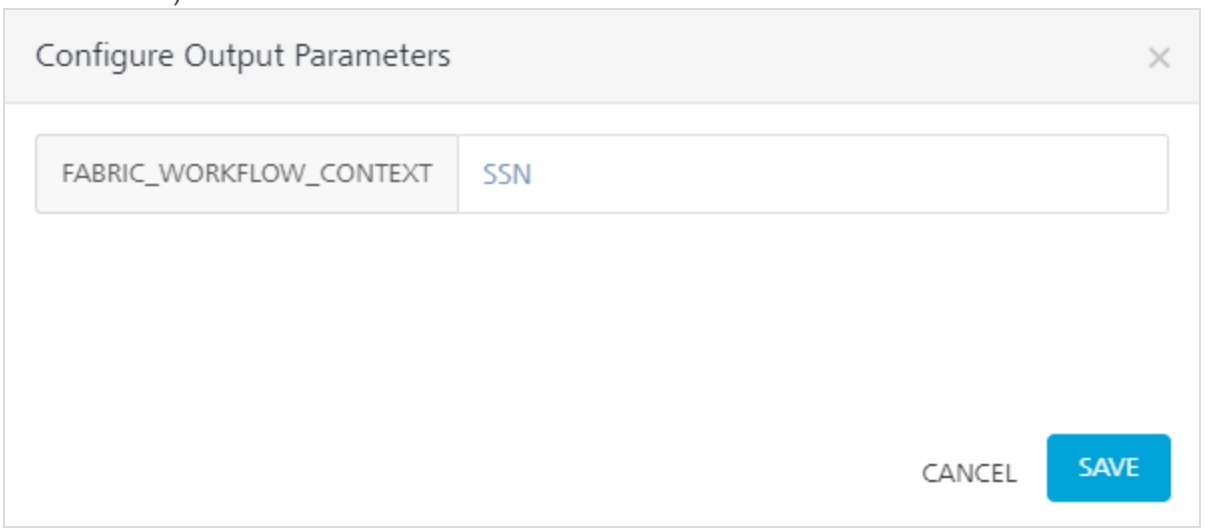


- Drag and drop a Service Task next to the Submit Loan Request user task and connect them. Click the Properties pane. You can do the following in the Properties pane:
  - In the **Name** field, type the required name of the service task based on the activity it handles. For example: **Get Customer SSN**.
  - From the **Service Type** list, select **Integration Service**.
  - From the **Integration Services Linked** list, select **Loan**.
  - From the **Operations** list, select **SSN**.
  - Click Input Parameters **Configure** to manage the integration service's request input parameters. In the **Namespace** column, select **None** for all the input parameters

displayed and type appropriate data for each parameter in the **Value** column.



- Click Output Parameters **Configure** to manage the integration service's output response. By default, response from each service task is saved in the Fabric_Workflow_Context namespace. Type **SSN** which acts as a variable for the namespace (Fabric_Workflow_Context.SSN).

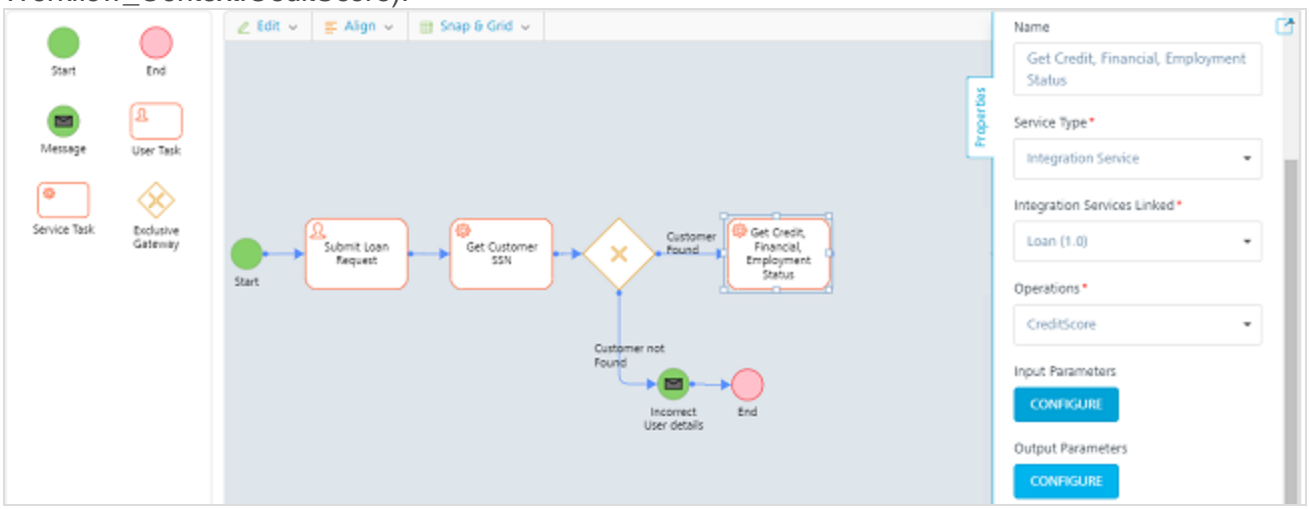


- Drag and drop an Exclusive Gateway next to the **Get Customer SSN** service task and connect them. Click the Properties pane. You can do the following in the Properties pane:
    - In the **Name** field, type the required name of the exclusive gateway based on the activity it handles. For example: **Check SSN**.
- As **Check SSN** is an exclusive gateway, two flows emerge from here. One flow determines the path of the workflow if the **SSN is valid**, and the other flow determines the path of the workflow if the **SSN is invalid**.
- Click the "SSN is valid" flow, and in the Properties pane, type `SSN!="Null"` in the **Condition** field.
- Click the "SSN is invalid" flow, and in the Properties pane, type `SSN="Null"` in the **Condition** field.
- Connect the **SSN is invalid** flow with a Message Task and click the Properties pane. You can do the following in the Properties pane:
    - In the **Name** field, type the required name of the message task based on the activity it handles. For example: **Incorrect User Details**.
    - Click Compose Email and configure the email template.

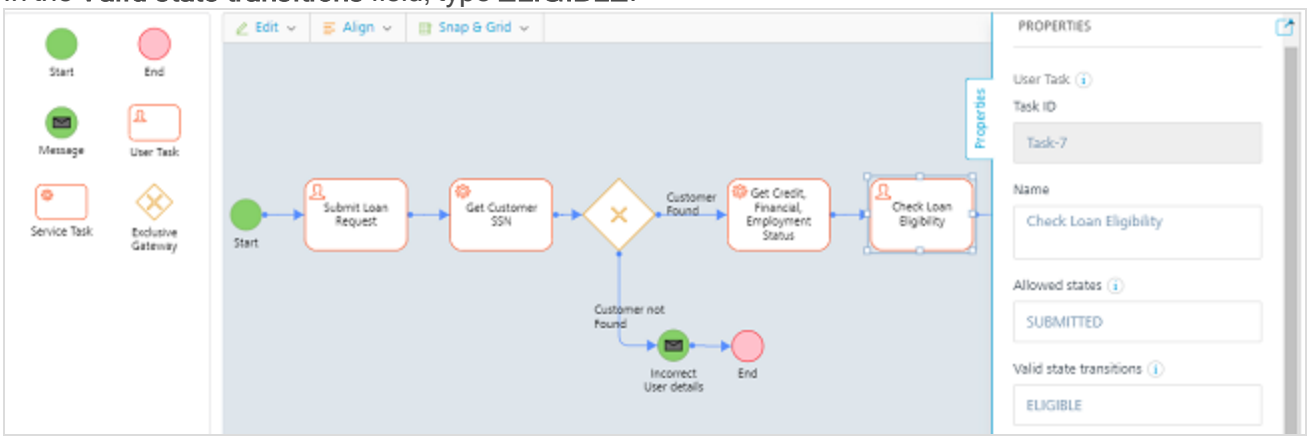
- In the Email-Template tab, type the TO, CC, and the name of the recipient in Body with appropriate message. You can provide $ variables like $email1, $email2, and $FirstName respectively in the required fields.
- In the Parameters tab, in the Name column, type the $variable given in email template, select the **Namespace** value as **None** and provide the required $variable data in the **Value** column.

- Place an **End** node after the **Incorrect User Details** message task to finish the **SSN is invalid** flow path.
- Connect the **SSN is valid** flow with a Service Task and click the Properties pane. You can do the following in the Properties pane:
  - In the **Name** field, type the required name of the service task based on the activity it handles. For example: **Get CreditScore, Financial and Employment Status**.
  - Link this task to an Integration service and select **Loan** and **CreditScore** in the respective dropdowns as you have done in the previous service task.
  - Click Input Parameters **Configure** to manage the integration service's request input parameters. In the **Namespace** column, select Fabric_Workflow_Context the input parameter and type SSN in the **Value** column as the output of the **Get Customer SSN** was stored in Fabric_Workflow_Context.SSN.
  - Click Output Parameters **Configure** to manage the integration service's output response. By default, response from each service task is saved in the Fabric_Workflow_Context namespace. Type **CreditScore** which acts as a variable for the namespace (Fabric_

Workflow_Context.CeditScore).



- Drag and drop a User Task next to the **Get CreditScore, Financial and Employment Status** service task click the Properties pane. You can do the following in the Properties pane:
  - In the **Name** field, type the required name of the user task based on the activity it handles. For example: **Check Loan Eligibility**.
  - In the **Allowed States** field, type **SUBMITTED**.
  - In the **Valid state transitions** field, type **ELIGIBLE**.

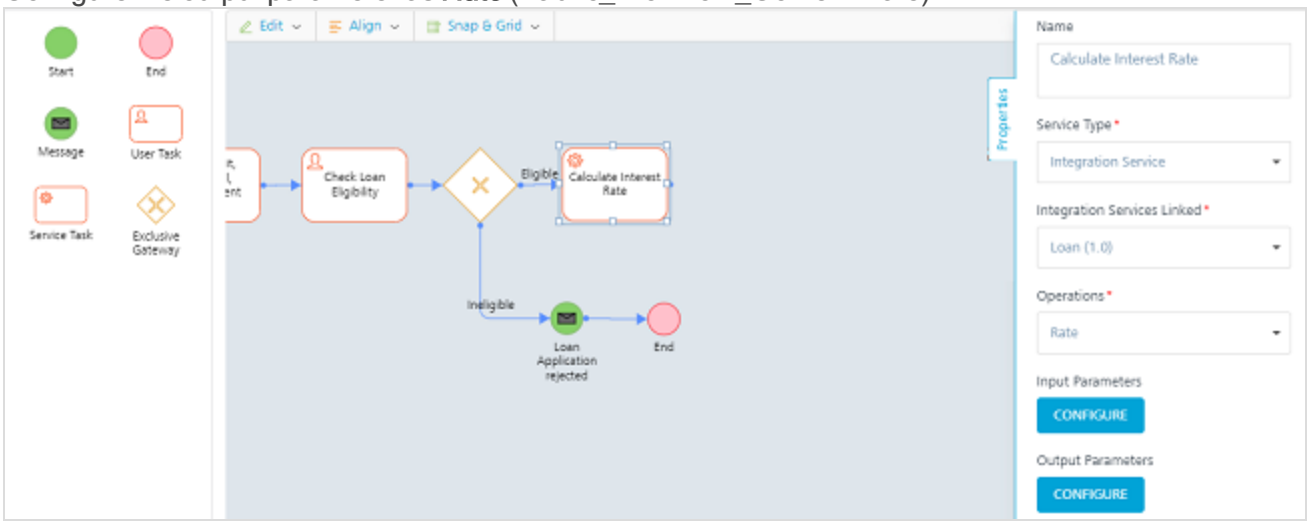


- Drag and drop an Exclusive Gateway next to the **Evaluate Loan Eligibility** user task click the Properties pane. You can do the following in the Properties pane:

    - In the **Name** field, type the required name of the exclusive gateway based on the activity it handles. For example: **Check Eligibility**.

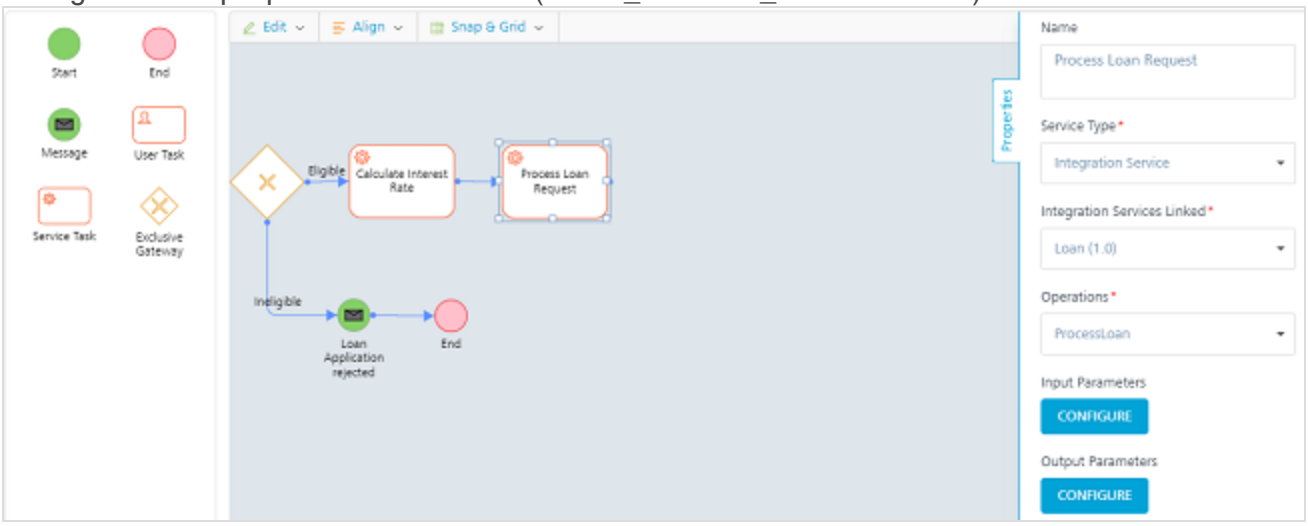
- As **Check Eligibility** is an exclusive gateway, two flows emerge from here. One flow determines the path of the workflow if the loan is **Eligible**, and the other flow determines the path of the workflow if the loan is **Ineligible**.
- Click the **Eligible** flow, and in the Properties pane, type `BACKEND_RESPONSE.WorkflowField == 'ELIGIBLE'` in the **Condition** field.
- Click the **Ineligible** flow, and in the Properties pane, type Backend_Response.`WorkflowField!= 'ELIGIBLE'` in the **Condition** field.
- Connect the **Ineligible** flow with a Message Task and click the Properties pane. You can do the following in the Properties pane:
    - In the **Name** field, type the required name of the message task based on the activity it handles. For example: **Loan Application Rejected**.
    - Click Compose Email and configure the email template.
    - In the Email-Template tab, type the TO, CC, and the name of the recipient in Body with appropriate message. You can provide $ variables like $email1, $email2, and $FirstName respectively in the required fields.
    - In the Parameters tab, in the Name column, type the $variable given in email template, select the **Namespace** value as **None** and provide the required $variable data in the **Value** column.
- Place an **End** node after the **Loan Application Rejected** message task to finish the **Ineligible** flow path.
- Connect the **Eligible** flow with a Service Task and click the Properties pane. You can do the following in the Properties pane:
    - In the **Name** field, type the required name of the service task based on the activity it handles. For example: **Calculate Interest Rate**.

- Link this task to an Integration service and select **Loan** and **Rate** in the respective dropdowns as you have done in the previous service task.
- Configure the output parameter as **Rate** (Fabric_Workflow_Context.Rate).



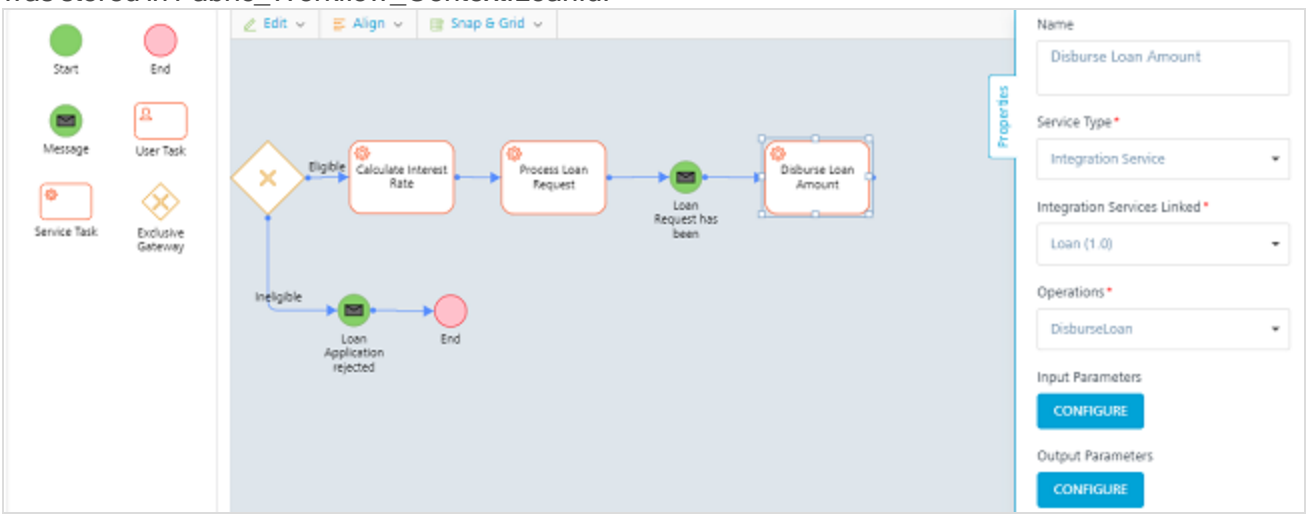
- Drag and drop a Service Task next to the **Calculate Interest Rate** service task and click the Properties pane. You can do the following in the Properties pane:
  - In the **Name** field, type the required name of the service task based on the activity it handles. For example: **Process Loan Request**.
  - Link this task to an Integration service and select **Loan** and **ProcessLoan** in the respective dropdowns as you have done in the previous service task.
  - Configure the input parameter as Rate as the output of the **Calculate Interest Rate** was stored in Fabric_Workflow_Context.Rate.

- Configure the output parameter as **LoanId** (Fabric_Workflow_Context.LoanId).



- Drag and drop a Message Task next to the Process Loan Request service task and click the Properties pane. You can do the following in the Properties pane:
  - In the **Name** field, type the required name of the message task based on the activity it handles. For example: **Loan Processed**.
  - Click Compose Email and configure the email template.
  - In the Email-Template tab, type the TO, CC, and the name of the recipient in Body with appropriate message. You can provide $ variables like $email1, $email2, and $FirstName respectively in the required fields.
  - In the **Parameters** tab, in the Name column, type the $variable given in email template, select the **Namespace** value as **None** and provide the required $variable data in the **Value** column.
- Drag and drop a Service Task next to the Loan Processed message task and click the Properties pane. You can do the following in the Properties pane:
  - In the **Name** field, type the required name of the service task based on the activity it handles. For example: **Disburse Loan Amount**.
  - Link this task to an Integration service and select **Loan** and **Disburse Loan** in the respective dropdowns as you have done in the previous service task.

- Configure the input parameter as **LoanId** as the output of the **Process Loan Request** was stored in Fabric_Workflow_Context.LoanId.



- Place an **End** node after the **Disburse Loan Amount** service task to finish the loan application flow path.

Once the Workflow is created, do the following to publish the Fabric app:

- Click the Publish tab.
- Select the required runtime environment.
- Click Publish.

23.8.3 Test the Workflow

After the app is successfully published, you can check the workflow functionality through the following means:

- Runtime server (Admin console)
- Postman

23.8.3.1 Runtime Server (Admin Console)

Click the Workflow icon in the Consoles section of the environment where the Fabric app is published. The Workflow Services screen from Admin Console appears in a new tab.

The Workflow Services screen lists all the workflows that were created and published to the current environment. To test the workflow which you have created (LoanApplication), do the following:

1. From the side menu, click Object Services.
2. Locate the Object service (ObjSvcLoanApplication) that is linked to the LoanApplication workflow. From the App Data Model Objects column select the operation (objLoanApplication) of the linked Object service. The Request Input screen of the selected operation appears.
3. From the Operations list, select create (POST).
4. Provide the input request body here. The request input will contain fields like the following;

```
{
  "WorkflowField": "",
  "CustomPrimaryField": "",
  "CreatedBy": "",
  "LastUpdatedBy": ""
}
```

5. Provide the value for "WorkflowField" parameter as **SUBMITTED** as it should be same as the value you provided for **Valid state transitions** field in **Submit Loan Request** user task.
6. Click **Get Response**. The response will have the following parameters:

```
{
    "CustomPrimaryField":1,"opstatus":0,"httpStatusCode":0
}
```

> ***Important:*** Use the same "CustomPrimaryField" parameter value received from response while providing input response for other user tasks while testing the workflow.

7. Now you can check the status of the LoanApplication workflow from Workflow service tab. The Workflow status will be Paused at **Check Loan Eligibility** user task.
8. Repeat the steps followed from step 2 to step 4.
9. Provide the value for "WorkflowField" parameter as **ELIGIBLE** as it should be same as the value you provided for **Valid state transitions** field in **Check Loan Eligibility** user task.
10. Provide a numeric value for "CustomPrimaryField" parameter as “1”.
11. Click **Get Response**.
12. Now you can check the status of the LoanApplication workflow from Workflow service tab. The Workflow status will be Completed.

> ***Note:*** In this testing process, it is assumed that all the service tasks, exclusive gateways and message tasks used were executed successfully.

> ***Note:*** To view the execution status of a workflow service by using Kony App Services Console, refer to Kony App Services Console > Workflow Services section.

23.8.3.2 Postman

Testing the Workflow through Postman involves two major steps. First you have to authenticate yourself as a valid user and then invoke the object service which would trigger the Workflow. Perform the following steps to test the workflow:

- Login to Postman and from the **Launchpad** tab, click **Create a Request**.
- In the **Enter Request URL** field provide the request URL. The request URL will be same as the Service URL you receive after publishing the Workflow.
  For example: https://XXXXXXX.auth. konycloud.com/login
- Click the Headers tab and provide the following details in KEY and VALUE columns respectively:

- Content-Type: application/json
- X-Kony-App-Key: 69f58xxxxxxxxxxxx9e35bdcdf2c9ca
- X-Kony-App-Secret: 92ee7xxxxxxxxxxxxxx2bdc8525c8cd

> ***Note:*** You can get the App Key and App Secret from the Publish page.

- Click **Send**. If you are an authentic user, you will receive a claims token.
  For example: `{"claims_token": {"value":"eyAidHlwIjogImp3dCITk9ORSIgfQ.xxxxxxx_kqjJUP1LNue_rA","exp":1578450870000,"integrity_check_required":false,"is_mfa_enabled":false,"mfa_meta":null,"session_id":"e6ecb1ed-606d-4a7c-b05d-442469ca8de8"}`
- From the verbs list, select POST.
- In the **Enter Request URL** field provide the request URL for the POST call. The format of the request is as follows:
  https://FabricAppName.konycloud.com:XXX/services/data/v1/ObjectServiceName/objects/ObjectName.
  For example: https://workflowtesting.konycloud.com:443/services/data/v1/ObjSvcDocTrial/objects/ObjDocTrial
- Click the **Headers** tab and provide the following details in KEY and VALUE columns respectively:
  - Content-Type: application/json
  - X-Kony-Authorization: <claims token>

  > ***Note:*** Provide the authorization key in the Value parameter in the Claims token that you received.

- Click the Body tab and provide the JSON payload here:

```
{
  "WorkflowField": "",
  "CustomPrimaryField": "",
  "CreatedBy": "",
  "LastUpdatedBy": ""
}
```

- Click Send.

24. Object Services

24.1 Introduction

Object Services is a feature of Kony Fabric that enables model-driven application design and development by following a microservices architectural approach to create reusable components and link them to fit into your solution. Using Object Services, you can define your preferred data model, which defines how your application wants to interact with its data. There is a clear separation between the data model and how it maps to the back-end systems of record. The defined data model and mappings encapsulate the back-end data and APIs, and abstract the complexity of the API from your client application.

You can use Object Services to create data models from line-of-business (LOB) objects and define service-driven objects from existing APIs in your enterprise. You can access LOB objects by using Kony Fabric business adapters. These business adapters enable you to visually discover and select the entities exposed by the LOB system. You can create a service-driven object from a set of existing Kony Fabric *Integration/Orchestration* Services. These Integration services connect to existing API endpoints or to Kony Fabric Orchestration Services, which combine multiple APIs into a new composite or aggregate API.

The Object Services feature enables you to control your API calls and data in a better manner.

- For simplistic applications, the use of integration services may be sufficient for you to get the job done.
- **As you start to build bigger and more complex applications, re-usability plays a vital role. In such scenarios, the use of Objects Services will help make your job easier.**

Object Services, as the name suggests, allows you to create and operate on an object for referencing your data. It helps you interact with your back-end data by using the Object data model you mentioned earlier. This feature makes it possible for you to develop applications in a faster and more intuitive manner.

> ***Note:*** For more hands-on approach on how to use Object Services, import and preview the Employee Directory app on to your Kony Visualizer.

24.2 How do Objects work?

Object Services makes application development easy by creating a data model to serve as a bridge between your back-end data and the client application.

The data model is the definition of your object. In the object-oriented world, an Object contains a set of Data fields and operations defined on that data. For example, if you want to build an application to display employee records and their contact information, you can create an “Employee” data model. This data model stores the attributes of an employee. You can then perform operations on the employee object and get a response in the format of the defined data model. For example, if you want to search for a particular employee by Employee ID, you can invoke a “getEmployeeById” method that is defined on your Object Service and get the response in the structure of the Employee data model.

Once your data model is defined, you can start to build your application around it. On the back-end side of things, you can map your data model to your data source. The strength of data models is that you can easily reuse your data model across applications, and easily switch to other data sources with a few changes in the front-end application.

> ***Note:*** Offline Objects feature helps you with a simplified approach to synchronize data to a client app for offline access. The Offline Objects feature connects directly with Object Services and does not require a sync run-time server. You can use the Offline Objects feature with apps that have been created by using Kony Visualizer along with apps developed in native iOS and Android by using Kony Fabric SDKs. For more information on Offline Enabled Object Services. refer Offline Enabled Object Services.

24.3 Why should I use Object Services?

A quick comparison of Objects Services with traditional Integration services is as follows:

Object Services	Traditional approach to APIs
Focus on designing the data model and mapping it to back-end data.	Focus on designing and invoking REST APIs.
Independent development of front end and back end.	Tightly coupled interactions between the front end and back end development.
Focus on application logic and enthralling user features.	Energy spent on creating back-end interactions.
Ease of service creation and invocation.	More code required, more prone to errors.
Reuse of same data model across different applications.	Does not encourage pattern-driven development.
Easy maintenance, code is generated automatically.	Tough to maintain varies with each developer's coding style.
Three times faster releases by reusing the same data model. Facilitates long-term savings!	No reuse – each application must invoke services from scratch.
Four times faster integration. Focus on configuring connectors to access your data.	Requires massive amount of custom code.

The process of application development is a volatile endeavor, comprising many variable factors. The two most important facets of your application consist of the front-end application and the back-end services. It is vital that these two facets communicate in the simplest manner possible, to pave your path to an awesome application.

Object Services tries to facilitate this communication by helping you create an interim bridge between these two facets. This is achieved by creating a data model, which acts as a contract between front-end and back-end developers.

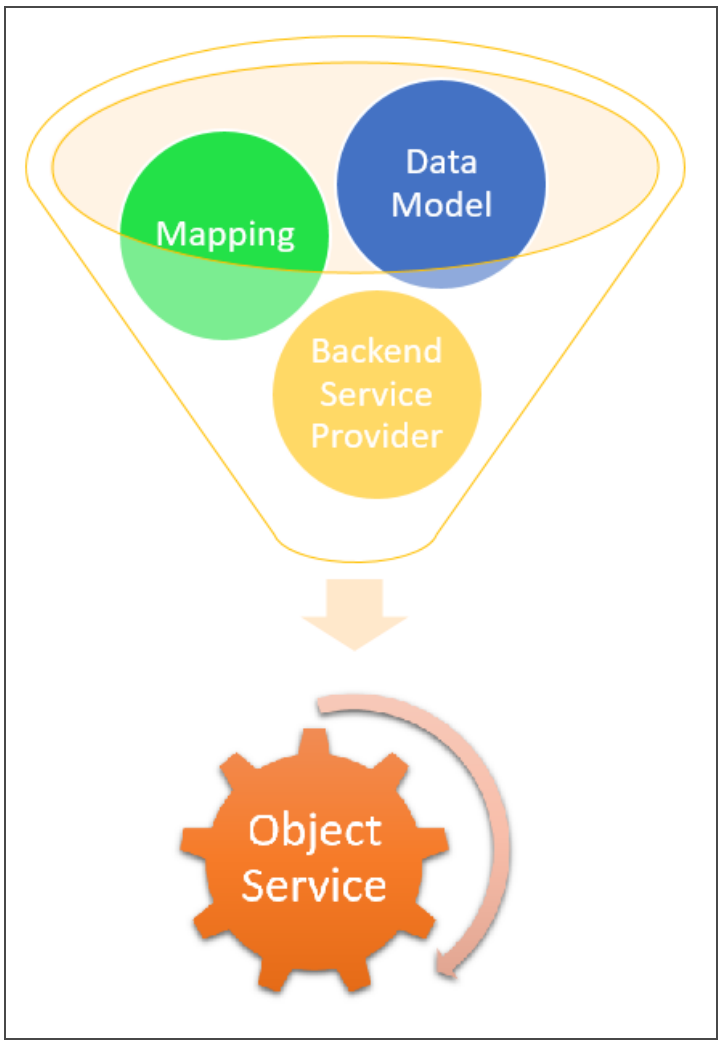
24.4 Object Services View

You can navigate to the Object Services View from the Objects tab in your application or from the API Management View. From there, you can create your services or use existing ones already added in your account.

To create an Object Service, you must understand the following basic components:

1. Data Model
2. Back-end Service Provider
3. Mapper

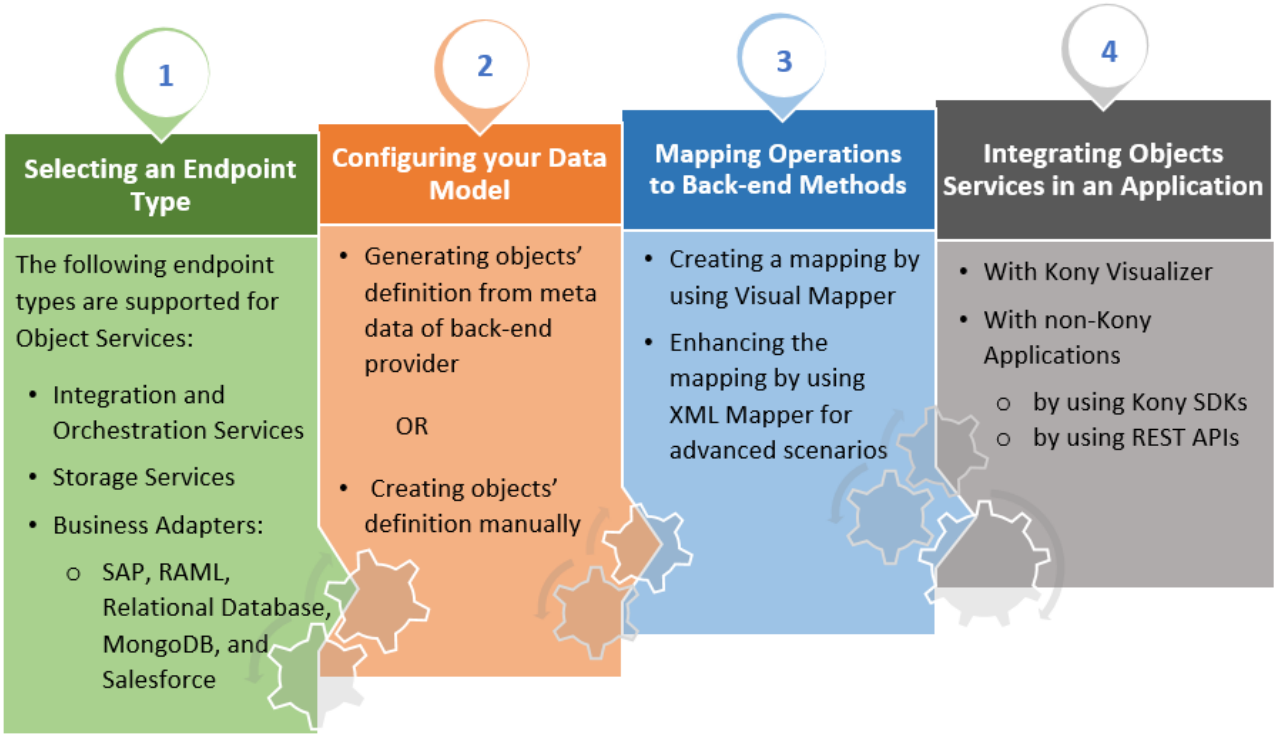
- Data Model is the core component of an Object Service and development plan. It functions as a vital contract between a client application and the back-end development team. As the first step of your application design, you need to create a data model that works for both your client application and the back-end provider. Once this model is finalized, both the development teams can proceed with their respective development processes without disrupting the other team.
- Back-end Provider refers to the source of data for your application. Object Services allow you to connect to multiple back-end providers such as SAP, Salesforce, RDBMS and so on. You also have the option to create your own database schema using the storage connector.
- Mapper helps you map data model and back-end service provider. Visual Mapper provides an intuitive way to connect to do the mapping.



Note: To walk-through creating an Object service with Kony Fabric, take a look at our hands-on tutorial for Object Services Overview

24.5 Workflow of Object Services

The following workflow describes the various stages of Object services:



24.6 Selecting an Endpoint Type

To create an Object Service, you can specify the endpoint details in the Object Service definition.

To create an object service, follow these steps:

1. Create an app.
2. Click the **Objects** tab.
3. Click **CONFIGURE NEW**.
4. Under the **Name**, type the name of the object, for example, **EmployeeModelSchema**.
5. Under the **Endpoint Type** field, select the endpoint that you want to use, and enter the details for the selected endpoint type.

For example, you can create objects services for the following three endpoint types:

- **Integration and Orchestration Services**
- **Storage:** Storage service helps you to develop apps by creating object data model and uploading data to the storage database which will then be available to app developers on the runtime instance.
    - When you select Storage endpoint, the **Sample Data** and **Download Template** buttons are displayed. You can store or import sample data by importing a .ZIP file.

      > ***Note:*** You can associate the sample data **at the design time of the service in Kony Fabric**, and you can import the sample data **at the run time in the App Services Console** as well.

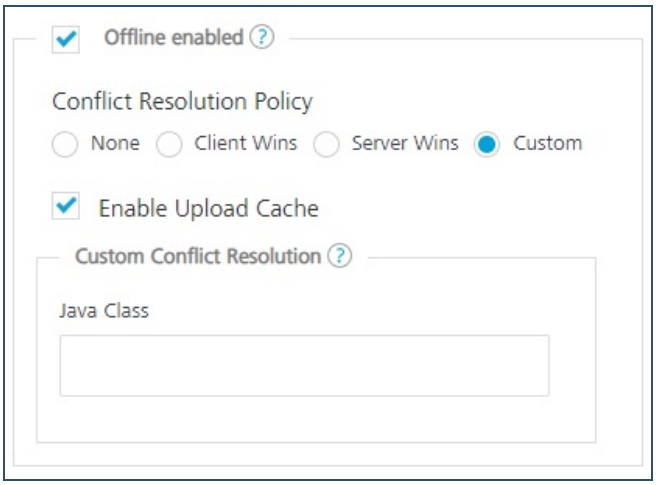
      For more information on generating sample data for Storage Object services, refer to How to Create Storage Object Services with Sample Data using Kony Fabric.
- **Business Adapters**: SAP, RAML, Relational Database, Salesforce, and MongoDB are the endpoint types for which you can create object services.

  > ***Note:*** If your database (RDBMS/MongoDB) is configured with a proxy server, you must select an **environment** and then click **Test Connection** to test the database connectivity. The environment should be `=> V8 SP3 or later`. If the entered details are correct, the system displays the message: Valid Database connection detail.

6. Set the **Security Level** field to secure the download of the object service metadata. Select any one of the following security levels to set the authentication type. By default, the field is set to Authenticated App User:
    - **Authenticated App User** - Restricts the download of object service metadata to users that have successfully authenticated using an Identity Service.

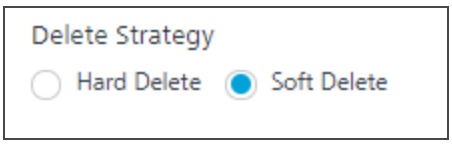
- **Anonymous App User** - Allows access from a trusted client that has the required App Key and App Secret, but the client does not have to authenticate the user through an Identity Service.
- **Public** - Allows any client to download the object service metadata without requiring any authentication.

7. In the **Offline enabled** section, you can configure the following options for objects services. All fields in the **Offline enabled** section are optional:



- Select the **Offline enabled** check box. You can limit the number of times the API can be invoked within a minute. If an API exceeds the throttling limit, the API will throw an error.
  - None
  - Client Wins
  - Server Wins
  - Custom
  - Select the **Enable Upload Cache**
    - Custom Config Resolution: Java Class

8. From Kony Fabric V9 onwards, the **Delete Strategy** section is displayed. This section is displayed when you have selected the **Endpoint Type** as **Storage**. It helps you delete the selected record data from the database. You can choose **Hard Delete** or **Soft Delete**.

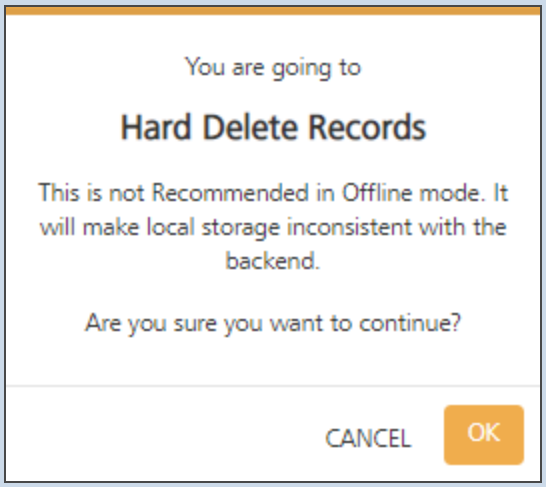


   - The **Soft Delete** option is selected by default. In this case, when a user performs the delete operation using the client app, the deleted record data is marked for the **SoftDeleteFlag** with value as `1` in the database. The soft deleted data is not available for the app user.

     > ***Important:*** Kony recommends you to use the **Soft Delete** option along with the **Offline enabled** mode to ensure the data is in sync with the device and the server.
     >
     > When you select the **Offline enabled** check box, the **Soft Delete** option is selected automatically.

   - Select the **Hard Delete** option to delete the selected record data permanently from the database.

Important: When you select the **Hard Delete** option after selecting the **Offline enabled** check box, the following waring message appears: *Warning! Hard Deleting Records in Offline mode is not Recommended. It will make local storage inconsistent with the backend.*

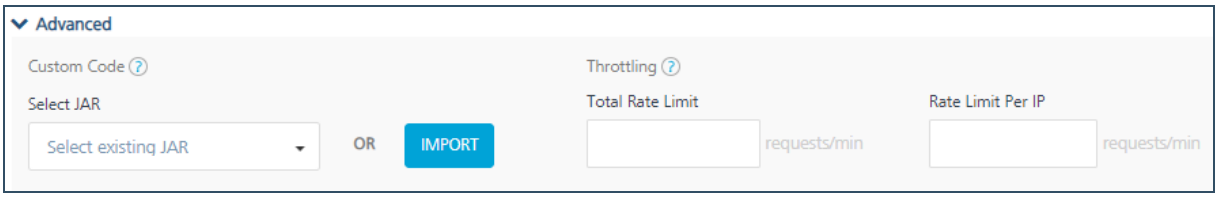


Click **OK** to continue enabling the **Hard Delete** option.

Note: You can also use the **Hard Delete** option from the App Services Console. For example, an app is created with the **Delete strategy** settings set to the default **Soft Delete** option but you want to delete some records permanently. In this case, you use the **Hard Delete** option by passing the `X-Kony-Soft-Delete` header parameter set to `false` with the delete request call. The delete request body must contain the primary key of the records that you want to delete.

For more information on how to use the Hard Delete option using the App Service Console, refer to **App Services Console > Object Services > Hard Delete option using the X-Kony-Soft-Delete header parameter for Storage Services**.

9. In the **Advanced >** you can configure the following options for objects services. All options in the **Advanced** section are optional:



- **Custom Code**

  Preprocessor and Postprocessor enable you to include any business logic on the data while transferring between external data source and the mobile device. Preprocessor is executed after service validation and authentication. Postprocessor is executed after the run time processing. Preprocessor and Postprocessor give you more control over request and response.

  To invoke preprocessor and postprocessor in an object service, you must specify a JAR file to associate with this service. Select from the **Select existing JAR** list, or click **IMPORT** to add a new JAR file.

- **Throttling:** You can limit the number of times an API can be invoked within a minute. If an API exceeds the throttling limit, the API will throw an error.

  - In the **Total Rate Limit** text box, enter a required value. This will limit the total number of requests processed by this API.
  - In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of requests made from an IP address. If a device exceeds the limit set, the API will return with an error message.

    To override throttling, refer to Override API Throttling Configuration.

    > ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1.
    > Refer to The Runtime Configuration tab on the Settings screen of App Services.

> The total limit set in the Kony Fabric Console will be divided by the number of configured nodes. For example, a throttling limit of 600 requests/minute with three nodes will be calculated to be 200 requests/minute per node.
> This is applicable for Kony Fabric Cloud and On-premises.

10. Click **Save & Configure**.

You can now Configure a Data Model.

24.7 Configuring a Data Model

You can design and optimize a data model based on the use cases of your mobile application. You can do this independent of the back-end system. Various data model entities, such as “Customer” and “Account” are grouped into reusable sets, such as “CRM” or “Work Order”. These sets encapsulate the related and dependent data model entities.

After you select an endpoint type, you can generate a data model from back-end LOB systems that already have their data model exposed as objects. Or, you can build a data model and **map the objects to a back-end system manually**. You can also create a service-driven object from an existing Service.

The following sections detail about Generating object's definition and object relationships:

- Generating Objects' Definition from Meta Data of Back-end Provider
- Creating Objects' Definition and Map to Back-end Objects Manually
- Configuring Relationships between Objects

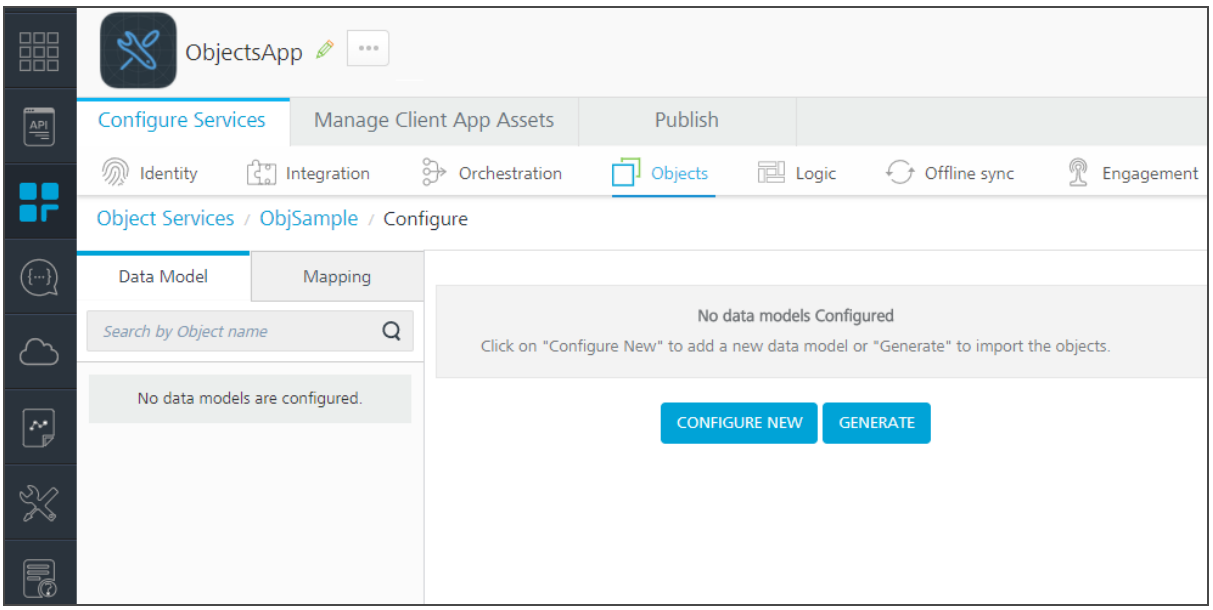
24.7.1 Generating Objects' Definition from Meta Data of Back-end Provider

You can generate a data model from a backend LOB system that has its data model exposed as objects. The example in the following section details the steps for generating a data model from a sample SAP backend.

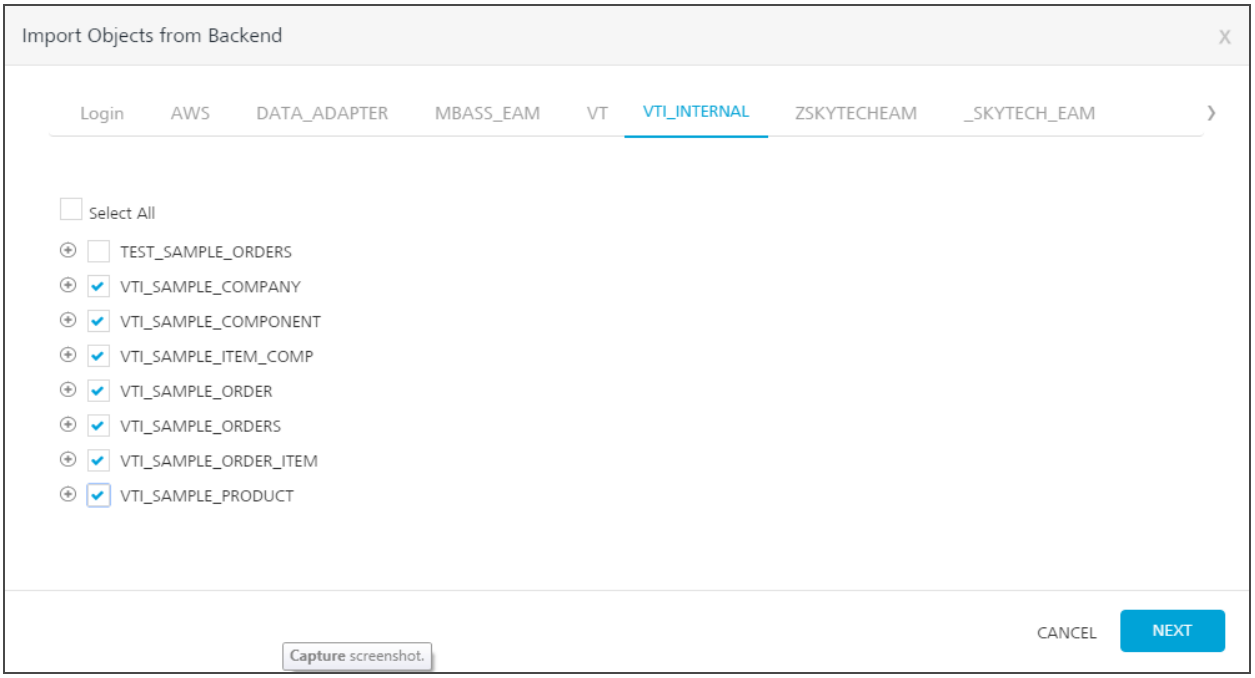
To generate a data model from back-end LOB objects, follow these steps:

1. You have Selected an Endpoint Type for your Object Service. Follow these steps to continue work with Objects Definition.

   The data model configure screen for the object service appears.



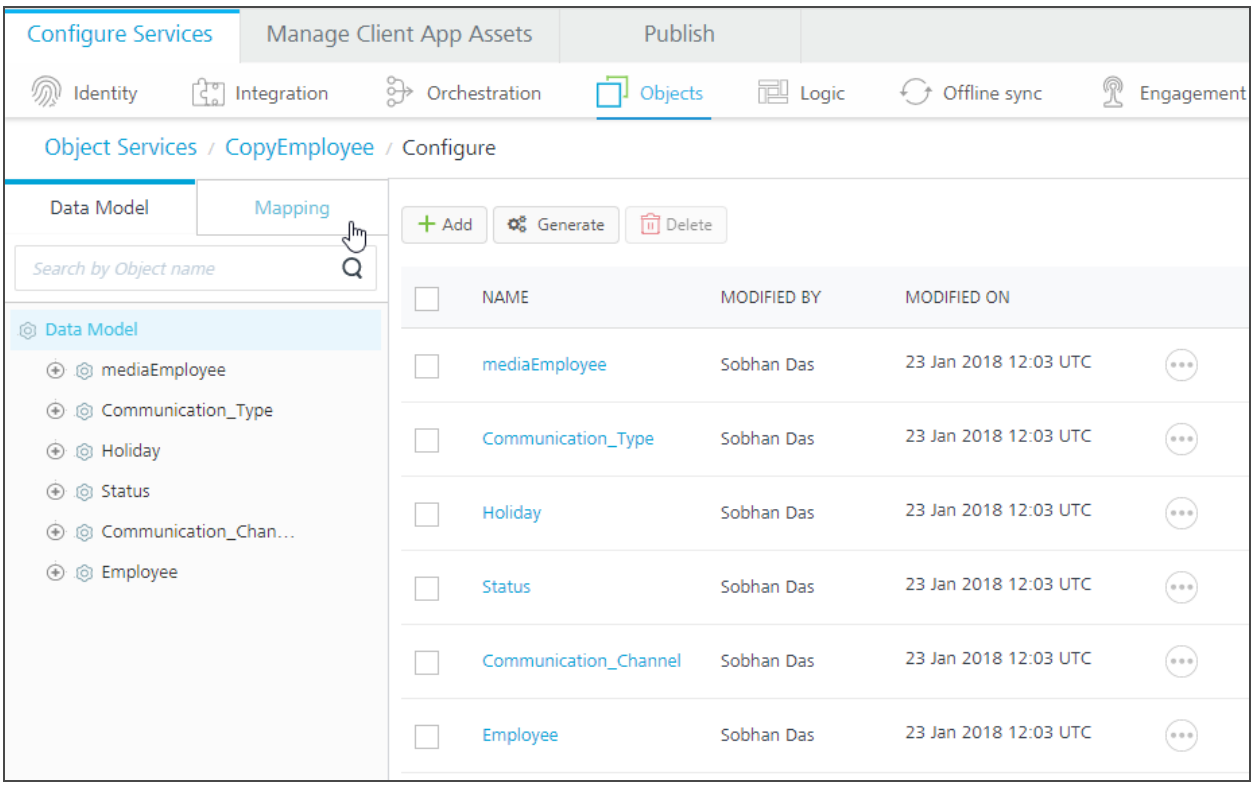
2. Click **Generate**.
3. From the **Import Objects from Backend** screen, click the **EmpDirDetails** schema to expand the tables details.
4. Under **Tables**, select the check boxes for the individual tables.



5. Click **NEXT**.

   The **Import Objects from Backend** screen shows the imported back-end objects and the names of the imported objects for the data model. You can edit the names of the objects for the data model.

6. Click **GENERATE**.

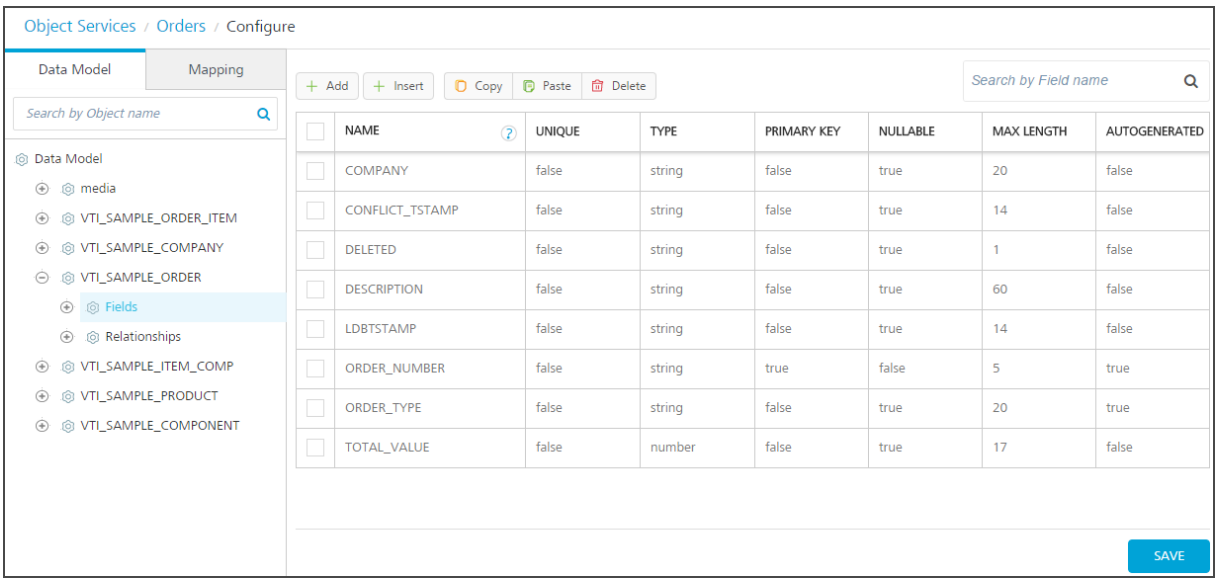


7. In the **Data Model** tab of the navigation pane, click the plus button next to the **contact** object.

   Fields and Relationships appear under the **contact** object.

8. Click **Fields**.

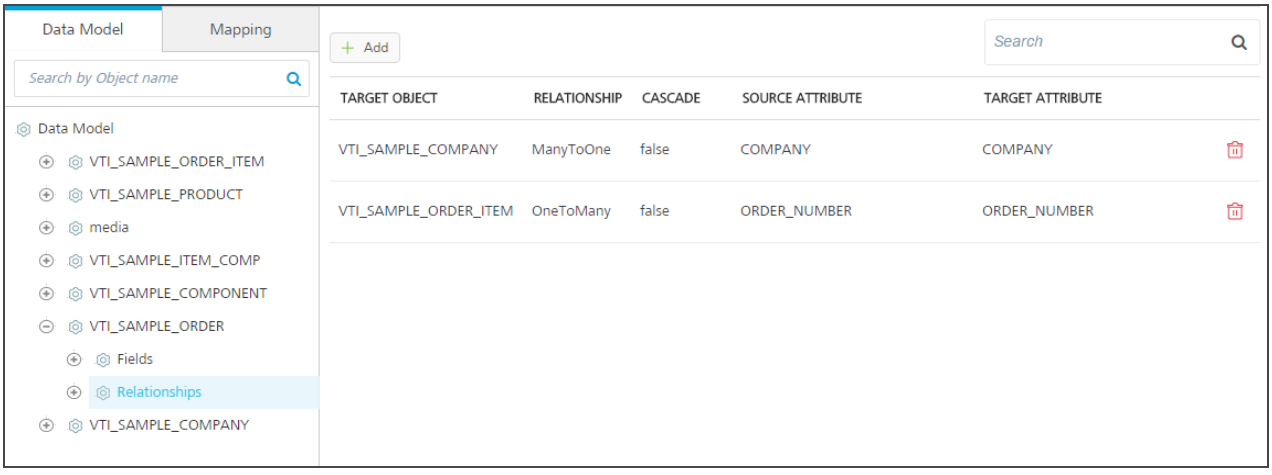
   The list of fields in the **contact** object appears in the Configure screen. You can change the name of the fields or modify the attributes. For example, you can change the primary key attribute. You cannot change the auto-generated attribute.



> ***Note:*** For metadata and Data Pre and Post Processors, refer to Automatic Field Level Encryption for Object Services.

9. In the **Data Model** tab of the navigation pane, under the **contact** object, click **Relationships**.

   The list of relationships for the **contact** object appears. The **contact** object has a many-to-one relationship with the VTI_SAMPLE_COMPANY object and a one-to-many relationship with the VTI_SAMPLE_ORDER_ITEM object. You can edit or delete the relationships, or add new relationships.



10. In the navigation pane, click the **Mapping** tab.

The Common Mapping configuration screen for the VTI_SAMPLE_ORDER object appears. The common mapping between a data model field and a back-end object field is applied to a transform request, response, or both, when methods are invoked on a back-end object. The double-headed arrow icon in the Type drop-down indicates that the mapping transformation is applied to both request and response. The right-arrow icon indicates only request mapping, and the left-arrow icon indicates only response mapping of the object.

11. Click **Save**.

You can now publish the Kony Fabric application as it is, or you can also configure the objects to be enabled for offline synchronization.

For more details on how to enable synchronization for the application, click Sync scope mapped to an object service.

After you have generated the data model from the back-end objects that you imported, you can publish your app. You can then provide the code that Kony Fabric generates to a mobile application developer. The mobile application developer integrates the code with platform SDKs and adds additional logic and modifies the presentation layer. The mobile application developer builds the client binary and publishes it to the enterprise app store.

> ***Note:*** To lock fields (read-only fields) in a data model of Object Services, refer to Configuring Read-only Fields for Object Services through MFCLI

At run time, Kony Fabric is the middleware that talks to the back end, manages the integration, and filters, transforms, and synchronizes the data it sends to the front-end clients.

24.7.2 Creating Objects' Definition and Map to Back-end Objects Manually

You can create a new data model manually and map it to your backend system. You must provide this data model to the application developer to integrate it with the app. The end user's device will use this data model to synchronize data with Kony Fabric. Mapping specifies how Kony Fabric will populate the data model based on the backend LOB objects. You can build the client application using your preferred data model and then map the application objects with the backend objects.

In this procedure, you will create a new data model: **EmployeeModelSchema**, add objects to the data model, and then add fields to the objects.

To create a data model and add an object, follow these steps:

1. **Steps to create an object:**
    a. Create a new object service. See Create an Object Service.
    b. Click **Save & Configure**.

       The Configure screen for the new object service appears. The Data Model and Mapping tabs appear. The Data Model tab is selected by default.

    c. Click **CONFIGURE NEW**.
    d. In the **Name** text box, enter **department**, and then click **SAVE**.

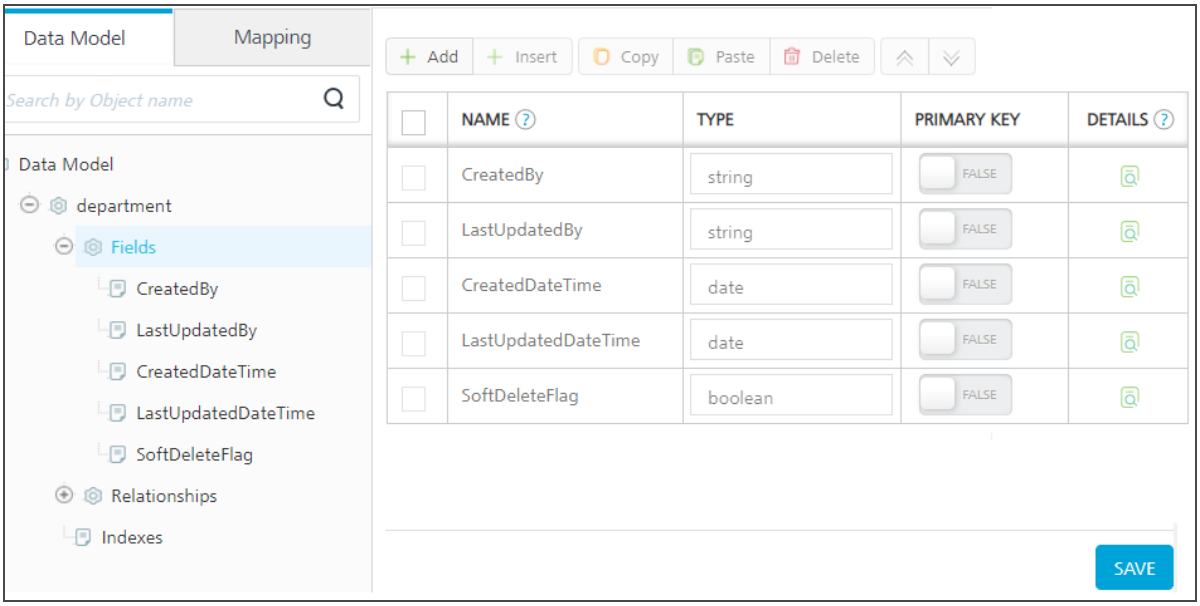
       In the Configure screen, the department object is created and appears in the list of objects.

2. **Steps to create fields:**
    a. In the **Data Model** tab of the navigation pane, click the plus button next to the **department** object.

       Fields and Relationships appear under the **department** object.

    b. Click **Fields**.

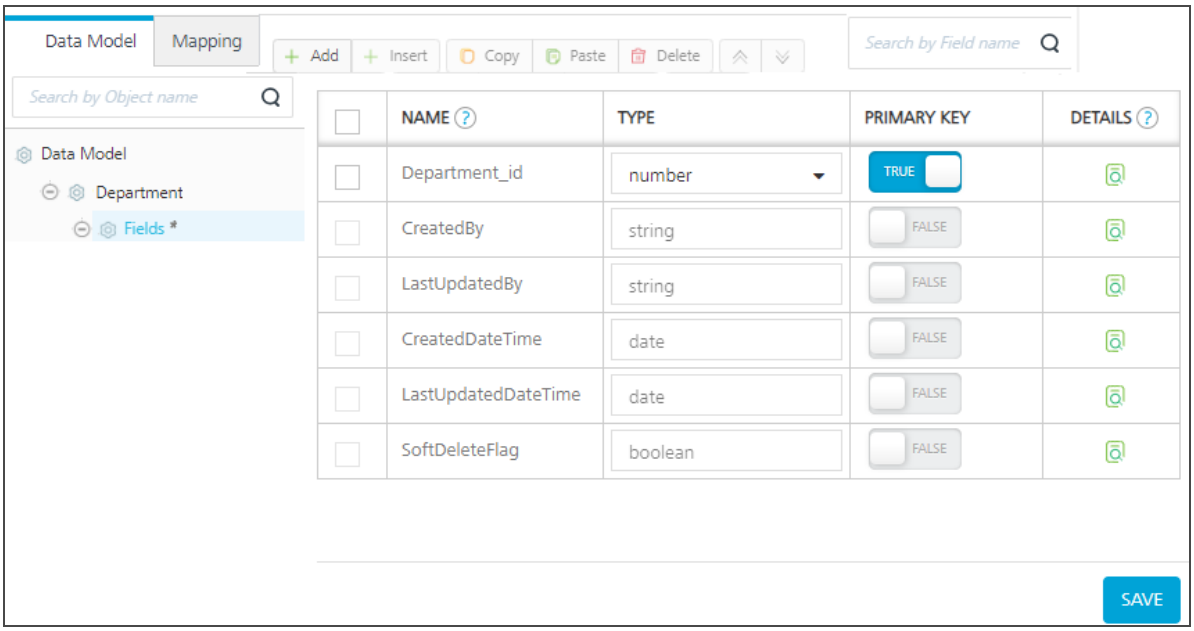
       The metadata details of the field is created by default and appears the same in the Field Configure screen.



c. Click the **Add** button.

A row is created for the new field including the basic columns *NAME*, *TYPE*, *PRIMARY KEY*, and *DETAILS*.

d. Under the **NAME** column, type the required name of the field, for example **Department_id**.



e. Under the **TYPE** column, select the data type of the field. The string data type is selected by default.
The available data types as follows:

Basic data types:	• `string` • `date` • `boolean` • `number`

Advanced data types	◦ `binary`: Used for uploading an image file format as evidence/proof. ◦ `enum`: Click here for more details on How to Configure Enum data type and examples. ◦ `workflow state`: Used for associating a back-end workflow. While defining a `workflow state` data type for a field, you can create a workflow or use an existing one. A workflow can contain a set of nodes represents a specific task or an event as per your business logic. Here, the workflow data type is a part of a field in an object. Backend workflow is linked to a verb in an Object Service and whenever that verb in the object service is invoked, the related workflow is triggers automatically. So that the workflow will execute all the subsequent tasks that were defined with in it and completes the entire backend process that is related to the linked verb. • Using Workflow, you can visualize and design a backend process. It justifies the low code concept and you can design the backend workflow simply by dragging and dropping different types of nodes and connecting them as per the logic you need. • For more information on how to configure a workflow, refer to Workflow

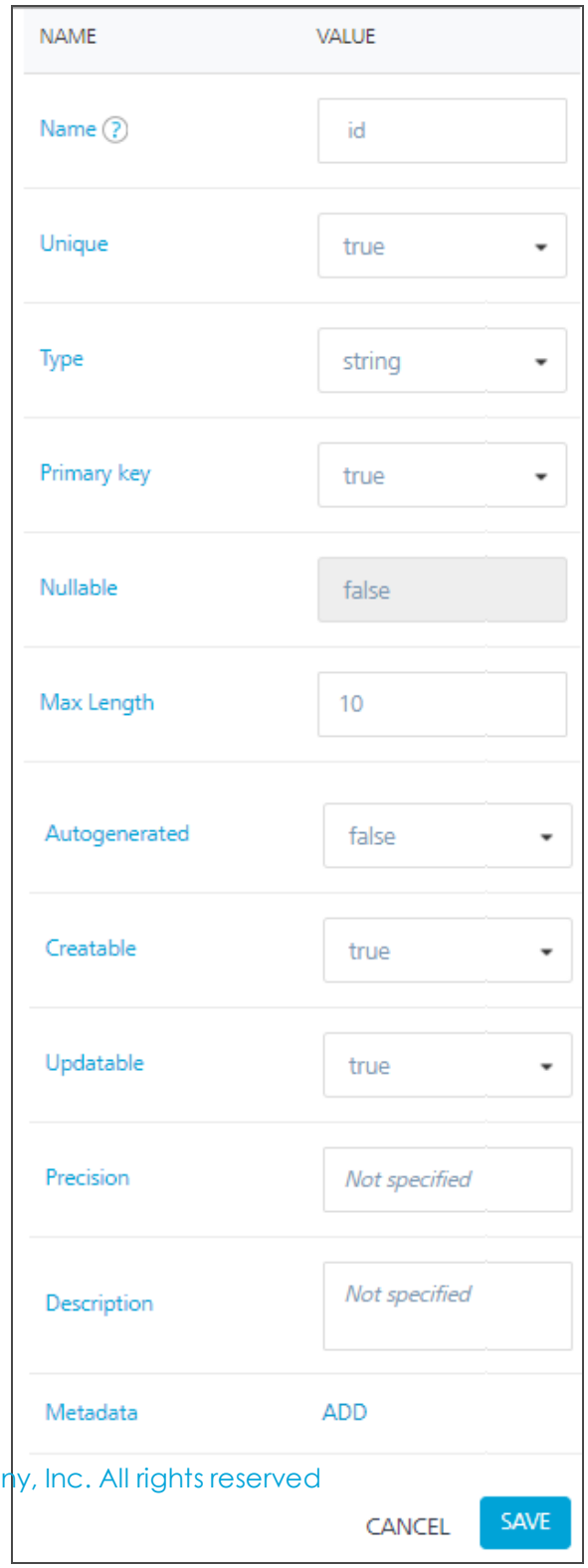
f. If you want to set the field as a primary key, under the **PRIMARY KEY** column, click the toggle button to set to **TRUE**. The primary key value is **FALSE** by default.

g. If you want to view and configure the additional columns of the field, click the **View Details** button under the **DETAILS** column.

> ***Important:*** The **View Details** button is active only after you save the field details.

The field details page displays the additional columns including *unique, nullable, max length, autogenerated, creatable, updatable, precision, description*, and *metadata*.

The following details to help you when to use some of the specific constraints to specify rules for a field:

Field constraints	Description
	**Unique**: ◦ The `unique` constraint ensures that all values in a column are different. ◦ Both the `unique` and `primary key` constraints provide a guarantee for uniqueness for a column or set of columns. ◦ A `primary key` constraint automatically has a `unique` constraint. ◦ However, you can have many `unique` constraints per table, but only one `primary key` constraint per table.

The following details to help you when to use some of the specific constraints to specify rules for a field:	
Field constraints	**Description**
	Primary key ◦ The `primary key` constraint uniquely identifies each record in a table. ◦ `Primary keys` must contain `unique` values, and cannot contain NULL values. ◦ A table can have only ONE `primary key`; and in the table, this primary key can consist of single or multiple columns (fields).

The following details to help you when to use some of the specific constraints to specify rules for a field:	
Field constraints	**Description**
	Precision is the number of digits in a number. ◦ For example, the number 123 has a precision of 3. ◦ For example, the number 123.45 has a precision of 5 and a scale of 2. Scale is the number of digits to the right of the decimal point in a number.

The following details to help you when to use some of the specific constraints to specify rules for a field:	
Field constraints	**Description**
	Metadata: This metadata of the Object allows app developers to control client-side logic by using the **Data Pre and Post Processors for Models** functionality, which helps to access the metadata dynamically at runtime and has custom JS code written to transform the data based on the metadata. This allows app developers to modify business logic at the client app and control which fields get transformed and how. For example, you can write your custom logic for fields in models to transform data in client before sending to back end and vice versa such as data encryption, decryption, currency or temperature conversion, prepend or append characters such as salutations (Mr. or Dr.) or currency symbol ($ or €) and so on. For more information on metadata, Click here for more information on Object Metadata for Controlling Client-side Logic.

h. If you want to add more fields to the object, repeat step 2 in the procedure.

i. Click **SAVE** to save the details of the field.

> ***Note:*** If you want to create more objects in the data model, select the **Data Model** root node and click **Add.** And then repeat steps 1 and 2.

> ***Note:*** To lock fields (read-only fields) in a data model of Object Services, refer to Configuring Read-only Fields for Object Services through MFCLI

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

> ***Note:*** You can also clone a data model of an existing object service. Refer to Actions in objects services.

24.7.3 Configuring Relationships between Objects

The following procedure helps you how to define relationships between the unique fields in objects in the **EmpolyeeModelSchema** data model.

To configure the relationships, follow these steps:

1. In the **Data Model** tab of the navigation pane, under the **department** object, click **Relationships**.

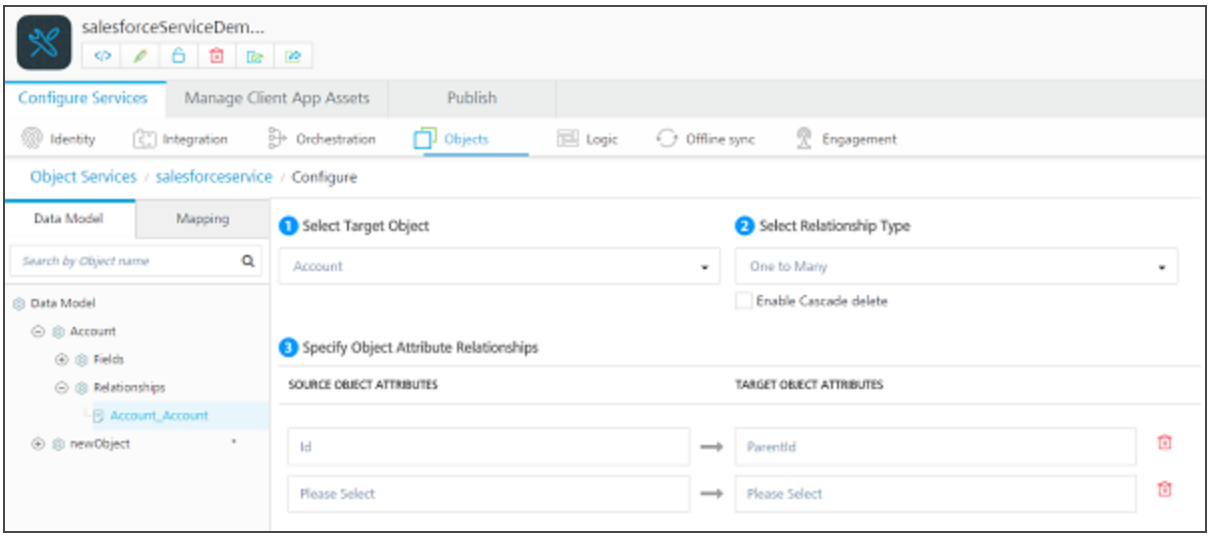
   The Relationships screen appears.

2. Click the **Add** button.

   The screen for configuring the relationships appears.

3. Click in the **Select Target Objects** field, and then select the **employee** object from the drop-down menu.

4. Click in the **Select Relationship Type** field, and select **One to Many** from the drop-down menu.
5. Under **Specify Object Attribute Relationships**, click in the **Source Object Attributes** field, and then select **Department_id** from the drop-down menu.
6. Click in the **Target Object Attributes** field, and then select **Department_id** from the drop-down menu.



7. Click **SAVE**.

> ***Note:*** You can add objects to the data model, configure relationships, and enable synchronization for the application.

> ***Note:*** You can sort fields by clicking the column name header in the data model.

You can now configure Common Mapping and Methods to the Fields on the Back-End, refer to Mapping Operations to Back-end Methods.

24.7.4 Enumeration Data Type

A field with the `Enumerated/enum` data type enables you to configure a set of values/options for that

field. So the enum data type field restricts app users to enter/select only the pre-configured list of options as the input for that field.

Use cases:

- In the **Contact** form, the **State** field needs to be a pre-configured list of 50 States of America. This enables the app users to choose only one of the available values from the list.
- A **Priority** field can have enumeration options as *Critical*, *High*, *Medium*, and *Low*.
- A **Gender** field can have only *Male* and *Female* as valid values.

How to configure enum data type for a field:

1. In Kony Fabric Console, create an object in the Data Model of an object service, for example **Address**.
2. Select the **Fields** node under the **Address** object and click **Add** to create a field.
3. Under the **NAME** column, specify the field name, for example, **State**.
4. Under the **TYPE** column, select the data type as `enum`. The string data type is selected by default.

   The **Enumeration Details** dialog box appears.

a. Under the **Enum Option** text box, enter the details and then type comma or space bar on your keyboard. These values are updated in the text box. Refer to Enum data type samples, in the next section.

> ***Note:*** You can also copy and paste the comma-separated text into the text box.

> ***Note:*** The number of enumeration options allowed is up to a maximum of 50.

b. Click **SAVE** to save the details and close the dialog box.
Use the **Edit** button shown next to the enum data type to modify the data, if required.

5. Click **SAVE** to save to the field.

24.7.4.1 Enum data type samples

- You can configure the enumeration options as a comma-separated set of values, as shown:

Example	Output
Critical, High, Medium, Low	Critical High Medium Low

- You can configure the enumeration options in Text with special characters within quotes (single/double) and comma-separated, as shown:

Option	Input for the Enum field in Console - Examples	Output in the form of an app
option 1	"Critical", "High", "Medium", "Low"	"Critical' "High" "Medium" "Low"
option 2	"Orlando FL", "Edison NJ", "Austin TX", "Cupertino CA"	"Orlando FL" "Edison NJ" "Austin TX" "Cupertino CA"

Option	Input for the Enum field in Console - Examples	Output in the form of an app
option 3	"Orlando, FL", "Edison, NJ", "Austin, TX", "Cupertino, CA	"Orlando, FL" "Edison, NJ" "Austin, TX" "Cupertino, CA"
option 4	'Orlando FL', 'Edison NJ', 'Austin TX', 'Cupertino CA'	'Orlando FL' 'Edison NJ' 'Austin TX' 'Cupertino CA'
option 5	'Orlando, FL', 'Edison, NJ', 'Austin, TX', 'Cupertino, CA'	'Orlando, FL' 'Edison, NJ' 'Austin, TX' 'Cupertino, CA'

24.8 Mapping Operations to Back-end Methods

The preferred data model is rarely in the same format or representation as the back-end data from the systems of record. The condition requires a robust mapping capability that can easily retrieve, filter, and transform the back end into the preferred representation. For example, the preferred view of the customer entity may comprise of a contact object from SAP and additional profile information from the enterprise service bus (ESB).

The following sections detail about common mapping, methods (verbs) mapping, custom verbs, and mapper elements:

- Mapping Verbs and Methods to the Fields on the Back End
  - Configuring **Common Mapping** to the Fields on the Back End
  - Configuring **Methods Mapping** to Fields on the Back End
  - Adding **Custom Verbs** for SAP Object Service
  - Mapping **verbs for Objects (for Service-driven Objects)**
  - Testing an Object Service in Kony Fabric
- Creating a Mapping by using Visual Mapper
- Enhancing the Mapping by using XML Mapper for advanced scenarios
- Mapper Elements
- Identity Support in Mapper

24.8.1 Mapping Verbs and Methods to the Fields on the Back End

24.8.1.1 Configuring Common Mapping to the Fields on the Back End

The common mapping that you define between a data model field and a back-end object field is applied to a transform request, response, or both. Kony Fabric applies this mapping when it invokes any method on the back-end system.

The common mapping contains three types of mapping icons:

- A **double-headed arrow** icon in the indicates both request and response mapping.
- The **right arrow** icon indicates only request mapping.
- The **left arrow** icon indicates only response mapping.

To configure the common mapping, follow these steps:

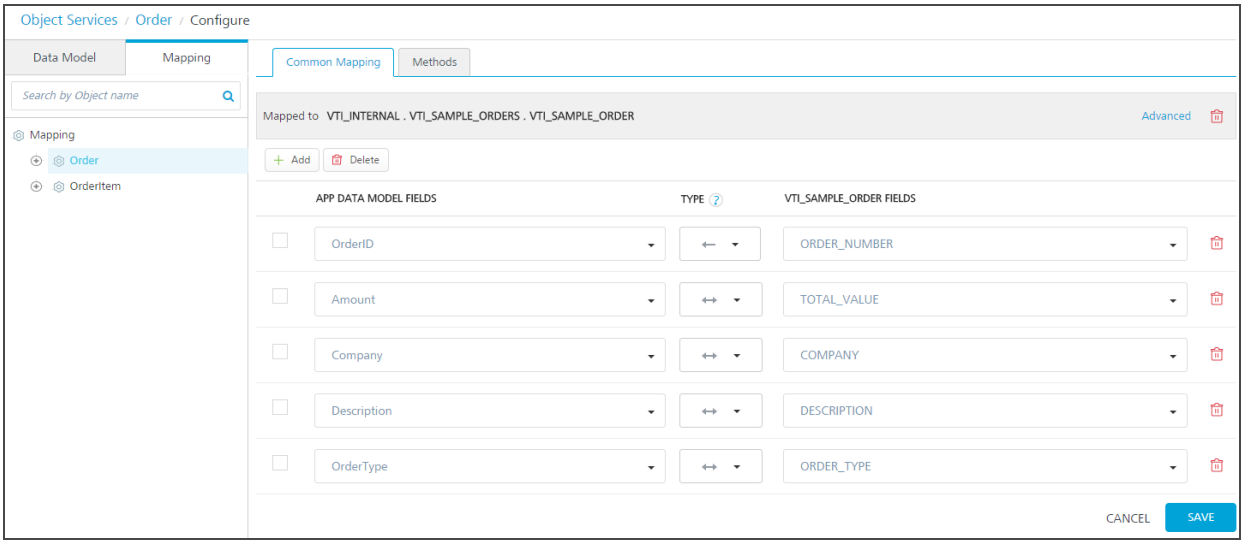
1. In the navigation pane, click the **Mapping** tab.

   The mapping configuration screen for the Order object appears.

2. Click in the first **Map to** field. A drop-down menu appears. Select **VTI_Internal**. The business object in the back end.
3. Click in the second **Map to** field. A drop-down menu appears. Click on the plus button next to **VTI_SAMPLE_ORDERS**, and then click on **VTI_SAMPLE_ORDER**.

   The Common Mapping tab shows that the Order object in your data model is mapped to the VTI_SAMPLE_ORDER table of the VTI_SAMPLE_ORDERS database in the back-end object VTI_Internal.
4. Click the **Add** button.
5. Click in the first field under **APP DATA MODEL FIELDS**. A drop-down menu appears. Select **OrderID**.
6. Click in the first field under **VTI_SAMPLE_ORDERS FIELDS**. A drop-down menu appears. Select **ORDER_NUMBER.**
7. Click the **Add** button.
8. Click in the second field under **APP DATA MODEL FIELDS**. A drop-down menu appears. Select **Amount.**
9. Click in the second field under **VTI_SAMPLE_ORDERS FIELDS**. A drop-down menu appears. Select **TOTAL_VALUE**.
10. Click the **Add** button.
11. Click in the third field under **APP DATA MODEL FIELDS**. A drop-down menu appears. Select **Company**.
12. Click in the third field under **VTI_SAMPLE_ORDERS FIELDS**. A drop-down menu appears. Select **COMPANY**.
13. Click the **Add** button.
14. Click in the fourth field under **APP DATA MODEL FIELDS**. A drop-down menu appears. Select **Description**.

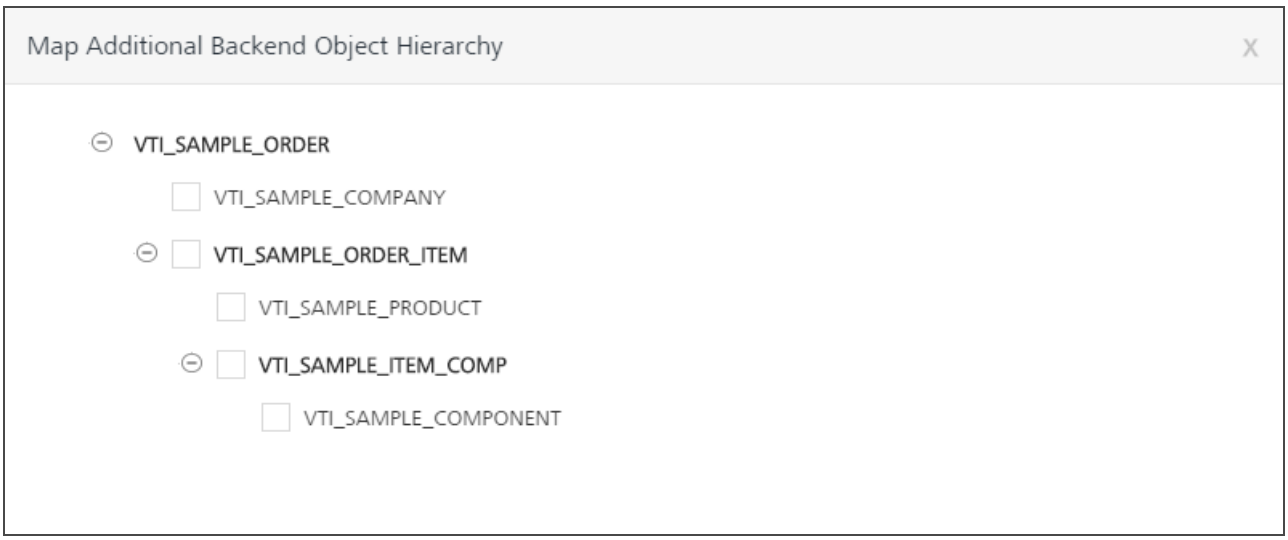
15. Click in the fourth field under **VTI_SAMPLE_ORDERS FIELDS**. A drop-down menu appears. Select **DESCRIPTION**.
16. Click the **Add** button.
17. Click in the fifth field under **APP DATA MODEL FIELDS**. A drop-down menu appears. Select **OrderType**.
18. Click in the fifth field under **VTI_SAMPLE_ORDERS FIELDS**. A drop-down menu appears. Select **ORDER_TYPE.**



19. Under **TYPE**, click in the drop-down menu for OrderID and ORDER_NUMBER. Select the left-arrow to indicate response mapping of the object field.

> ***Note:*** You can sort fields by clicking the column name header in the Mapping tab.

You can also map an object in your data model to additional child objects in the backend object hierarchy. To map an object to additional child objects in the backend, click **Advanced**, and then select the child objects in the hierarchy.



24.8.1.2 Configuring Methods Mapping to Fields on the Back End

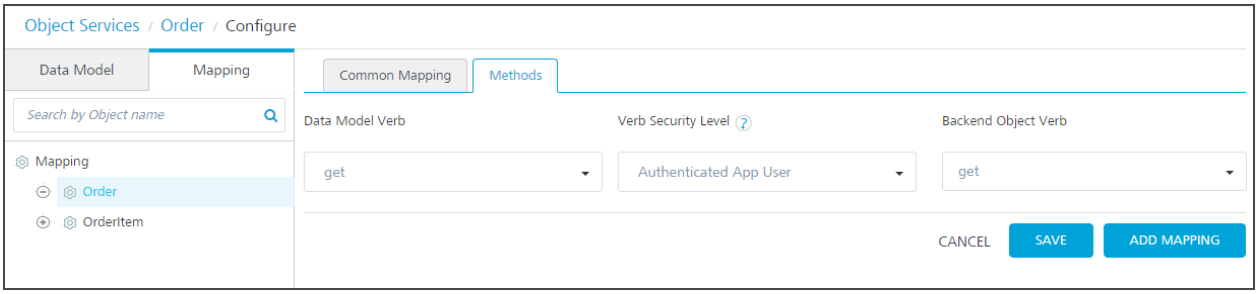
Methods mapping is where you can configure what the system should do for all the methods the client app will use. For example, a work order will use the Get, Post (create), Put (update), and Delete methods. When you perform Get on a work order object, the mapping specifies the target object in the SAP back end and the method that you can use against that object. You can control the mapping for both the request to SAP and the response.

To configure the methods mapping, follow these steps:

1. In the mapping screen, click the **Methods** tab.
2. Click the **Add** button.
3. Click in the **Data Model Verb** field, and then select **Get** from the drop-down menu.
4. Use the default **Verb Security Level**, Authenticated App User.

   The Verb Security Level specifies how the client must authenticate to the verb. You can restrict access to this verb to only authenticated app users that have successfully authenticated using an Identity service. An anonymous app user verb allows access from a trusted client that has the required App Key and App Secret, but the client does not need to authenticate the user through an identity service. Set the security level to Public to allow any client to access this verb without any authentication requirement.

5. Click in the **Backend Object Verb** field. Click the plus button next to VTI_SAMPLE_ORDER, and then select **Get**.



> ***Note:*** For Integration services, Object Services provides built-in variants of the get verb. Click the get verb in the navigation page to configure the mapping of the variants of the get verb. The built-in variants of the get verb are getAll, getbypk, getupdated, getbatch, and getdeleted. These get verbs do not have an individual mapping. They have a common request mapping and a common response mapping.

6. Click **Add Mapping**.

   The method mapping configure screen appears.

   The Security Filters, OData Query Options, Request Mapping, Response Mapping, and Test tabs are displayed.

   The **OData Query Parameters** tab lists the parameters of the method that you can use for the VTI_SAMPLE_ORDER object.

7. Click the **Request Mapping** tab, and then click **Edit** button under Mapper area.

   The common mapping of the object is applied to the verb by default.

8. To override the common mapping, click **Clear Mappings**, and specify a custom mapping in Request Mapping and Response Mapping.

> ***Note:*** If you want to override a custom mapping with common mapping, click **Apply Common Mapping**.

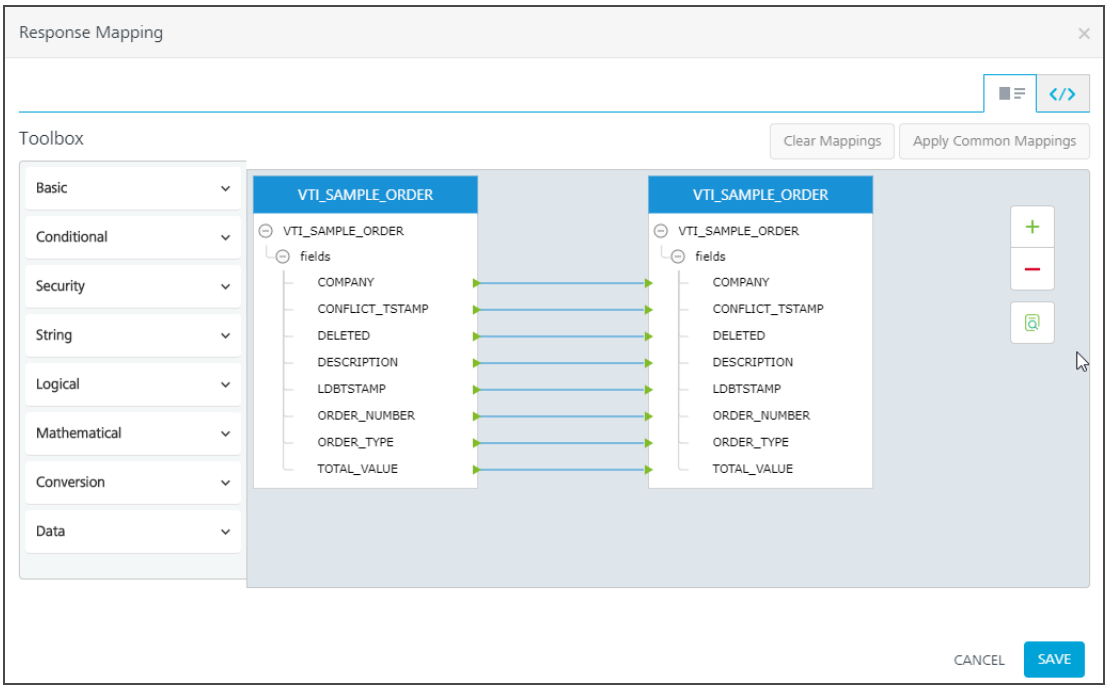
9. In OData Query Options, do the following:
    a. Click in the **Value** field for the **$filter** parameter.

    > ***Note:*** The `&` and `=` operators are not supported in values for OData string key (`$filter`).

    b. Enter the expression **orderStatus ne 'Draft'**. This will filter the orders that have an order status of Draft.

10. Click **Request Mapping**.

    If you do not apply common mapping of the object to the get verb, this is where you specify a custom mapping on the request to SAP. Common mapping by default maps one-to-one the data model method to the method of the target object in the back end.

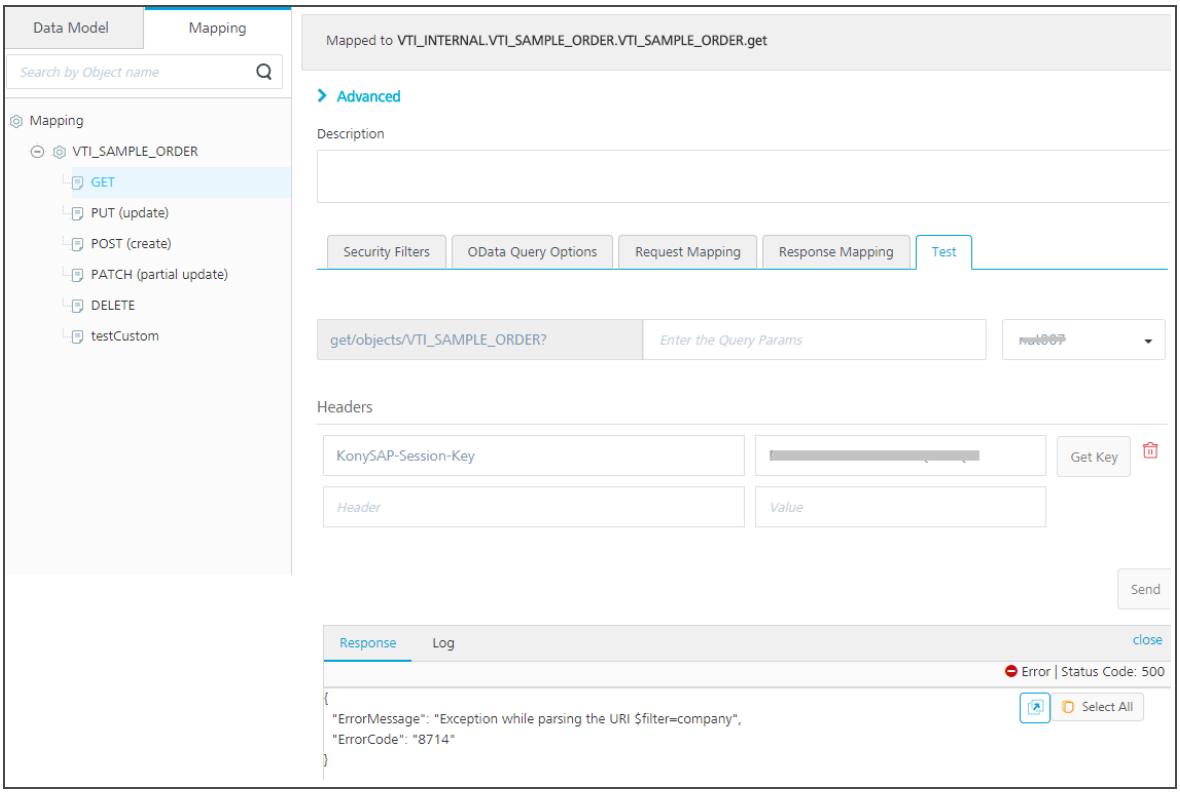


11. Click **Response Mapping**.

If you do not apply common mapping of the object to the get verb, this is where you specify a custom mapping on the response to SAP.

12. Click **Test**.

    The Test panel appears. You can use the Test panel to test the mapping for a method. To test the mapping, enter the query parameters, select an environment, enter the headers and header values that you want to include with the test. Then, in Request Payload, enter values for the fields for which you want to test the mapping, and then click **Send**.

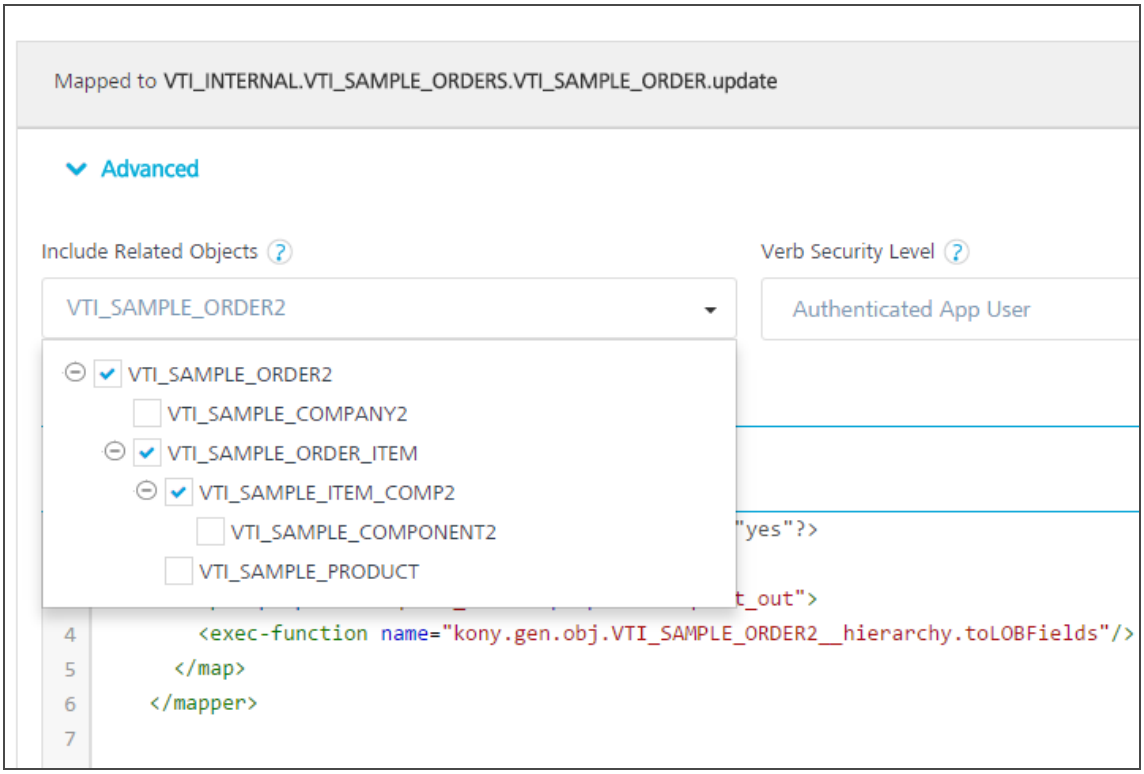


13. In the navigation pane, click the **Orders** object.

14. For the **Get** verb, click **Advanced**.

    - The **Include Related Objects** field and the **Verb Security Level** field appear. The Include Related Objects setting specifies which part of the data model Object's hierarchy can be handled by the verb. This information helps in optimizing the number of calls to the

back end in case the verb also deals with other objects in the hierarchy. You can multi-select multiple objects in the drop-down to specify the information.



- **Custom Code Invocation** section appears. Configure the parameters for the preprocessor and postprocessor to filter the request and response objects for your business requirements. You can specify multiple preprocessors and postprocessors. This is supported for integration/Orchestration services and Object services.

> ***Important:*** To invoke preprocessor and postprocessor, you must **import a valid JAR file** while creating an object service. The corresponding jars have to be uploaded to the Kony Fabric console under the **Advanced tab in the object service definition** service definition.

- **Java Preprocessor and Postprocessor** - The preprocessor and postprocessor are Java classes that implement **ObjectServicePreProcessor** and **ObjectServicePostProcessor**

interfaces. A developer can write custom code in the **execute** method of the preprocessor or postprocessor class.

- Under the **Custom Code Invocation**, follow these steps:
    i. Under **Preprocessor**, configure the following:
        - For **Java**, you can configure multiple Preprocessors. This is supported for Integration/Orchestration services and Object services. If you have defined your logic for multiple preprocessors in the uploaded JAR file in the service definition, you can then select the available one or other preprocessors. You can arrange the selected pre-processors to be executed in the desired order during the operation call.

          Use Case

          When customers have a large amount of custom code, the maintainability of the code becomes an issue. The issue becomes much more complicated when multiple stakeholders working on custom code. In such cases, the custom code can be split into multiple pre/post processors so that stakeholders can work on their respective modules. This increases the upgradability and maintainability of the custom code.

          Select **Java**, and from the **Class** list, select a preprocessor class. You can select one or more classes.
          This step enables a developer to include any business logic on the data before sending the response to a mobile device.

    ii. Under **Postprocessor**, configure the following:
        - For **Java**, you can configure multiple Postprocessors. This is supported for Integration/Orchestration services and Object services. If you have defined your logic for multiple postprocessors in the uploaded JAR file in the service definition, you can select the available one or other post-processors. You can arrange the selected post-processors to be executed in the desired order during the operation

call.

Use Case

When customers have a large amount of custom code, the maintainability of the code becomes an issue. The issue becomes much more complicated when multiple stakeholders working on custom code. In such cases, the custom code can be split into multiple pre/post processors so that stakeholders can work on their respective modules. This increases the upgradability and maintainability of the custom code.

Select **Java**, and from the **Class** list, select a postprocessor class. You can select one or more classes.
This step enables a developer to include any business logic on the data before sending the response to a mobile device.

You can rearrange the order of the classes to be executed, if required.

- ## Preprocessor and Postprocessor interfaces in Object services

ObjectServicePreProcessor

Implement this interface and write your custom preprocessor logic by overriding execute method. You are provided with *FabricRequestManager*, *FabricResponseManager* and *FabricRequestChain* as method arguments.

```
 public interface ObjectServicePreProcessor {
void execute(FabricRequestManager fabricRequestManager,
FabricResponseManager fabricResponseManager,
FabricRequestChain fabricRequestChain)
throws Exception;
}
```

Method arguments:

FabricRequestManager **:** This argument specifies the different types of handlers to operate on request related information.

FabricResponseManager **:** This argument specifies the different types of handlers to operate on response related information.

FabricRequestChain **:** This argument specifies a chain similar to servlet *FilterChain* used to control the request flow.

For a sample Java class code for ObjectServicePreProcessor, refer to Java Sample Code for Preprocessor - Object services.

ObjectServicePostProcessor

Implement this interface and write your custom postprocessor logic by overriding execute method. You are provided with *FabricRequestManager and FabricResponseManager* as method arguments.

```
 public interface ObjectServicePostProcessor {
void execute(FabricRequestManager fabricRequestManager,
FabricResponseManager fabricResponseManager)throws Exception;
}
```

Method arguments:

FabricRequestManager **:** This argument specifies the different types of handlers to operate on request related information.

FabricResponseManager **:** This argument specifies the different types of handlers to operate on response related information.

For a sample Java class code for ObjectServicePostProcessor, refer to Java Sample Code for Preprocessor - Object services.

> ***Important:*** **Limitations of custom code invocation in Object services are as follows:**
>
> - Custom code is not invoked when Bulk Sync V2 APIs are in use.
> - Custom code is not invoked when orchestration services which make use of Object services are in use.
> - Custom code is not invoked when we use Kony Fabric console test API.

> ***Note:*** **For Kony Cloud / For on-premises:** Runtime jars to write custom code can be downloaded from Admin console. In the **Downloads** section, you can download the middleware jars as a zip file. Here the `middleware-api.jar` will have all the classes related to custom code.

> ***Note:*** For details on middleware APIs for preprocessor and postprocessor, contact refer to http://docs.kony.com/konylibrary/integration/MiddlewareAPI/index.html

15. Click **Save**.

You can enable the application for offline synchronization, publish the application, and then provide the code that Kony Fabric generates to a mobile application developer.

24.8.1.3 How to Use Actions in Existing Objects Methods

You can perform the following actions on a method in existing object services:

- **Edit**: Allows you to edit a method. After you edit a method, you have to republish all the apps that are using the service to apply the changes.

  > ***Note:*** To know more about publishing an app, refer to Publish an app.

> ***Note:*** If a service is part of a published app, you can rename that service only after the app is unpublished.

- **Sample Code**: A dynamic code is generated based on the configuration of a method. You can use this code in your SDK.
- **Delete**: Allows you to delete a method.

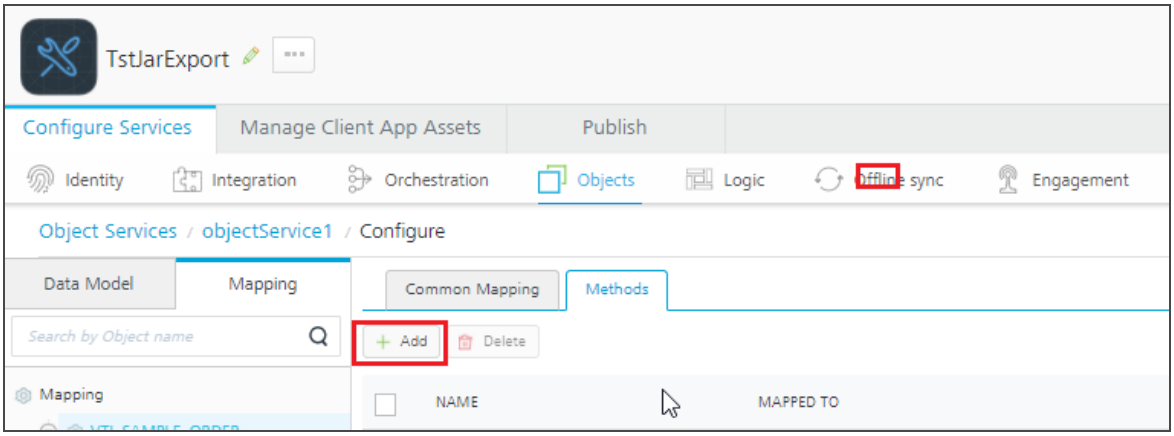
> ***Note:*** If a service is a part of a published app, you can delete that service only after you unlink the service from all the published app.

24.8.1.4 Adding Custom Verbs for SAP Object Service

SAP Object Service provides you the ability to add custom verb names and map it to an operation of a back-end object. You can now create your own custom verb in addition to the existing verbs (Create, Read, Update, Delete, and Partial Update) and map to a back-end business operation.

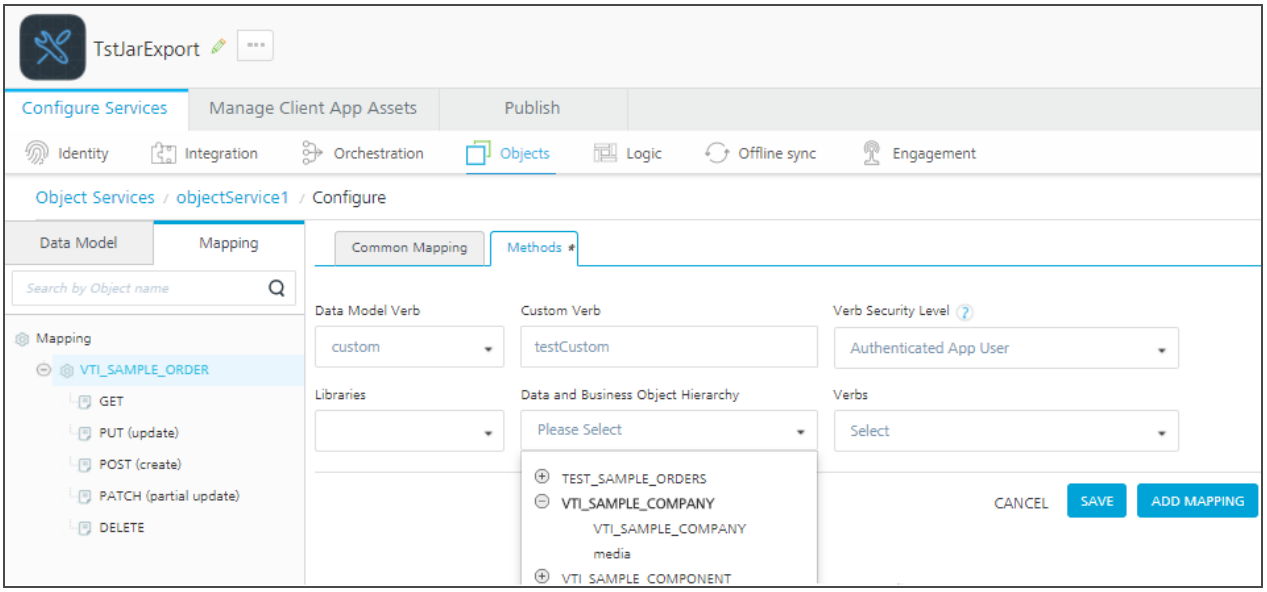
To create a custom verb, follow these steps:

1. In the **Mapping** screen. click the **Methods** tab.
2. Click the **Add** button.



3. Click in the **Data Model Verb** field. A drop-down menu appears. Select **custom**.

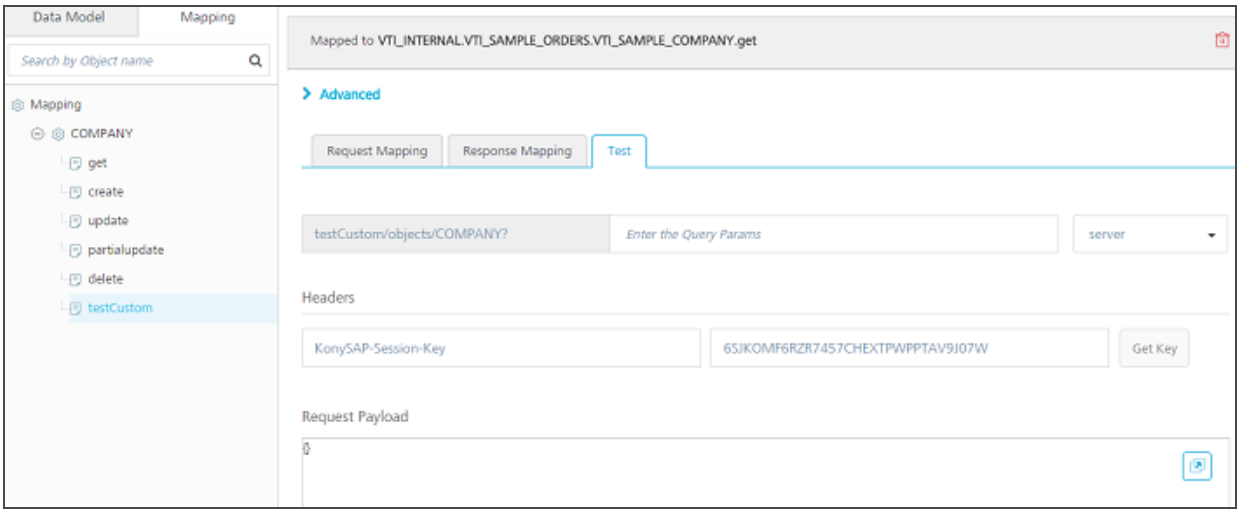
4. Provide the **Custom Verb** name.
5. Use the default **Verb Security Level**, Authenticated App User.



6. Select the **Library**, it automatically populates the list of data and business object hierarchies available.
7. Select the **Business Object** from the drop-down list.
8. Select the verb name from the **Verb** drop-down list and click **SAVE**.

   The created custom verb is successfully added to the list of existing verbs.

9. Select the created custom verb from the **Mapping** section in the left pane.



10. Click **Request Mapping**.

    The verb has the Common Mappings applied by default. You can click on **Clear Mappings** and specify a custom mapping on the request to SAP to map the backend object to application model object. The custom verb will be available to the end users like the other verbs in the application. The rest client can test the custom verb from the published application.

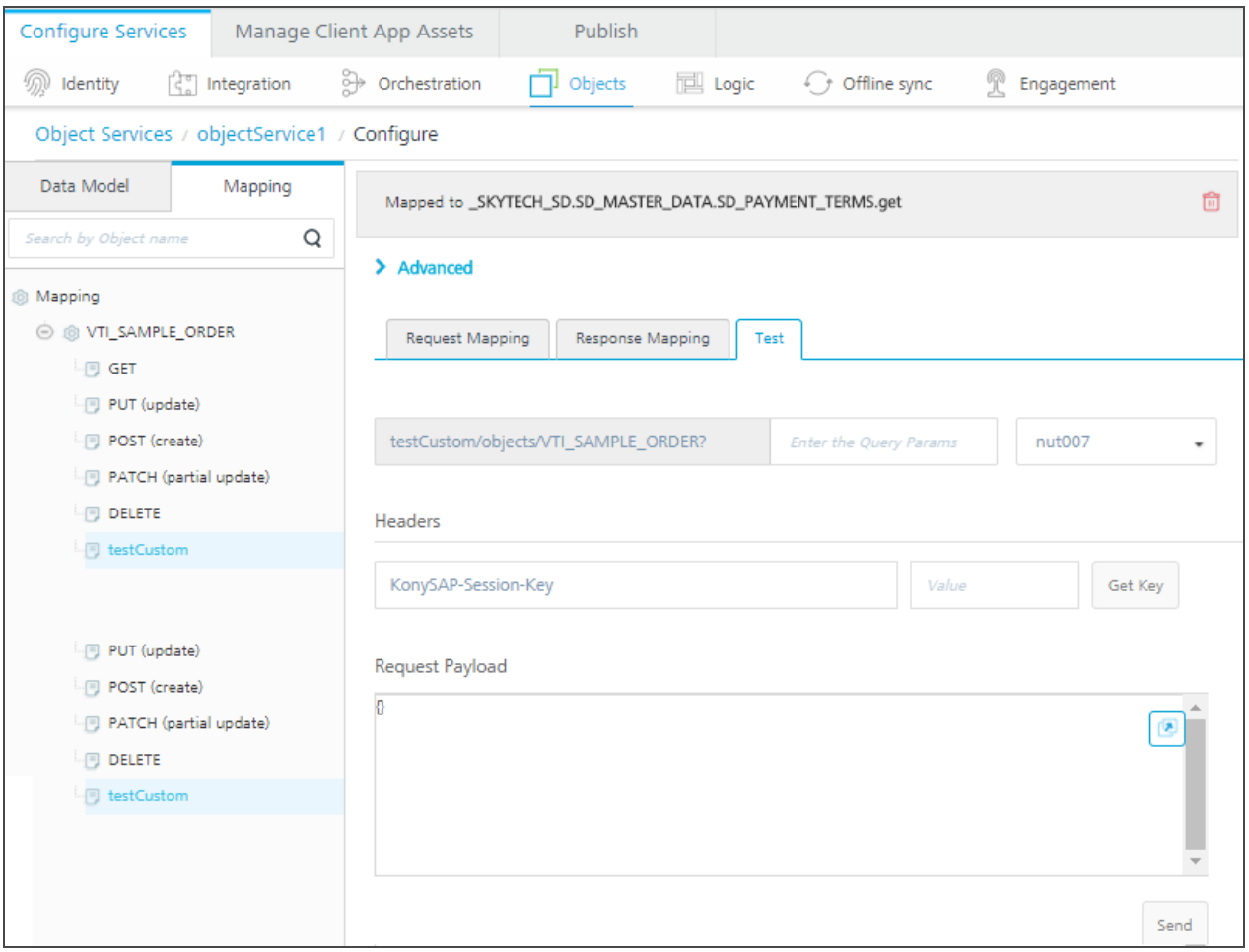
11. Click **Response Mapping**.

    If you do not apply common mapping of the object, you should specify a custom mapping on the response to SAP.

12. Click **Test**.

    You can use the Test panel to test the mapping for a method. To test the mapping, enter the query parameters, select an environment, enter the headers and header values that you want to include with the test. Then, in Request Payload, enter the values for the fields for which you want to test the mapping, and then click **Send**.

    If the test is successful, the end users can invoke the created custom verb from their application post publish.



24.8.1.5 Mapping verbs for Objects (for Service-driven Objects)

To map the verbs for the Defect object, follow these steps:

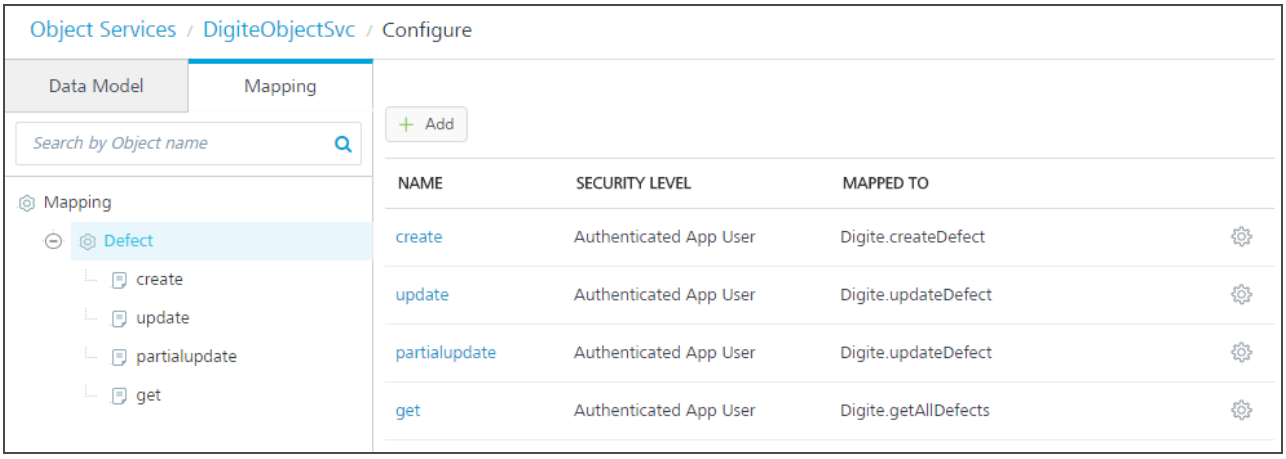
1. In the navigation pane, click the Mapping tab, and then click the object. For example, **Defect**.
2. Click the **Add** button.

   The verb mapping configuration screen for the Account object appears.
3. Click in the **Data Model Verb** field. A drop-down menu appears. Click **Create**.
4. Under **Verb Security Level**, use the default, Authenticated App User.

The Verb Security Level specifies how the client must authenticate to the create verb. Authenticate App User restricts access to the create verb to users that have successfully authenticated using an Identity service. You can also set the security level to Anonymous App User and Public.

5. Click in the **Services** field. A drop-down menu appears. Select the **Digite** integration service.
6. Click in the Operations field. A drop-down menu appears. Click **createAccount**.
7. Click **Save**.
8. Repeat steps 2 through 7, and map the data model verb **update** to the **updateDefect** operation.
9. Repeat steps 2 through 7, and map the data model verb **partialupdate** to the **updateDefect** operation.
10. Repeat steps 2 through 7, and map the data model verb **get** to the **getAllDefects** operation.

    The mapping for the verbs is automatically generated. Mapping is auto-generated only when the dataset name of integration services and object name matches. The mapping is populated in the Request Mapping and Response Mapping tabs based on the input and output of the operations. You can click the **Clear Mappings** button to clear the existing mapping and then specify the custom mapping.

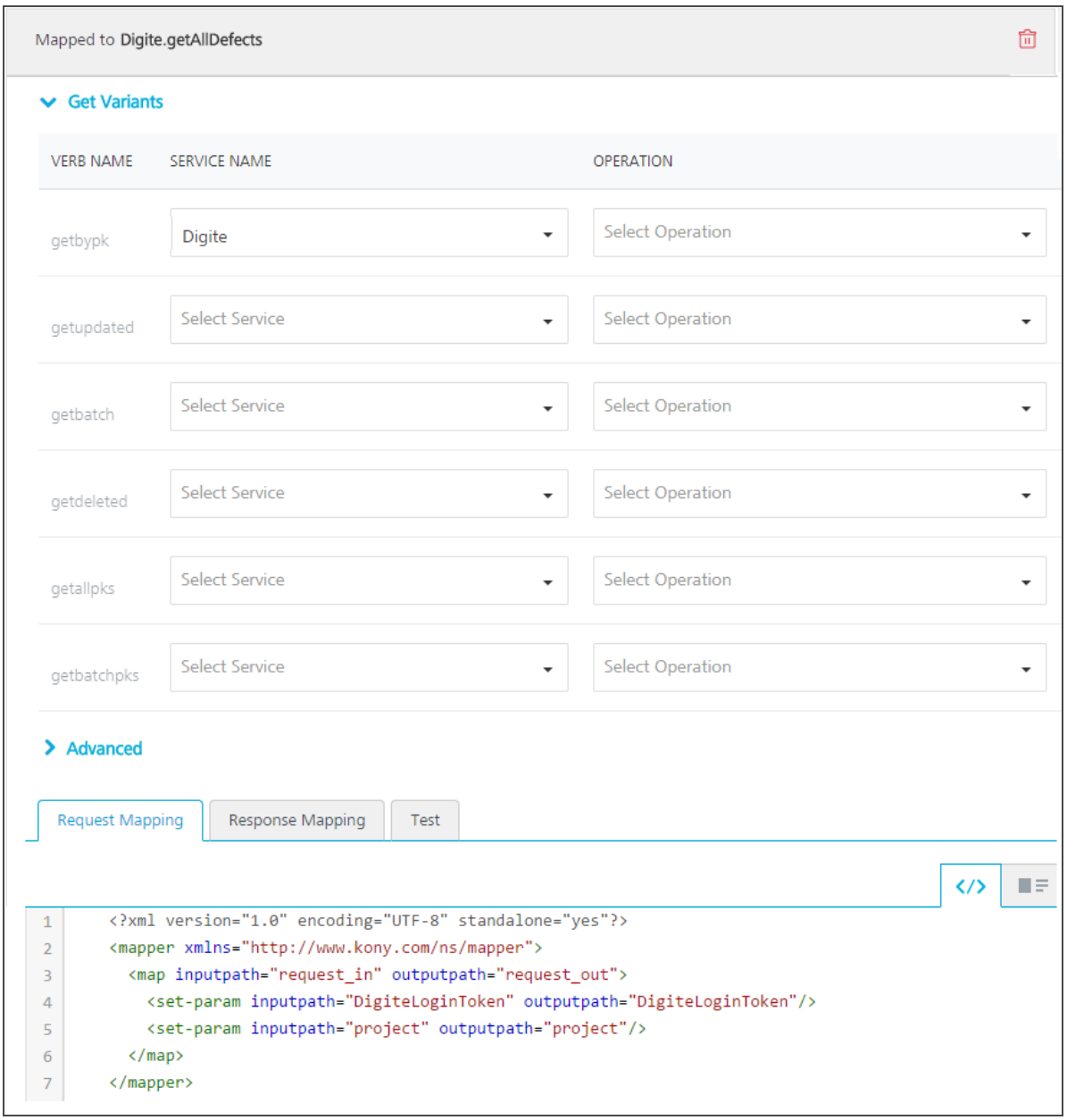


You can view the mapping for a verb by clicking the verb on the Mapping tab in the navigation pane. Request mapping is the parameters that are mapped from the device side to the back end. Response mapping is the parameters that the backend sends to the object's fields.

11. In the navigation pane, under the Defect object, click the **get** verb.
12. Click **Get Variants**.

    The mapping configuration for the get verb appears. Object Services provides built-in variants of the get verb. You can click **Get Variants** button to view the variants for methods. The built-in variants of the get verb are getAll, getbypk, getupdated, getbatch, and getdeleted. These get verbs do not have individual mapping. They have a common request mapping and a common response mapping.
13. Click in the **Service Name** field for the getbypk verb, and select **Digite**.
14. Click in the **Operation** field for the getbypk verb, and select **getDefectDetailsById**.

    A green check mark indicates that the operation mapping has succeeded.



15. Click **Save**.

> ***Note:*** For more information on how to configure Get Variants for SDO, refer to Configuring SDO Get Verb for Offline Objects.

Support for SDO Services from Offline Objects

To support SDO services from Offline Objects, you must configure the multiple variants of a get verb from the integration services.

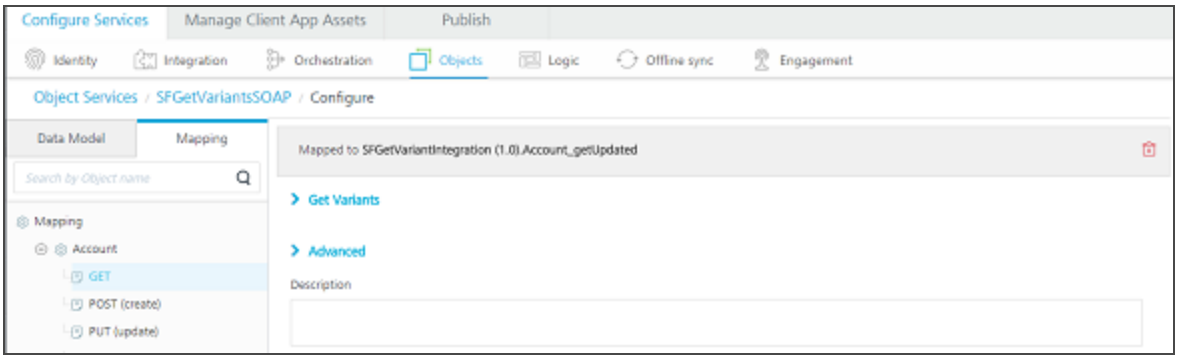
The three types of get variants that support SDO services from Offline Objects are:

- **getUpdated** - Retrieves the initial records updated within the last updated time stamp.
- **getByPK** - Retrieves the records based on the ID. This variant supports the records based on a primary key ID.
- **getBatch** - Retrieves the records of a batch when a batch pointer is provided.

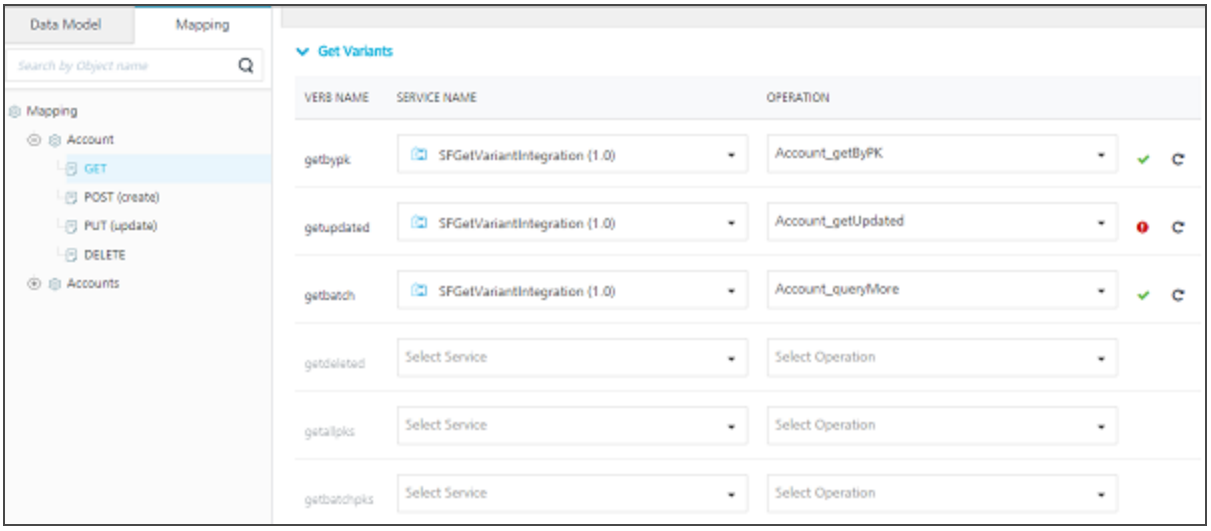
To configure SDO services in Object Service, follow these steps:

1. Create a service driven object service from the integration service.
2. Go to Configure **Services > Objects**, select the corresponding SDO object service and click **Mapping**, and click **Get Mapping**.

   You can find the Get verb is mapped to a single Integration service.



3. Expand Get Variants.
4. Map each get variant getByPK, getUpdated, and getBatch with the corresponding integration service.



All the Get variants share the same mapper. So, all the parameters of the get variants must be defined with the corresponding Integration service.

The get variants getByPK, getUpdated and getBAtch for Offline Objects must return the following mandatory output parameters:

- **hasMoreRecords** – Indicates if there are more records to be downloaded.
- **batchID** – If batching is applicable, the batchID value must be displayed. If batching is not supported, the parameter does not exist in the output params.

In addition to the built-in verb options, you can also create custom verbs for an object. For example, you can create the custom verb **assign**, and map it to the **assignDefect** operation on the back end.

24.8.1.6 Testing an Object Service in Kony Fabric

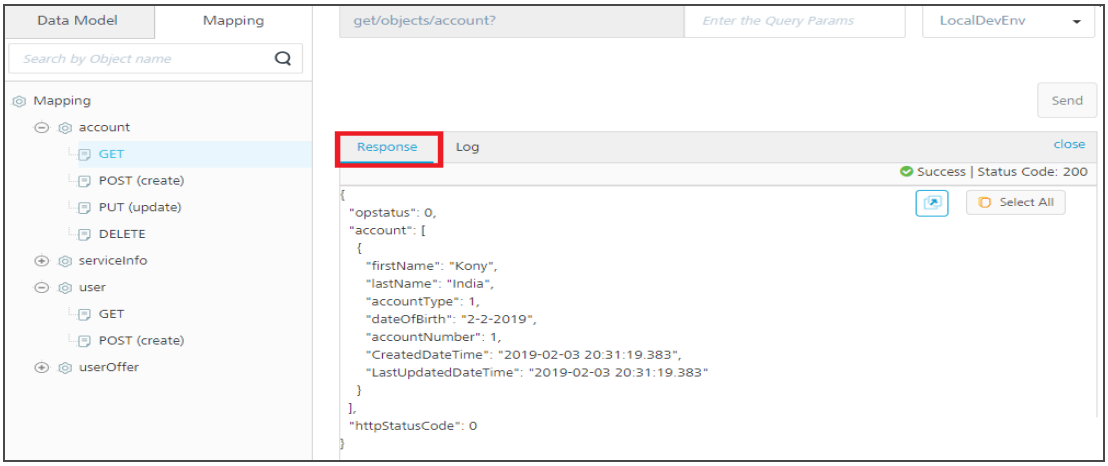
After you publish the service, you can test the service from Kony Fabric. You can only test the published state of the service. You can test the changes you make in the service designer only if you publish the app again.

To test a published service, follow these steps:

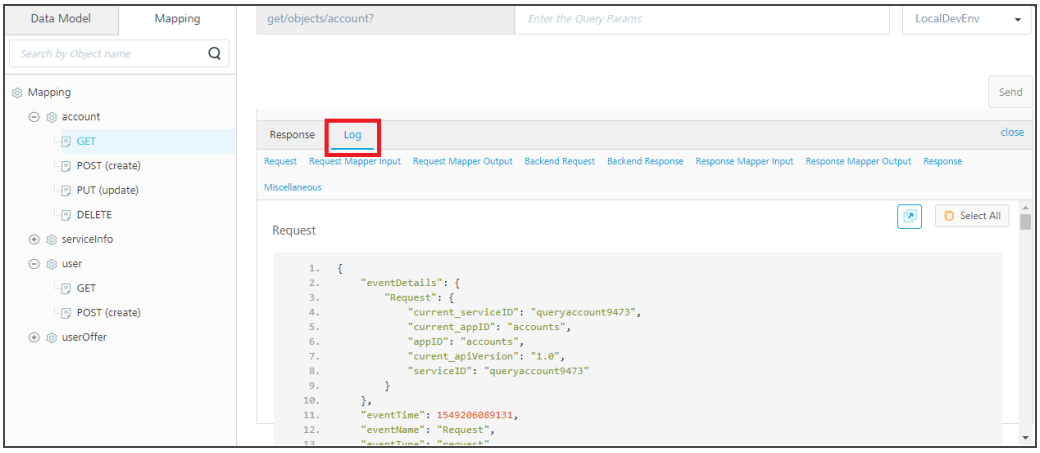
1. In the **Mapping** tab for a Salesforce service, click a verb (for example, get).
2. In the **Configure** screen, click the **Test** tab.
3. In the **Test** tab, select the environment and click **Send**.

   The result dialog displays the **Response** and **Log** tabs:

   - **Response** tab displays the final response of the service.



   - **Log** tab displays the following details



- **Request** displays the request data such as Service ID, App ID and so on.
- **Request Mapper Input** displays the mapper input in the request flow.
- **Request Mapper Output** displays the output of mapper in request flow.
- **Backend Request** displays the data sent to the backend.
- **Backend Response** displays the response received from the backend.
- **Response Mapper Input** displays the input to mapper in response flow.
- **Response Mapper Output** displays the output of mapper in response flow.
- **Response** displays the final output of service and the status showing success or failure of test call.

Limitations for testing an Object Service

- Testing of Pre-Processor and PostProcessor is not supported.
- Testing of Storage Objects is not supported without publishing.

You can now "Creating a Mapping by using Visual Mapper" on page 914

Java Sample Code for Preprocessor and Postprocessor - Objects

Sample ObjectServicePreProcessor

The following is a sample ObjectServicePreProcessor:

```
 public class SamplePreProcessor implements ObjectServicePreProcessor
{
@Override
public void execute(FabricRequestManager requestManager,
FabricResponseManager responseManager,
FabricRequestChain requestChain) throws Exception {
PayloadHandler requestPayloadHandler =
```

```
requestManager.getPayloadHandler();

JsonObject object = requestPayloadHandler.getPayloadAsJson
().getAsJsonObject();
object.addProperty("Kony", "Hello World!");


requestPayloadHandler.updatePayloadAsJson(object);
requestChain.execute();
}
}
```

Sample ObjectServicePostProcessor

The following is a sample ObjectServicePostProcessor:

```
 public class SamplePostProcessor implements
ObjectServicePostProcessor {
@Override
public void execute(FabricRequestManager requestManager,
FabricResponseManager responseManager)
throws Exception {
PayloadHandler responsePayloadHandler =
responseManager.getPayloadHandler();

JsonObject responseAsPayload = responsePayloadHandler.getPayloadAsJson
().getAsJsonObject();
responseAsPayload.addProperty("Kony", "Hello World!");

responsePayloadHandler.updatePayloadAsJson(responseAsPayload);
}
}
```

24.8.1.7 Configuring SDO Get Verb for Offline Objects

You can create a service-driven object (SDO) from a set of existing Kony Fabric Integration/Orchestration Services. These Integration services connect to existing API endpoints or to Kony Fabric Orchestration Services, which combine multiple APIs into a new composite or aggregate API.

This section helps you how to configure the Get variants supported for the Get verb for SDO services.

Types of Get variants in Offline flow:

- Download Flow: indicates when back-end responses are sent to client devices.
    - Initial Sync
    - Delta Sync
    - Batch
- Upload Flow: indicates when client requests are sent to back ends.
    - Conflict
- Mapping SDO which supports OData
- Mapping SDO which doesn't support OData
    - Creating Integration services.
    - Object Service mapping of get variants
- Sample App for Get Variants Verbs for SDO

Types of Get variants in Offline flow

The following are the different types of the `Get` variants help you to achieve the core functionalities of the offline objects.

Download Flow

In Download Flow, the back-end responses are sent to client devices.

24.8.1.8 Initial Sync

You must download the initial set of records from a back-end server to a device. If a back end supports batching, the initial sync of data will be downloaded in batches.

This is referred as `getall`. Generally, the GET verb mapper should be able to do this.

24.8.1.9 Delta Sync

You must download all the records that are changed (for example: `created`, `updated`, or `deleted`) after the previous download (`lastUpdateTimestamp`) from the back-end server.

For example, the back-end server sets the `$filter` like this `$filter = <LastModifiedDate> gt <value>` internally to achieve this. This form of get is referred in terms of offline as `getupdated` variant.

24.8.1.10 Batch

You need to use the `batchpointer` to download a particular set of records in that batch.

This form of get is referred in terms of offline as `getbatch`variant.

Upload Flow

24.8.1.11 Conflict

While determining conflict, back-end data-source is first queried with the given primary key which has to be uploaded. So, if a record is present in a client device, and the client record different from the record in the back-end server, then it is called conflict.

This form is get is referred in terms of offline as `getbypk` variant.

A back-end server sets the `$filter` like this `$filter = <PrimayKeyColumnName> eq <value> eg $filter = Id eq 10`

In the upload flow, if a conflict policy is defined, then the get variants must be configured.

Mapping SDO which supports OData

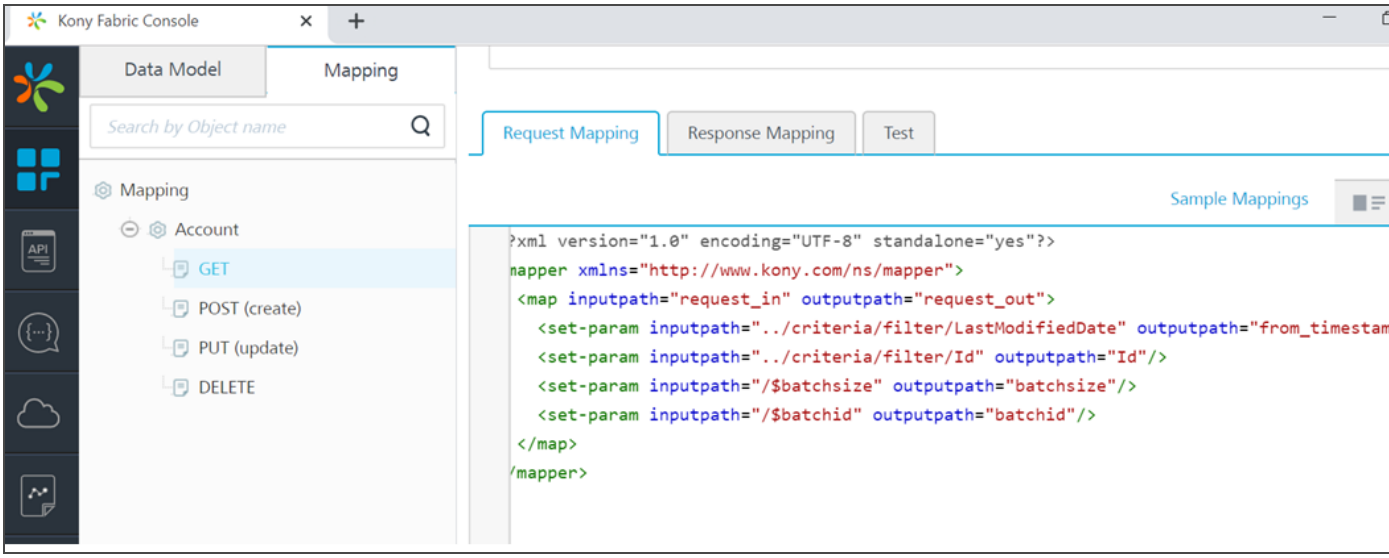
Follow these steps to map the get verbs if a back-end server supports OData:

1. Create an object service type as **Integration/Orchestration**.
2. Define the corresponding GET verb.
3. As underlying integration supports OData, edit the GET verb request mapper as below to pass on the corresponding OData query options.
4. Add the lines highlighted to the **Request Mapper** of the `GET` verb. This is to ensure that you are passing OData fields to underlying integration services to act accordingly.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mapper xmlns="http://www.kony.com/ns/mapper">
<map inputpath="request_in" outputpath="request_out">
<set-param inputpath="/$filter" outputpath="/$filter "/>
<set-param inputpath="/$batchsize" outputpath="batchsize"/>
<set-param inputpath="/$batchid" outputpath="batchid"/>
</map>
</mapper>
```

   - The `$filter` confirms to OData 2 specifications.
   - The `$batchid` represents the batchpointer to be fetched. (optional; required only when batching is supported)
   - The `$batchsize` represents the size of a batch to be fetched. (optional ; required only when batching is supported)

   No changes to **Response Mapper** is required.



Mapping SDO which does not support OData

Follow these stages to map the get verbs if a back-end server does not support OData.

1. **Stage:** Creating Integration services
2. **Stage:** Object Service Mapping of get variants

Creating Integration services

Create one of the following integration types:

- **A single integration service is present.**

  If a single integration service supports all the features required by `initialSync`,`deltaSync` and `getbypk`, then create that service and map that to the Get verb and provide the mapper accordingly.

- **Multiple integration services are present**

- **For Initial sync.**

  A service which would download all the records. This is ideally general get service mapped to GET verb.

  Input -> No input params required.

  Output -> The following conditions are mandatory:

  - The response should contain the list of all records of the selected object.
  - The response should return the `hasMoreRecords` parameter, which indicates more records to be downloaded

- **For Batch sync:** The following is to be configured when batching is supported:

  The `nextBatchId`which would represent the current download yielded in to batch and this represents the next `batchPointer`.

  > ***Note:*** While mapping to ObjectService verbs, this can be mapped to GET verb.

- **For Delta sync:** `getUpdated service`

  A service which would take the `timestamp`as an input and fetches the data from that timestamp.

  Input ->

  - The request should take timestamp as the parameter.

  If batching is supported can also take

  - One more parameter which would take batchsize as pointer.

Output ->

- The response should contain the list of all records of the selected object.
- The response should return the `hasMoreRecords` parameter, which indicates more records to be downloaded
- The `nextBatchId`, which represents the current download yielded in to batch and this represents the next batchPointer. Mandatory only when a batching is required.

> ***Note:*** In `ObjectService getVerbMapping` this can be mapped to the `getupdated` verb. If the same service can download, all the records for some default value of timestamp same service can also be mapped to GET verb.

To download progressive batches - getbatch Service

A service which would take the batch pointer as the parameter and download that batch

Input

- The request should take batchid as the parameter.

Output

- The response should return the list of records of an object
- The response should return the `hasMoreRecords` parameter, which represents still any more records to be downloaded
- The `nextBatchId`, which represents the current download yielded in to batch and this represents the next batchPointer.

> ***Note:*** In ObjectService getVerbMapping this can be mapped to getbatch service. In this case the batchPointer should return from the above defined getUpdated and get services

- **If Conflict needs to be supported by SDO - getbypk service**

  A service which takes the primary key of an objects as an input param and downloads that particular record.

  Input

  - A param which represents the PK of an object as an input.

  Output

  - The response should return all the fields of a particular record.

  > ***Note:*** In ObjectService getVerbMapping this can be mapped to getbypk service.

  > ***Important:*** The response of the `getbypk` verb should be JSON array format.

  These are the different flavors of get that supports initial sync, delta sync, batching and conflict functionality of offline objects.

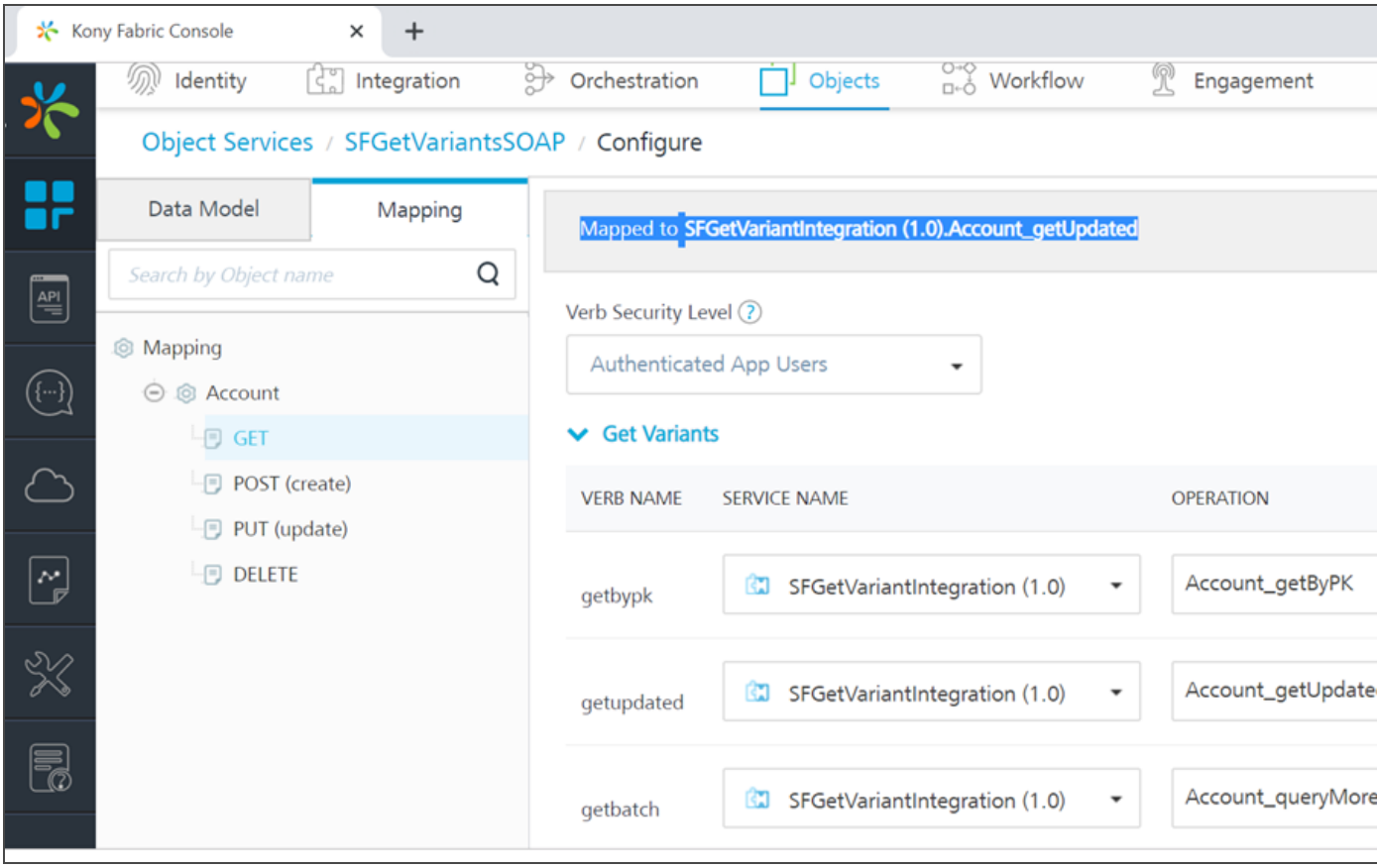
Object Service Mapping of get variants

1. After you complete the steps in the previous section, now create an object service from the above defined integration service.
2. Go to **GET** verb mapping to map all the above defined integration services as below:
3. Go to **Configure services >Objects >** Select the corresponding SDO object service -> click on Mapping of a particular object -> and click on Get Mapping.

In the screen shot above you could see main GET verb is mapped to getupdated service only.

> ***Note:*** In the GET verb mapping, you can also configure extra variant mappings such as `getByPk` and `getbatch` and so on. For more information, refer to Known Issues.

4. Click on Get Variants and map the corresponding integration services to each get variants as below:

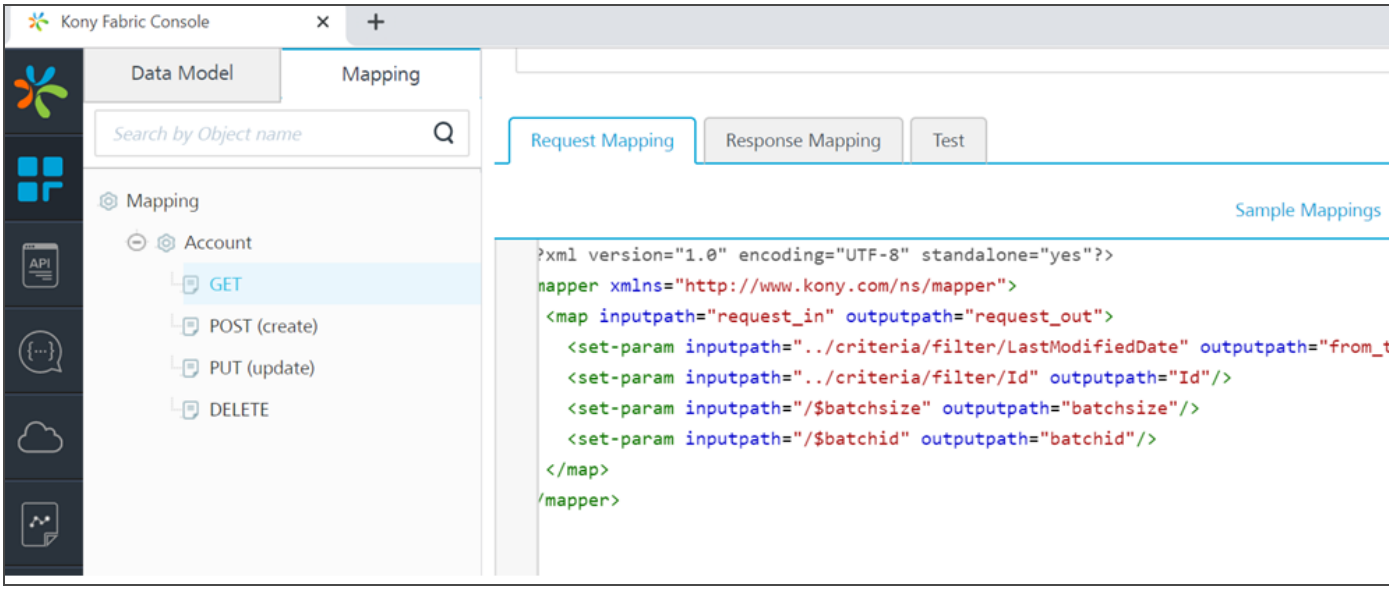


Each getvariant getbypk and getupdated and getbatch should be mapped to corresponding integration service as defined above.

All the variants of get share the same mapper. So, all the params required for all the above mentioned getVariants should be defined.

The ones highlighted in blue needs to be added to the generated mapper.

Request Mapper



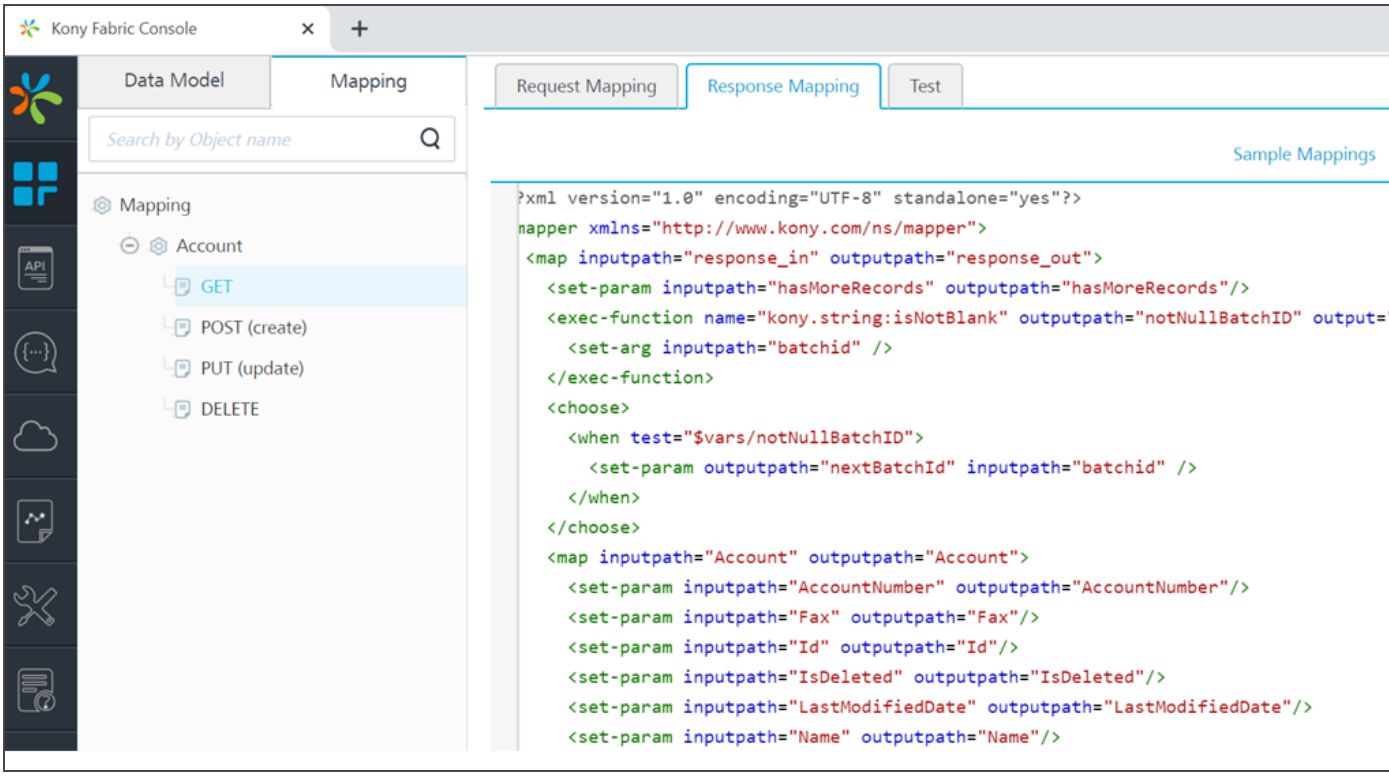
Filter propagation

You can access the filter from the criteria node like this `../criteria/filter/<key> ..`.
This will give that value of `<key>` present in filter.

For example, `$filter = Id eq 10`

The above `../criteria/filter/Id` will give value 10.

Response Mapper



The ones highlighted in blue needs to be added **Response Mapper**.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mapper xmlns="http://www.kony.com/ns/mapper">
<map inputpath="response_in" outputpath="response_out">
<set-param inputpath="hasMoreRecords"
outputpath="hasMoreRecords"/>
<exec-function name="kony.string:isNotBlank"
outputpath="notNullBatchID" output="$vars">
<set-arg inputpath="batchid" />
</exec-function>
<choose>
<when test="$vars/notNullBatchID">
<set-param outputpath="nextBatchId" inputpath="batchid" />
</when>
</choose>
<map inputpath="Account" outputpath="Account">
```

```
<set-param inputpath="AccountNumber" outputpath="AccountNumber"/>
…..
<default generated mapper ctnd…>
</map>
</map>
</mapper>
```

The get variants `getall`, `getupdated`, and `getbatch` for offline should return the following mandatory output params:

- **hasMoreRecords**: This indicates whether all the records have been downloaded or not. Based on Boolean value client will decide whether to continue further downloading or not.
- **batchId**: If download falls to batching, this represents the `nextbatchid`value. If batching is ended, this parameter should not be present in the output params even with empty values.

  In the above mapper null check on `batchid`is done to make sure no `nextBatchId` param is present if it is null.

  Please note that above two params are case sensitive.

Sample App for Get Variants Verbs for SDO

For more hands-on approach on how to implement GetVariants verbs, download and import the SFSoapGetVariants.zip the sample app to preview the details using Kony Fabric ?

If required, replace the Salesforce credentials with yours.

https://konysolutions.atlassian.net/wiki/pages/SFSoapGetVariants.zip

24.8.2 Creating a Mapping by using Visual Mapper

Object Services includes a visual designer that lets you add, modify, and delete request and response mappings of any of your Object Services by using a drag-and-drop interface.

The back-end object, data model, and the other objects and functions associated with the service are each displayed in their own container. Each container has one or more items listed within it. You can click on an item and drag your cursor to another item in another container to create new mappings.

Each mapping has a corresponding color associated with it.

- Gray corresponds to a valid mapping.
- Blue corresponds to a common mapping.

The design view is available along with a XML code view of your mappings. You can quickly switch between code view and the visual design view by clicking on the appropriate tab in the viewing pane.

> ***Note:*** This video will walk you through understanding the Visual Mapper in your application: https://youtu.be/5JH9s3Qzx5g

The procedures in this section describe how to use the visual designer to manage your Object Service mappings.

24.8.2.1 How to View and Edit Mappings

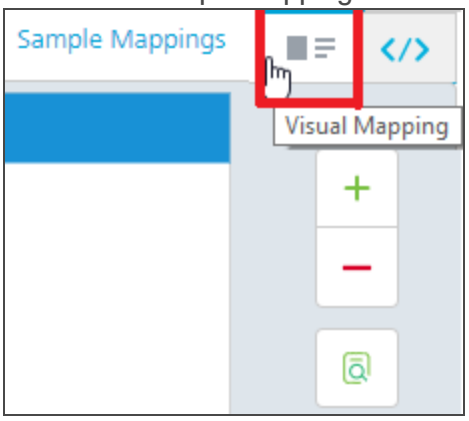
You can examine any Object Service in visual design view that has request and response mappings. This section describes how to access visual design view for your services.

> ***Note:*** The default visual design view pane is view-only and cannot be edited.

You can also edit your mappings by clicking the Edit button at the bottom of the viewing pane.

To access visual design view for a specific object service, do the following:

1. In the **Object Services** list, click the **…** button on the right side of the service listing you want to view, and then click **Edit Configuration**.
2. In the navigation pane, click the **Mapping** tab.
3. In the **Mapping** tab, click the item you want to view.
4. Click either the **Request Mapping** or the **Response Mapping** tab.
5. Under **Request Mapping** or **Response Mapping**, click the design view tab, which is located next to the Sample Mapping button.

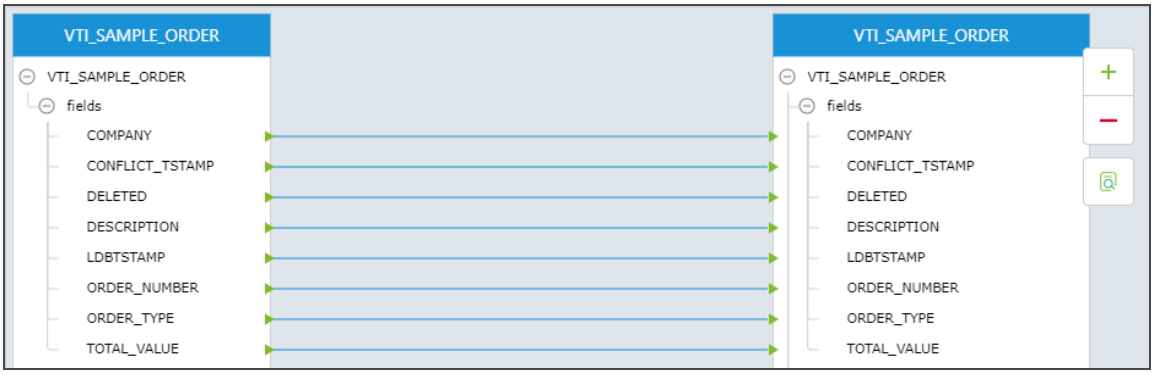


After you are in visual design view, you can increase and decrease the zoom level of the viewing pane by clicking the **+** and **-** buttons in the upper-right corner of the pane. In addition, you can hide mappings by clicking the button below the zoom buttons and selecting which type of mappings you want to hide.

6. If you want to edit your mappings for this service, click the **Edit** button in the lower-right corner of the page.

24.8.2.2 How to Add Mappings

You can map objects to each other by using your mouse to click on one item and then dragging to another item.



Depending on the type of item, the item can have multiple mappings from several different sources.

To map one item to another, do the following:

1. Click and hold on the item you want to connect.
2. While continuing to hold the mouse button, move your mouse cursor to the item you want to connect to and release the mouse button.

> ***Note:*** If you attempt invalid mapping, an error message will appear.

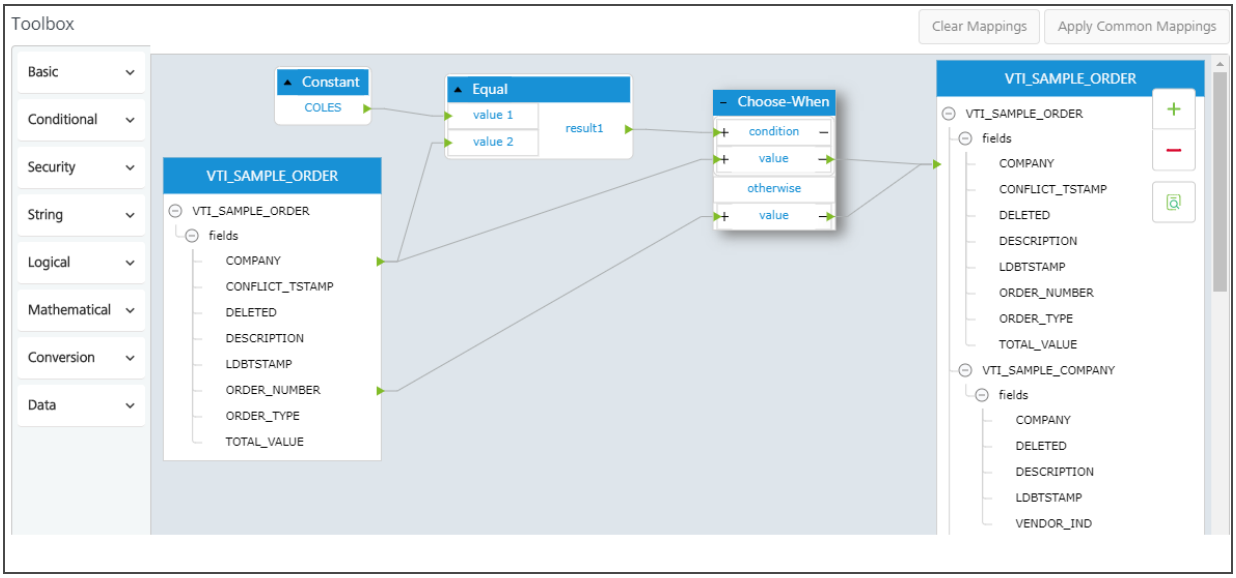
24.8.2.3 How to Remove Mappings

You can remove all the existing mappings by clicking the **Clear Mappings** button.

If you want to remove individual mappings, select the mapping you want to remove, and press delete on your keyboard. You can select multiple mappings by holding down either `Shift` or `Ctrl` keys and then selecting more mappings.

24.8.2.4 How-to Use Functions in the Visual Designer

When the visual designer is in **Edit** mode, the **Toolbox** pane lists built-in functions on the left side of the view.



The list includes all built-in mapper functions, as well as any custom functions you have created.

You can add new functions to your service by dragging and dropping the function from the list. You can also add, edit, import, export, and delete any custom functions you have.

To add a built-in function, do the following:

1. Select the function you want to use from the function list.
2. Click and hold the mouse button on the function.
3. While continuing to hold the mouse button, move your mouse cursor to the location you want to place the function and release the mouse button.

You can now Enhance the Mapping by using XML Mapper for advanced scenarios

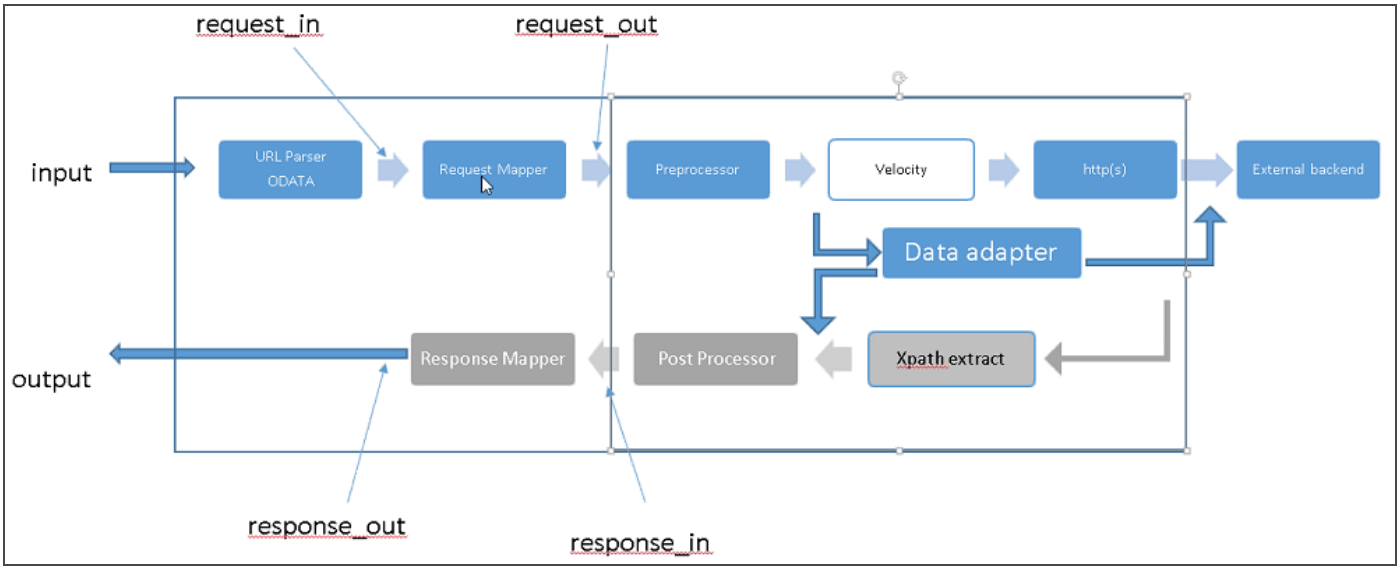
24.8.3 Enhancing the Mapping by using XML Mapper for Advanced Scenarios

The core mapper engine is integrated into the Object Services pipeline. The mapper is an engine that logically takes in JSON, by using the declarative XML that it is passed, modifies the JSON, and creates nodes in the JSON and updates existing nodes.

For the request that is coming in to the mapper, after basic URL parsing and ODATA parsing are performed, all the context is made available to the request mapper. The request mapper modifies the payload and sends it into the next stage of the pipeline, which goes to the back end, typically by using an integration connector.

After the response is picked up, the whole context is made available to the response mapper. The Response mapper makes changes to the data, and outputs the data after some formatting. The internal mapper logic is the same for both the input and output pipelines, but the context that is made available to the mapper changes.

The following diagram shows the Object Services pipeline.



24.8.3.1 Request Mapper

On the input, when the mapper is applied, it is equivalent to the following.

```
mapper("input mapper xml", $current_input = data)
```

The $data is logically a blob with all the context made available to the mapper. You are passing in transformational XML to run on the current input. The mapper input can access any of the data that is passed to it.

request_in : Read-only, input payload JSON from API request.

request_out : Read-write, (a copy of input payload), transformed JSON that is sent to next stage.

vars : Empty. This is referenced using $vars and is used by mapper XML as a global variable store

- For more details, refer Simple Mapping example

24.8.3.2 Response Mapper

On the output, when the mapper is applied, it is equivalent to the following.

```
mapper("output mapper xml", $current_input = data)
```

The $data is logically a blob with all the context made available to the mapper.

response_in : Response from back end logically represented as JSON.

response_out : Transformed JSON using the XML mapper sent in the "records" field of the API response.

vars : Empty. This is referenced using $vars and is used by mapper XML as a global variable store.

- For more details, refer Simple Mapping example

24.8.4 Mapper Elements

The mapper includes the following built-in variables, built-in functions, and blocks.

Built-in Variables

```
 a/b/c   /* An input path of a.b.c in the given object */
../b      /* inputpath = "b" on $current-input.getParent() */
 $current-input /* Input instance of current map block */
 $current-output /* Output instance of current map block */
 $mapper-input /* Mapper instance's global input */
```

```
 $mapper-output  /* Mapper instance's global output */
 $vars/x /* Variables are stored in this hash.  All variables are
GLOBAL */
 $args/<arg name> /* Arguments passed to a function */
```

On a function call, by default, `$args.current-input` is set to `$current-input`, and `$args.current-output` is set to `$current-output`.

`input="myinput" inputpath="firstname"` *is equivalent to* `$myinput["firstname"]`

`inputpath="firstname"` *is equivalent to* `$current-input["firstname"]`

`Input="3" /* not starting with a $, it is constant value. To use $ as a constant, use the escape character, for example, "\$".*/`

`output="$myoutput" outputpath="lastname"` *is equivalent to* `$myoutput["lastname"]`

Built-in functions

The Kony namespace contains a number of built-in functions that you can use with the mapper. The built-in functions are:

- equal
- concat
- substringBefore
- substringAfter
- choose-when-otherwise
- random
- min

- max
- sum

> ***Note:*** The kony namespace is reserved for Kony-defined functions only. User-defined functions cannot use the kony namespace.

Blocks

The following table describes the blocks available:

Block	Description	Example
map	This is the main block in mapper.	`<map inputpath="x/y/z" input="$current-input" outputpath="x" output="$current-output"> </map>`
set-param	Sets an output parameter from input.	`<set-param inputpath="x/y/z" input="$current-input" outputpath="x" output="$current-output" /> Output.outputpath=input.inputpath`

Block	Description	Example

set-arg	Variant of set-param to set the argument of a function.	`<set-arg name="x" inputpath="x/y/z" input="$current-input" />`
		is a shortcut for the following:
		`<set-param inputpath="x/y/z" input="$current-input" outputpath="x" output="$args" />`

set-variable	Variant of set-param to define and set a global variable.	`<set-variable name="x" inputpath="x/y/z" input="$current-input" />`
		is a shortcut for the following:
		`<set-param inputpath="x/y/z" input="$current-input" outputpath="x" output="$vars" />`

Block	Description	Example
exec-function	Execute function. The return value of exec-function is written to the output/outputpath attributes of exec-function.	<map inputpath="contact" outputpath="contact"> <exec-function name="contact-field-map"> /* input is set to $current-input and output is set to $current-output by default */ </exec-function>/ </map> /* Complex */ <map inputpath="contact" outputpath="contact" output="$current-output" > <exec-function name="field-map" outputpath="myresult" output="$vars"> <set-arg name="a" inputpath="firstname" input="$current-input" > <set-arg name="b" input="3" > </exec-function> </map> $vars[myresult] = field-map({a: "$current-input" ["firstname"], b: "3" })

Block	Description	Example

choose-when-otherwise	When a "test" condition evaluates to true, then the "when" block is executed. If the "test" condition evaluates to false the "otherwise" block is executed.	`<choose>`
		`<when test="$vars/NTFCondition">`
		`...`
		`</when>`
		`<when test="$vars/CONCondition">`
		`...`
		`</when>`
		`<otherwise>`
		`...`
		`</otherwise>`
		`</choose>`

Create-lookup	Create-lookup loops on the inputpath array and creates a hashmap with a lookup-key parameter as key and value as “Node” object refers to corresponding customer row of the input object.	`<create-lookup inputpath=”Customers” output=”$vars” ouputpath=”customerMap”>`
		`<lookup-key inputpath=”Id”/>`
		`</create-lookup>`

Lookup	Retrieves the “Node” object from the hashmap and makes it available in the output.	`<lookup input=”$vars” inputpath=”customerMap” outputpath=”customerRef” output=”$vars”>`
		`<lookup-key inputpath=”CustomerI`
		`</lookup>`

Block	Description	Example
Create-group	Create-group generates an aggregate group of objects based on a key designated by the group-key block. Each object is contains multiple entries. Each entry is a key/value pair.	<create-group inputpath=”Time_Entry” output=”$vars” ouputpath=”customerMap”> <group-key inputpath=”Timesheet_Id”/>
Group-key	Key used to group items for aggregation.	<group-key inputpath=”Timesheet_Id”/>
javascript	JavaScript element must occur one time only as a child element to the Function element. Then name of outputpath attribute represents a field in the app data model.	<function name="NameConcat"> <javascript outputpath="FullName"> <set-arg name="firstName" inputpath="FirstName" /> <set-arg name="middleName" inputpath="MiddleName" /> <set-arg name="lastName" inputpath="LastName" /> <script> You logic goes here..... </script> </javascript> </function>

Block	Description	Example

script	The Script element is a required child element in a JavaScript element. It must occur one time only. The JavaScript snippet should be written in **CDATA** section of this element.	<script>
		<![CDATA[
		function concat(){
		var result = firstName+' '+ middleName + ' '+lastName.substring(0, 1).toUpperCase() + '. '
		return result;
		}
		concat();
		]]>
		</script>

24.8.4.1 Example: Mapper Structure

The following example shows the structure of the mapper in declarative XML.

```
 <mapper xmlns="http://www.kony.com/ns/mapper">  /* root element */
<function name="field-map-a2b"> /* Function definitions */
<set-param />
<map input= output= inputpath= outputpath=></map>
</function>
<map input= output= inputpath= outputpath=> /* Mapper blocks */
<set-param input= inputpath= output= outputpath = />
<set-variable input= inputpath= output= outputpath=  />
<exec-function name="field-map-a2b">
<set-arg />
</exec-function>
</map>
<map> </map>  /* another map node */
</mapper>
```

24.8.4.2 Example: Simple Mapping

In this simple mapping example, the input is a list of records under A, and each record has P1 and P2 fields that we want to output to the API response by transforming P1 to Q1 and P2 to Q2.

The back end is returning columns by the name P1 and P2, where the object on the client side has fields named Q1 and Q2. For example, the backend could be returning FName and LName, and these are mapped to fields with more friendly names on the client side, Firstname and Lastname. The mapper, in each of the records it gets, picks up the value of P1, put it into a key of Q1 in the output.

The mapper is output-driven, so in the output, choose a path with the name A (outputpath="A"), and set up a parameter Q1 in output (set-param outputpath="Q1" inputpath="P1"), by taking the value from P1 from the input from a path with the name A (inputpath="A").

Input

```
 {
"A": [
{
 "P1": "value1",
 "P2": "value2"
}
]
}
```

Output

```
{
"A": [
{
 "Q1": "value1",
 "Q2": "value2"
}
]
}
```

Mapping

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mapper xmlns="http://www.kony.com/ns/mapper">
//"The following map tag for response or request"//
<map inputpath="response_in" outputpath="response_out">
<map outputpath="A" inputpath="A">
<set-param outputpath="Q1" inputpath="P1" />
<set-param outputpath="Q2" inputpath="P2" />
</map>
</mapper>
```

24.8.5 Identity Support in Mapper

The mapper can access identity security or profile attributes if they are needed by your service. For example, your service requires an input parameter that is available from the identity profile attribute.

To use identity support, follow these guidelines:

- The inputpath should be defined in the format "identity/*<IdentityProviderName>*/*<Security_Or_Profile>*/*<token name>*"
- In the defined structure, you must select either security or profile, depending on the specific identity token type that you want.
- Input should be defined as "$mapper-input", which points to the mapper input node.

24.8.5.1 Example: Accessing the Identity Profile Attribute

The following example shows how to access the identity profile attribute using the mapper. The example is a request mapper that retrieves the value for the "PRODUCT" parameter from the "user_id" token of the "SAPIdentity" identity provider.

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mapper xmlns="http://www.kony.com/ns/mapper">
<map inputpath="request_in" outputpath="request_out">
```

```
<set-param inputpath="identity/SAPIdentity/profile/user_id"
input="$mapper-input"    outputpath = "PRODUCT "/>
</map>
</mapper>
```

24.8.5.2 Example: Accessing the Identity Security Attribute

The following example shows how to access the identity security attribute using the mapper. In this example, an identity security attribute value is mapped to a result parameter named "SAP-Session-Key-Value".

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mapper xmlns="http://www.kony.com/ns/mapper">
<map inputpath="request_in" outputpath="request_out">
<set-param inputpath="identity/SAPIdentity/profile/user_id"
input="$mapper-input"    outputpath = "PRODUCT "/>
</map>
</mapper><?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mapper xmlns="http://www.kony.com/ns/mapper">
<map inputpath="response_in" outputpath="response_out">
<map inputpath="Sample_Object" outputpath="Sample_Object">
<set-param inputpath="identity/SAPIdentity/security/KonySAP-Session-
Key" input="$mapper-input" outputpath="SAP-Session-Key-Value"/>
</map>
</map>
</mapper>
```

For more details on enhanced identity data-filtering, refer Enhanced Identity support in Object Services

24.9 Integrating Object Services in an Application

After you complete mapping operations to methods for your objects, you can integrate these objects in an application by any of the following methods:

- Integrating Object Services in an application by using Kony Visualizer SDKs.
- Integrating Object Services in an application by using the following Kony Fabric SDKs:
  - for iOS
  - for Android
  - for Cordova (PhoneGap)
  - for JavaScript
  - for .NET (Visual Studio)

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

24.10 Object Metadata for Controlling Client-side Logic

Object services from Kony V8 SP4 allow app developers to define metadata per field of the object in the Fabric server. This metadata of the Object allows app developers to control client-side logic by using the **Data Pre and Post Processors for Models** functionality which helps to access the metadata dynamically at runtime and has custom JS code written to transform the data based on the metadata. This allows app developers to modify business logic at the client app and control which fields get transformed and how.

The feature is available with object models generated from the Object services in Kony Visualizer and works for Kony Reference Architecture as well as Kony free form project types.

You can use **data pre and post processors functionality** on client apps as standalone or in conjunction with **pre and post processors on Object services** on server apps to achieve a variety of

business requirements. For example, you can write your custom logic for fields in models to transform data in client before sending to back end and vice versa such as data encryption, decryption, currency or temperature conversion, prepend or append characters such as salutations (Mr. or Dr.) or currency symbol ($ or €) and so on.

24.10.1 Use Cases

- **Use Case 1: Secure your Sensitive Data with Encryption over network calls:**

  If you wish to encrypt sensitive fields in a transformation such as a credit card number the user specifics the card number in a client app, while non-sensitive fields like amount and vendor can go unencrypted. The user-specified value for the credit card number in the request operation gets encrypted based on the metadata specified on server and logic defined in the **Data Pre-processor** on a client in the network call. In this case, the encrypted data from a client request can be stored in databases, in Cloud, or computer hard drives.

  - To decrypt the user value to the original state before sending it to the back end, you must specify custom logic in **Data Pre-processor** in the request operation of the service in Kony Fabric.

- **Use Case 2: Data Conversion from a server to a client:**

  You can convert a temperature value in Celsius (°C) from a server and display it in Fahrenheit (°F) degrees to a client. That is, Server data is always in Celsius and client display is always in Fahrenheit. This can be extended to ensure same front-end app can be made to display temperature in Celsius or Fahrenheit depending on country. In this case, you need to specify custom logic in **Data Post-processor** in the response operation of the service in Client app for display in Fahrenheit, and **Data Pre-Processor** in client for any data modified from client app to reach server as Celsius for an upload request.

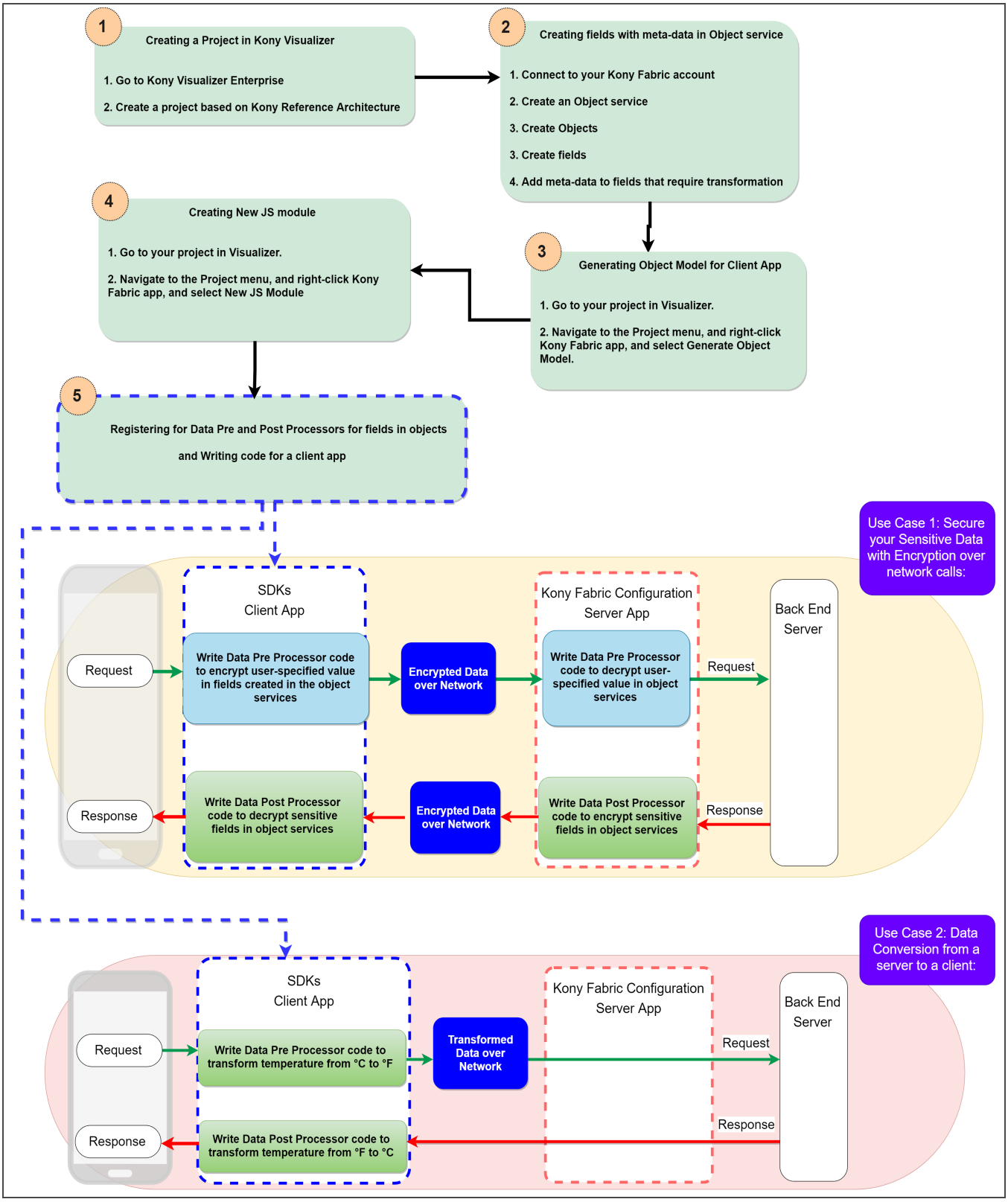
24.10.2 Pre-requisites

- Kony Visualizer Enterprise V8 SP4 or higher
- Fabric application is based on Object services (data models)

24.10.3 Steps to enable Client-side Logic

1. Define metadata for fields in Objects for which transformation is required.
2. Generate models on the client app through Visualizer for the object services.
3. Register Pre and Post processors for the models that require transformation.
4. Write custom logic to process/transform data in the pre/post processors for the models.

24.10.4 Workflow of Data Pre and Post Processors on Client for Object Services



24.11 API signatures for Data Pre and Post Processors for Client Objects

24.11.1 For Visualizer (JavaScript)

24.11.1.1 Registering Processors

Registering Data Pre and Post Processors for Models in Object Services

```
<model>.registerProcessors
```

Models expose the `registerProcessors` API which allows registration of processors for the model object. This step should be performed at least once for each model.

24.11.1.2 Signature

```
<model>.registerProcessors(options, successCallback,
failureCallback);
```

24.11.1.3 Parameters

Parameter	Type	Description	Required
successCallback	Function	The function is invoked on successful registration.	Yes
failureCallback	Function	The function is invoked on failure to register.	Yes
options	JSON	See "Register Processor Options" for supported options.	Yes

24.11.1.4 Register Processor Options

Parameter	Type	Description	Required
preProcessor	Function	A callback function that is invoked before saving field details into the model instance. See "Pre/Post Processor Definition" syntax and details.	No
postProcessor	Function	A callback function that is invoked before returning a value from the model instance. See "Pre/Post Processor Definition" syntax and details.	No
getFromServer	Boolean	If the **getFromServer** is set to true, refreshes object's metadata from Fabric server. By default, it is set to false and object properties meta-data is fetched from the client device cache.	No

24.11.1.5 Return Type

void

24.11.1.6 Pre/Post Processor Definition

This section details the definition of Data pre and post-processor callbacks.

Processors should be valid function reference. A processor should accept two arguments - **value** and **context**.

- **value**: value of the model object property to be transformed. Value is of type Object.
- **context**: model object property specific context. context is a valid JSON. A context can contain the following properties:

> ***Note:*** Each of the registered pre/post data processors callback function must return the transformed value, failure to do so results in undefined behavior.

Properties	Description
datatype	string type, denotes the datatype of object property as defined in Object service in Kony Fabric
object	string type, contains the name of the model object
objectService	string type, contains the name of the object service which contains above object
field	string type. contains the name of the object property/field
metadata	JSON type, field associated metadata as defined in Kony Fabric object service app model.

Note: Registered pre/post processor callbacks are not invoked while updating or reading those fields for which metadata is not defined in an object service of an app in Kony Fabric.

Note: A single callback function can be used as both pre and post processor callbacks.

24.11.2 Creating an Object service with meta-data and enabling Data Pre and Post processors on a Client app

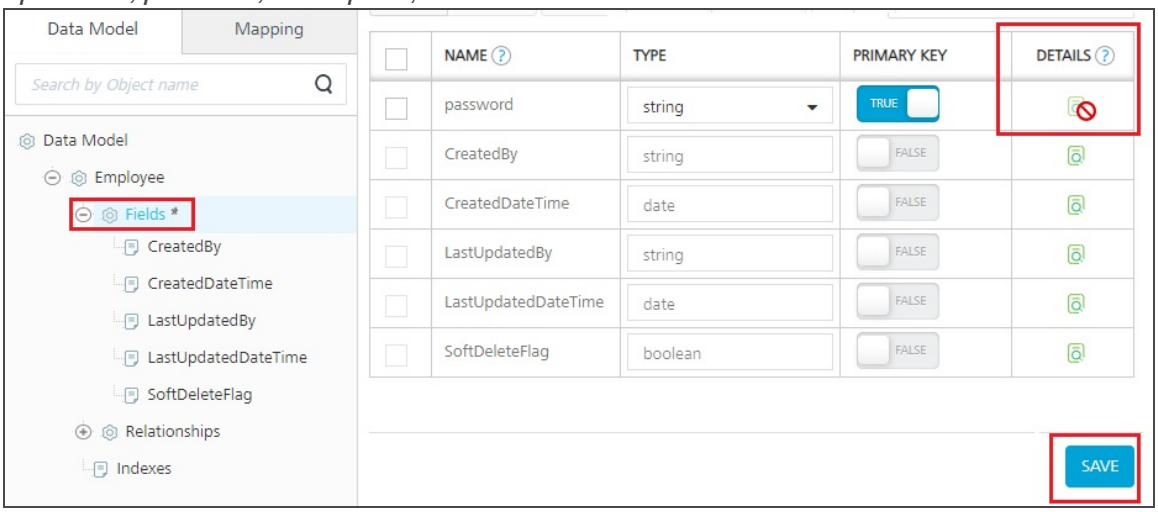
You need to have an Object service with required fields enabled with meta-data in Kony Fabric. The following procedure explains you how to configure **meta-data** for fields in object services and enable **Data pre and post processors** based on meta-data for a client app.

1. Go to Kony Visualizer Enterprise.
2. Create a project based on Kony Architecture Reference.
    - For example, from the **Project** menu, navigate to **New Project** > **Create Custom App**.

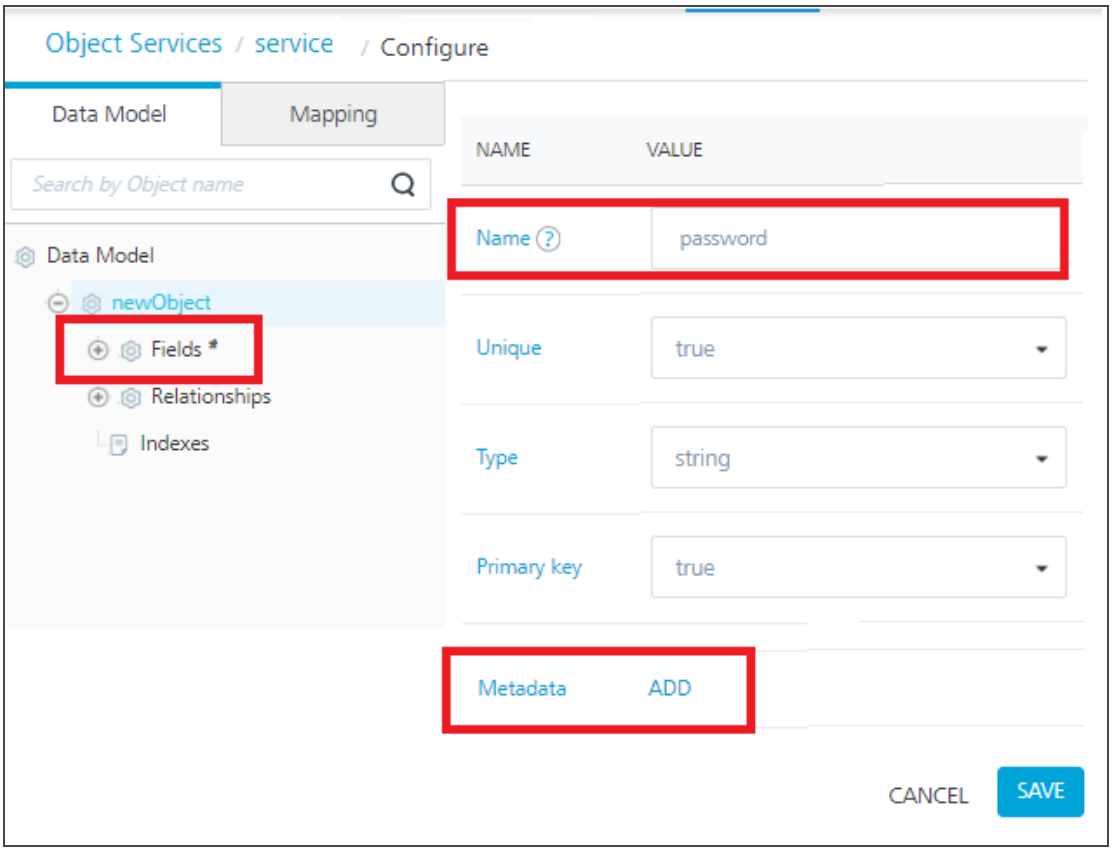
3. Connect to your Kony Fabric account.
4. Configure an object and add fields:
    a. Create an object service.
    b. Create an object.
    c. Navigate to the object > **Fields**.
    d. Click **Add** to create a field. Specify the field name and data type and primary key, as required. Add fields in the object.
    e. Click **SAVE** to save the field.

    > ***Important:*** The **View Details** button is active only after you save the field details.

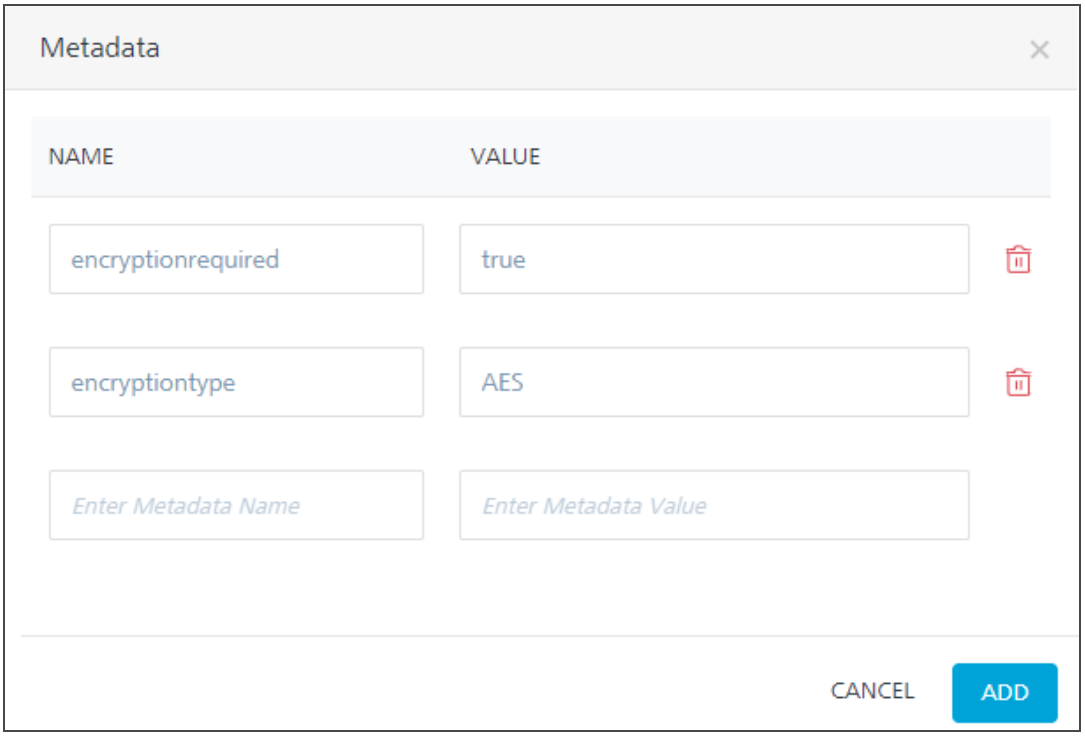
    f. Click the **View Details** button under the **DETAILS** column. The field details page displays the additional columns including *unique, nullable, max length, autogenerated, creatable, updatable, precision, description*, and *metadata*.



g. In the **Metadata** column, click **ADD**.



The Metadata dialog appears.



> ***Note:*** You must enable sensitive fields with **meta-data** in object services to use the **Data Pre and Post Processors for Models** functionality.

h. Specify name and value meta-data for the selected field.

i. Click **ADD**.

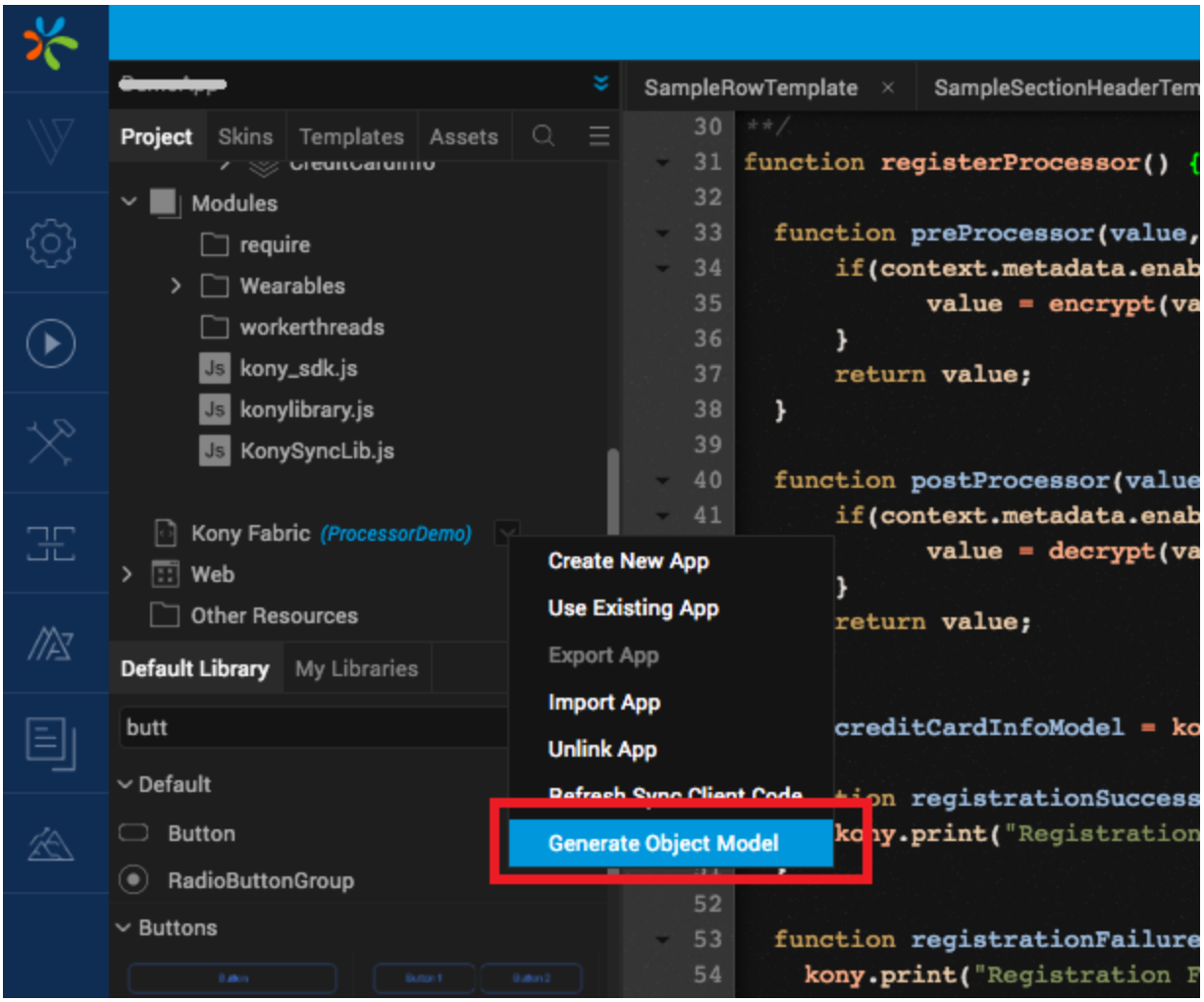
j. Publish Fabric app.

5. Generate the object model as follows:

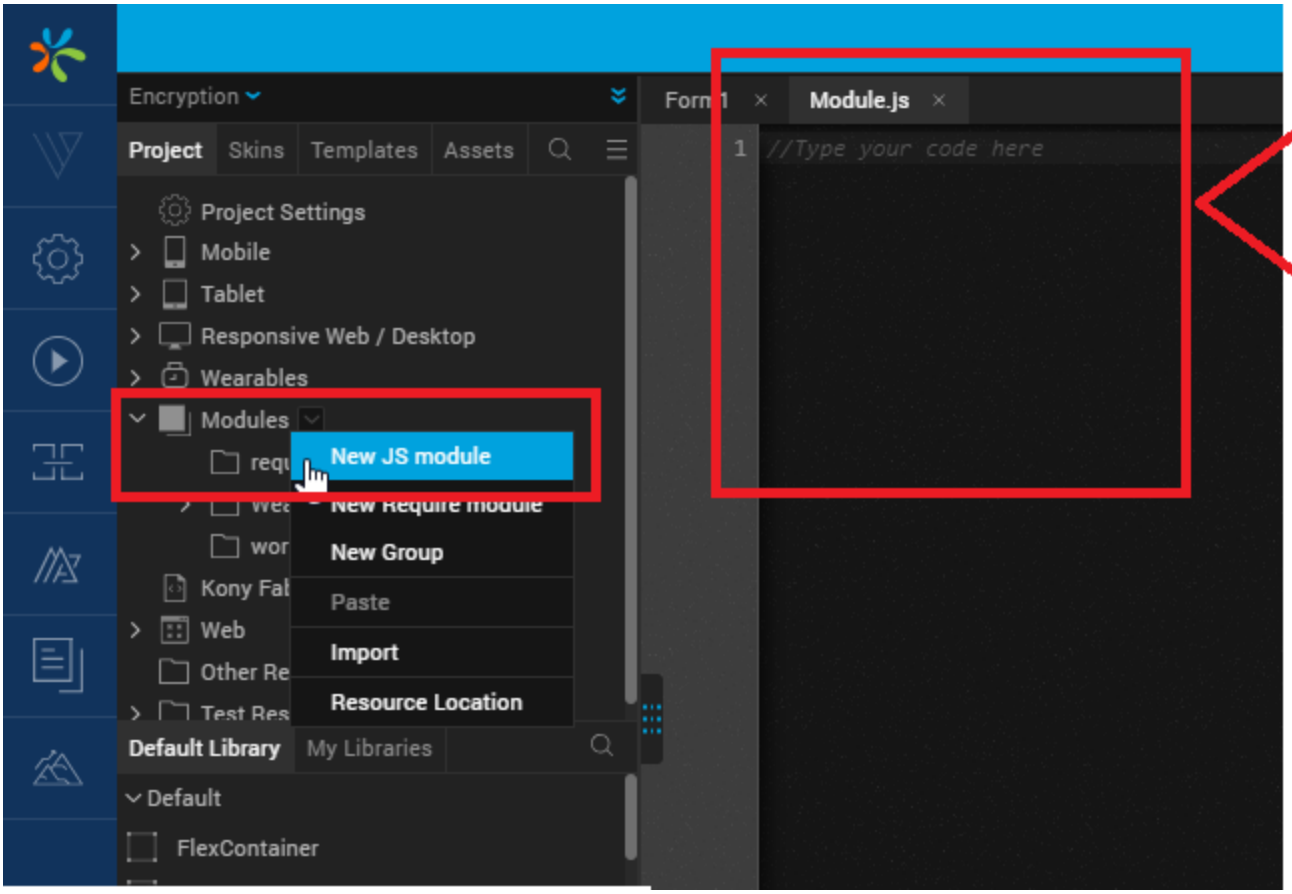
   a. Navigate to the project in Visualizer.

   b. Navigate to the **Project** menu, and right-click Kony Fabric app, and select **Generate Object Model**. Now, the object service is enabled for **Data Pre and Post Processors for Models** registration. You can now write your logic to be applied for selected fields in the

data model.



6. Create a JS module and write your custom code as follows:
    a. Navigate to the **Project** menu, and right-click Kony Fabric app, and select **New JS module**. A new JS module is created, and the **Code** editor become active.



b. In the **Code** editor, specify your custom JS code as follows:

i. Specify your code for [Data Pre and Post Processors](#).

```
function preProcessor(value, context) {
    value = encrypt(value); //transformation logic to be
applied before sending over the network.
    return value;
}

function postProcessor(value, context) {
    value = decrypt(value); //transformation logic to be
applied before returning value to user.
```

```
    return value;
}
```

ii. Register fields in objects with the Data Pre and Post Processors for the client app.

After you register callbacks as pre and post processors for objects, the specified callbacks are automatically invoked for every CRUD operations on that object, as shown in the following sample code snippet.

```
 var objModelDef=
kony.mvc.MDAApplication.getSharedInstance
().modelStore.getModelDefinition("ObjectName");

function registrationSuccess() {
    alert("Registration Success");
}

function registrationFailure(err) {
  alert("Registration Failure" + JSON.stringify(err));
}

var options = {'preProcessor' : preProcessor,
"postProcessor" : postProcessor };
objModelDef.registerProcessors(options,
registrationSuccess, registrationFailure);
```

7. Save and build the project.

So now you have enabled security to the sensitive fields in objects in your client app.

24.11.3 Sample Code to Encrypt/Decrypt Meta-data of the Password field for Client App

The following sample code demonstrates how to encrypt and decrypt a value for the field password over the network call on a client app.

You have created an object service as follows:

- Object service name: **Organization**
- Object: **Employee**
- Field: **password**
- Meta-data: **enabled**

Write the logic/code for post data processor for that value password.

Registering Processors

```
function preProcessor(value, context) {
   //transformation logic to be applied before sending over the
network.
    return value;
}

function postProcessor(value, context) {
    //transformation logic to be applied before returning value to
user.
    return value;
}
```

i. PreProcessor performs encryption for fields for which metadata is defined. In this implementation, it performs encryption for the password before sending data over network.

Below snippet shows implementation of PreProcessor for the Employee.

```
 var AESKey = "ABCDEFGHIJKLMNOP";
var algo = "aes";
var prptobj = {padding:"pkcs5", mode:"ecb"};

/**
** returns Base64 encoded cipherText using AES 128 algorithm.
** AES is symmetric algorithm
*/
function encrypt(plainText) {
  //pass key in wordArray format.
  var key = CryptoJS.enc.Utf8.parse(AESKey);
  var myEncryptedText = kony.crypto.encrypt(algo, key, plainText,
prptobj);
  return (JSON.parse(myEncryptedText).ct);
}

function preProcessor(value, context) {
      if(context.metadata.enableEncryption === "true" &&
context.metadata.encryptionAlgorithm === "AES") {
            value = encrypt(value);
      }
      return value;
}
```

ii. PostProcessor performs decryption for fields for which metadata is defined. In this implementation, it performs decryption for the password before sending data over network.

Below snippet shows implementation of PostProcessor for the Employee.

```
 var AESKey = "ABCDEFGHIJKLMNOP";
var algo = "aes";
var prptobj = {padding:"pkcs5", mode:"ecb"};
```

```
/**
** returns plainText from encrypted using AES 128 algorithm
** AES is symmetric algorithm
*/
function decrypt(cipherText) {
  //pass key in wordArray format.
  var key = CryptoJS.enc.Utf8.parse(AESKey);
  var stringifyCipher = JSON.stringify({'ct':cipherText});
  var myClearText = kony.crypto.decrypt(algo, key,
stringifyCipher, prptobj);
  return myClearText;
}

function postProcessor(value, context) {
      if(context.metadata.enableEncryption === "true" &&
context.metadata.encryptionAlgorithm === "AES") {
            value = decrypt(value);
      }
      return value;
}
```

Sample Code for Context

The following sample JSON code snippet demonstrates for `context`, for which context is defined for the password property of Employee model object in Organization object service:

```
context = {
    "datatype": "string",
    "object": "Employee",
    "objectService": "Organization",
```

```
    "field": "password",
    "metadata": {"enableEncryption": true, "encryptionAlgorithm":
"AES", "encryptionKey": "XXXXXX"}
}
```

24.11.4 Advanced Features in Data Pre and Post Processors for Client App

De-registering Processors

To de-register the processors, use null/undefined as callback function, as shown in the following sample code snippet.

```
var employeeModel = kony.mvc.MDAApplication.getSharedInstance
().modelStore.getModelDefinition("Employee");

function registrationSuccess() {
    alert("Registration Success");
}

function registrationFailure(err) {
  alert("Registration Failure" + JSON.stringify(err));
}

//unregister preProcessor.
var options = {'preProcessor' : null};

employeeModel.registerProcessors(options, registrationSuccess,
registrationFailure);
```

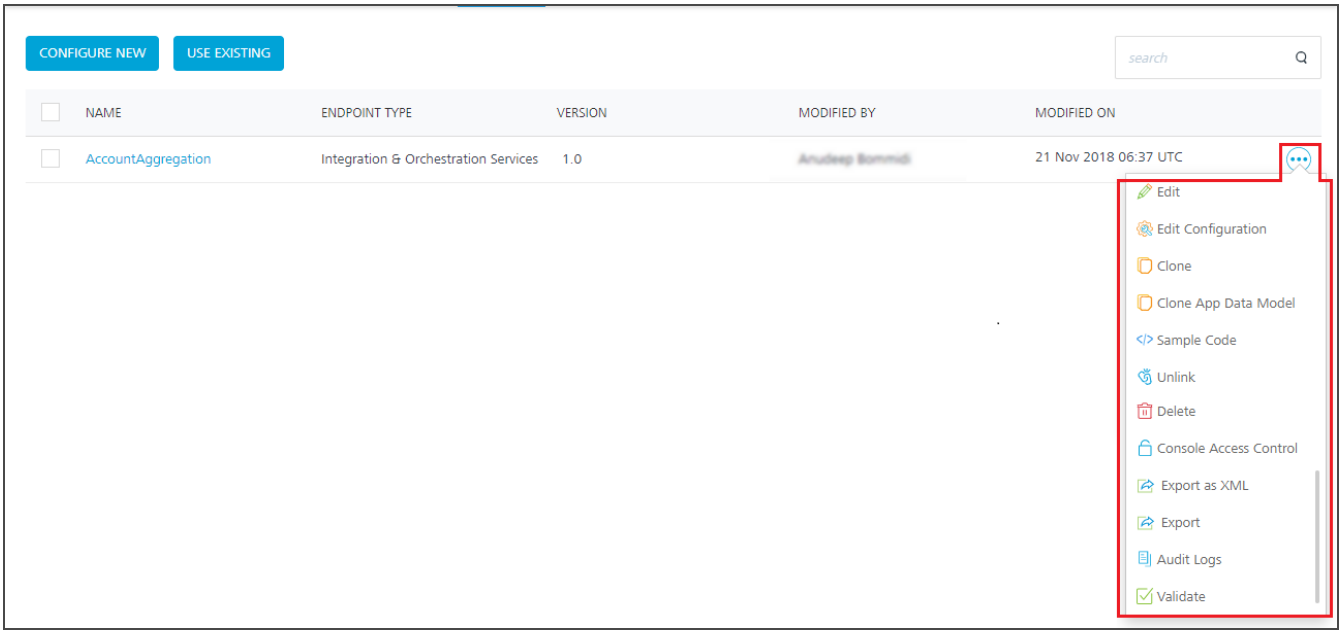
Refreshing metadata in Processors

Use the option `getFromServer` while registering the processors to refresh meta-data associated with object properties from the server. By default the `getFromServer` is set to false and cached meta-data is used.

```
function preProcessor(value, context) {
    value = encrypt(value); //transformation logic to be applied
before saving value to database or sending to network.
    return value;
}

function postProcessor(value, context) {
    value = decrypt(value); //transformation logic to be applied
before returning value to user.
    return value;
}

var employeeModel = kony.mvc.MDAApplication.getSharedInstance
().modelStore.getModelDefinition("Employee");
function registrationSuccess() {
    alert("Registration Success");
}

function registrationFailure(err) {
  alert("Registration Failure" + JSON.stringify(err));
}

// Refresh metadata from server
var options = {'preProcessor' : preProcessor, "postProcessor" :
postProcessor, "getFromServer": true };

employeeModel.registerProcessors(options, registrationSuccess,
registrationFailure);
```

24.12 Context Based Options

To perform various actions on an existing service, click the contextual menu of the required service.



The contextual menu contains the following options:

- **Edit**: Allows you to configure parameters of an object service.
- **Edit Configuration**: Allows you to configure the Data Model and Mapping of an object service. After you edit a service, you have to republish all the apps that are using the service to apply the changes.

> ***Note:*** To know more about publishing an app, refer to Publish an app.

> ***Note:*** If a service is part of a published app, you can rename that service only after the app is unpublished.

- **Clone**: Allows you to duplicate an existing service. Changes made to a cloned service do not impact the original service.
- **Clone App Data Model**: Allows you to duplicate only the data model of an existing object service. This option does not duplicate the verbs. Changes made to a cloned app data model service do not impact the original service.
- **Sample Code**: A dynamic code is generated based on the configuration of a service. You can use this code in your SDK.
- **Unlink**: Allows you remove the service from the **Objects** tab of an app. When a service is unlinked, it is disassociated from a particular app.

  > ***Note:*** If you want to use an unlinked service, select the service from the **Use Existing Objects Service** dialog box.

- **Delete**: Allows you to delete a service.

  > ***Note:*** If a service is a part of a published app, you can delete that service only after you unlink the service from all the published apps.

- **Console Access Control**: Controls the access to the applications and services of apps.
- **Export as XML**: Allows you to export the Object Service to your local system. The exported file is of .xml format.
- **Export**: Allows you to export the Object Service to your local system. The exported file is of .ZIP format.
- **Audit Logs** helps you to capture all the user activities performed in a service. **Object Name**, **Object Type** and **Modified On** fields are prepopulated with the **Service Name**, **Services**, and **Last 7 Days** respectively.

  For more information on Audit Logs, refer to Audit Logs documentation.

- **Validate**: Allows you to validate the Object Service.

24.13 Enhanced Identity Filters - Objects

Identity filters are an enhanced data-filtering mechanism available in Storage Services and Service-Driven Objects. For Service-Driven Objects, enhanced identity filters are configured on the underlying integration/orchestration services. For more details, refer to Enhanced Identity Filters - Integration Services. You can use identity filters to filter data for a mobile app based on dynamic fields returned from an identity provider.

When a user logs on to a mobile app, the logon can invoke a custom Kony Fabric Identity Service. The response to the back-end logon contains identity or security attributes for the logged-on user. For example, a user_id field. The storage service can use the identity or security attributes from the logon response as input parameters for operations, such as get, create, and update.

Updating a Storage Object from a User Profile

A key advantage of identity filters is the ability to update certain parts of a storage object from the Identity Service and not the mobile app. With a storage object, the mobile app cannot update certain fields or elements of the object. For example, a field that identifies who created a record (CreatedBy) or who updated a record (LastUpdatedBy).

By using an identity filter in a storage service, the userID from the identity profile is mapped to the CreatedBy field as an input parameter. When the mobile app user creates a record, Kony Fabric sets the CreatedBy value from the identity provider and not the mobile app. The mobile app user cannot set the CreatedBy field because the mobile app does not include the CreatedBy field for data entry.

Constrain Data with an Identity Filter that the User Cannot Modify

Another key advantage of using identity filters to query data is that the mobile app user cannot access and modify these filters. For example, an employee that uses the mobile app can view and access only the records that he/she created, while a manager can access the records that all employees created. Kony Fabric maps a user attribute from the user profile to an input parameter for the storage service query. The mobile app user cannot modify the identity filter because the query was executed on the Kony Fabric storage service, and not on the mobile app.

24.13.1 Use Case: Auto Dealership Sales Manager

Bill is the sales manager of an auto dealership that sells Steed brand cars. Bill uses a mobile app that is provided by Steed. Bill uses the mobile app to get sales reports, learn about special internal sales programs, and look up invoice prices for his dealership.

The sales manager has a user name that he uses to log on to the account. Steed maintains an internal dealerId in Bill's user profile that identifies the dealership that Bill is associated with. The dealerId is sensitive information and a sales manager is not privy to and cannot see his internal dealerId. All the back-end APIs need this dealerID as an input parameter during any subsequent integration queries.

When Bill logs on to the mobile app, the response to the logon from the back end contains Bill's internal dealerId. The Kony Fabric Identity Service stores this identity attribute for the session. In the storage service, the get operation uses this identity attribute as an input parameter to filter the request for data that is sent to the mobile app. For example, when Bill looks up manufacturer dealer incentives or invoice costs, the storage service adds Bill's dealerId to the query. The query returns only dealer incentives and invoice costs for Bill's dealership. At no point does Kony Fabric send Bill's dealerId to the mobile app. This eliminates the possibility that Bill, or anyone else that gains access to Bill's mobile device, can use the dealerId to access incentives or invoice costs for other dealerships or breach company data.

24.13.2 Use Case: Banking App

Oakway National Bank (ONB) has a mobile banking app. The mobile app uses a custom SOAP Identity Service configured on Kony Fabric. When a customer of ONB signs in to the app, the sign-in invokes the identity service. The response to the back-end sign-in contains a user security attribute, for example, CRM_Id.

After the user signs in successfully, every subsequent integration request to the back end must have the CRM_Id as an input parameter to identify the authorized user who is accessing the private bank accounts. A user's CRM_Id is sensitive and is not shared with the user. A hacker could use a CRM_Id to illegally access a user's personal financial information. By using an identity filter with the storage service, the CRM_Id is preserved on Kony Fabric itself and is not sent to the user's mobile app. This reduces the possibility of an intruder gaining access to sensitive data on the mobile device.

24.13.3 How to Configure Identity Filters

You can configure identity filters on the operations (verbs) of an object in a Storage Service. For Get operations, you can configure identity filters on the Security Filters tab. For the Create, Update, and Delete operations, you can configure identity filters on the Request Input Parameters tab. To configure identity filters for a storage service, you must link the Storage Service to a Kony Fabric Identity Service.

To link a storage service to an Identity Service, do the following:

1. In Kony Fabric, click the **Objects** service tab.

   A list of object services appears. This is the list of object services that have been created for the Kony Fabric app.

2. Select a storage service.

   A storage service is an object service that you have configured to use storage on the Kony Fabric back end. When you create a storage service, you select Storage as the endpoint type.

3. Click in the **Identity Service for Backend Token** field, and select the identity provider that you want to use.

4. Click **Save**.

Configuring a Security Filter

To configure a security filter for a Get operation in a storage service, do the following:

1. In Kony Fabric, click the **Objects** service tab.

   A list of object services appears. This is the list of object services that have been created for the Kony Fabric app.

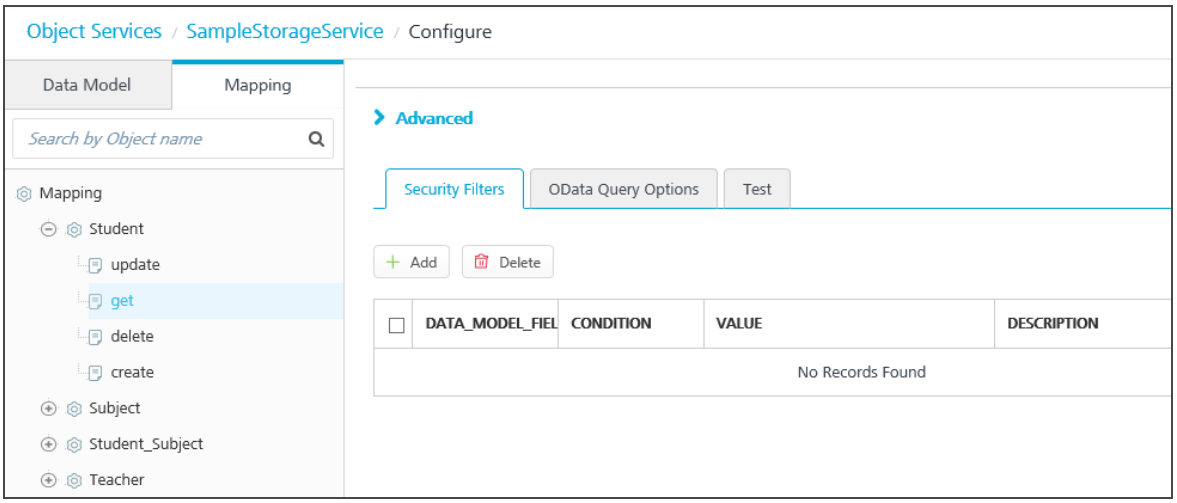
2. Select a storage service.

3. On the **Mapping** tab of the navigation pane, click the plus button next to an object.

The list of operations, or verbs, for the object appears.

4. Click the **GET** verb.

   The configure screen for the get verb appears.



5. Click the **Add** button.
6. In the **Data_Model_Field** column, click the drop-down menu, and select a field. For example, **LastUpdatedBy**.

   Select a field from the object that you want to use to filter the data returned by the query.

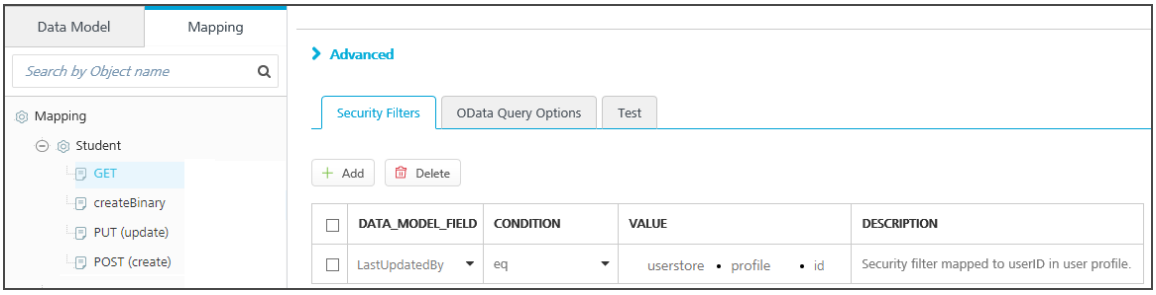
7. In the **Condition** column, click the drop-down menu, and select **eq**.

   The available conditions are eq (equals), gt (greater than), and lt (less than).

8. In the **Value** column, click the text box and enter **profile.user.id**. When you start editing the field, dependent identity services are auto populated. Select the identity service you want.

   This maps the user ID attribute from the identity service to the Id field in the storage object.

9. In the **Description** column, enter a description of the security filter.



10. Click **Save**.

    This security filter is added to any **get** operations performed on the Student object.

The security filter constrains the results of a get operation that are returned to the mobile app. For example, if the mobile app has an option to get a list of student records that the user had recently updated, Kony Fabric adds this security filter to the query. The mobile app retrieves only the student records that were updated by the logged on user. The field that Kony Fabric used to filter the results, LastUpdatedBy, is added to the query only in Kony Fabric and is not sent to mobile app with the results. The mobile app user cannot access this field and use it to retrieve records that were updated by another user.

Configure Identity Filters

To configure identity filters for a create operation in a storage service, do the following:

1. In the navigation pane, click the **Mapping** tab, and then click the plus button next to the an object. For example, Teacher.
2. Click the **create** verb.

   The Request Input Parameters tab appears in the Configure screen. You define how to use the user profile data by selecting a verb and the data model field that you want to map to the user profile data.
3. Click **Add**.
4. In the **Data_Model_Field** column, click the drop-down menu, and select the **CreatedBy** attribute.

5. Click the drop-down menu in the **Value** column, and select **identity**.

   An identity value indicates that Kony Fabric will retrieve the value specified from the user's security profile in the identity service that is linked to the object service.

6. In the text box next to the **identity** value, enter **profile.user.id**.

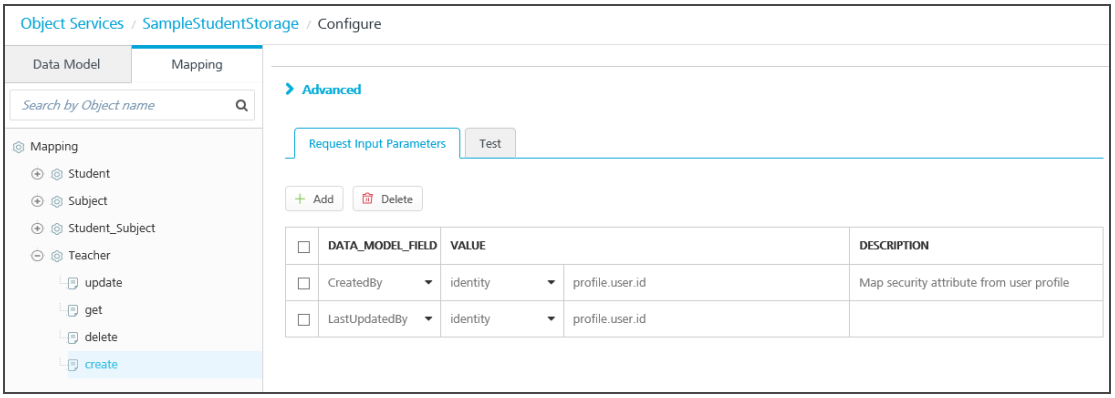
   This specifies that Kony Fabric will populate the value of the input parameter from the user's security profile.

7. Click **Add**.

8. In the **Data_Model_Field** column, click the drop-down menu and select the **LastUpdatedBy** attribute.

9. Click the drop-down menu in the **Value** column, and select **identity**.

10. In the text box next to the **identity** value, enter **profile.user.id**.



11. Click **Save**, and then in the navigation pane, click the **update** operation.

12. Click **Add**.

13. In the **Data_Model_Field** column, click the drop-down menu in the **Data_Model_Field** column, and select the **LastUpdatedBy** attribute.

14. Click the drop-down menu in the **Value** column, and select **identity**.

15. In the text box next to the **identity** value, enter **profile.user_id.**

16. Click **Save**.
17. In the navigation pane, under the Teacher object, click the **update** operation.
18. Click **Add**.
19. In the **Data_Model_Field** column, click the drop-down menu, and select the **LastUpdatedBy** attribute.
20. Click the drop-down menu in the **Value** column, and select **identity**.
21. In the text box next to the **identity** value, enter **profile.user_id**.
22. Click **Save**.

> *Important:* Object services enabled with security filters cannot be tested from Kony Fabric Console.

24.13.3.1 Preprocessor and Postprocessor References

An identity session is read-only and cannot be modified in the preprocessor and postprocessor, or an integration service. Once the context is accessible, a Kony Fabric developer can refer to the identity session context.

24.13.3.2 Support Identity Provider Types

The following identity providers are not supported for Storage Services and identity filters:

- Microsoft Active Directory
- Open LDAP
- Active Directory Federation Services (ADFS) over SAML
- Azure SAML.

24.14 How to Create and Store Sample Data to Storage Object Services

The Kony Storage service provides you the ability to create your own database schema/ data tables on the relational database hosted on Kony Cloud by using the storage endpoint connector under Object services. Storage service helps you to develop apps by creating an object data model and uploading data to the storage database which will then be available to app developers on the runtime instance.

You can store or import sample data to storage object services by importing a .zip file. The sample data can be uploaded at the design time of service in Fabric console which gets imported to storage service during publish.

Alternately, sample data can also be imported at run time in the App Services console.

You can create your data model for your object service endpoint type Storage by either of the two methods:

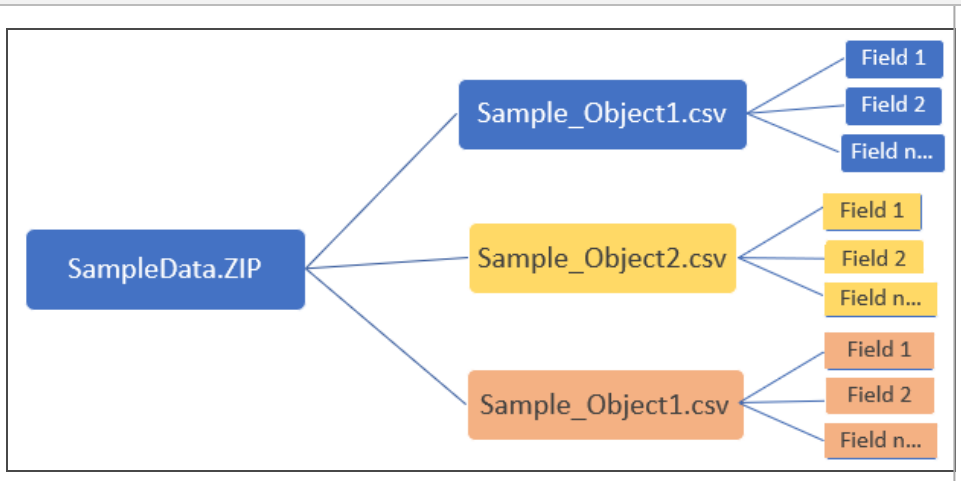
Steps	Method 1	Method 2
Step 1 - Creating Data Model	**By creating objects and fields manually** a. Create objects and fields in the Data Model pane. The data model is created.	**By uploading an excel (workbook) file** a. Upload the data model in one of these file formats: XLS and XLSX. (Each Tab/worksheet names of an XLSX file are used for object names in the data model. Column names in the first row of each tab/worksheet are used for field names in an object. The data model is created.

Steps	Method 1	Method 2
Step 2 - Adding Data to the Fields in Objects, for data model.	b. Import the data model template. A sample template `.ZIP` file contains multiple objects as separate `.CSV` files. c. Unzip the file downloaded template, add data to the fields in .CSV files, and save them. d. Zip the .CSV files. The zipped file name should be the original name that you have downloaded. e. Finally, upload the zip file to the storage service and publish the service.	

> ***Note:*** For more details on how to create a data model for an object service type storage, refer to Generating Sample Data Template for Storage Object Services based on a Data Model section.

> ***Important:*** If you associate sample data while creating the service in Kony Fabric, the sample data gets attached to the object service. While publishing the service, the sample data gets stored to the storage database and is available to the client app as well.
>
> If you associate sample data at the run-time server, the sample data gets stored in the storage database. The behavior is the same when data is associated/modified from the Runtime Server.

Structure of Sample Data .ZIP File for Storage Objects

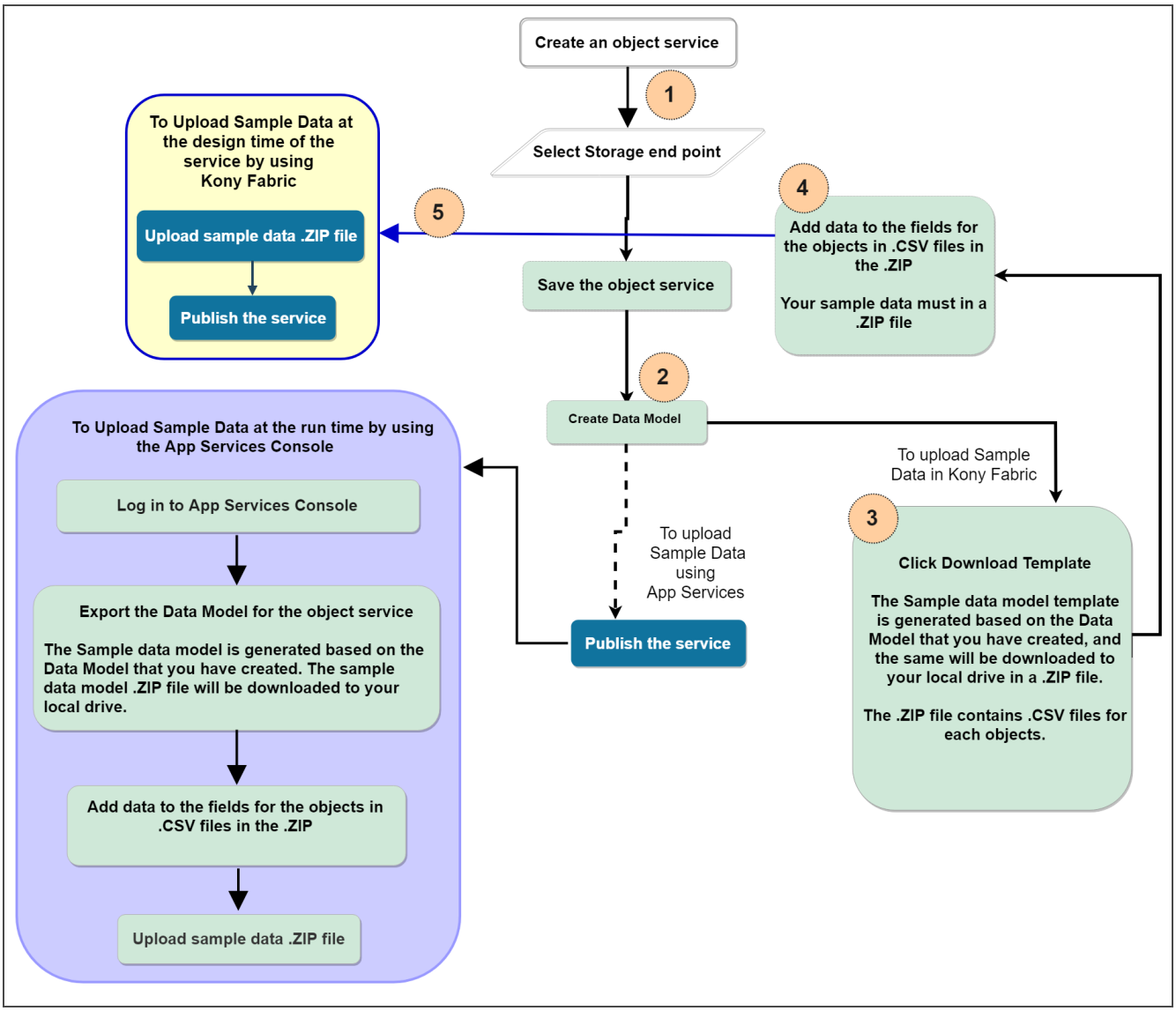


- A sample data `.ZIP` file contains multiple objects as separate `.CSV` files.
- A `.CSV` file must contain fields of an object as defined in the data model.
- Users can add data in the correspondin g fields in a `.CSV` file.

Benefits of Storing Sample Data to Storage Object Services in Kony Fabric

- You can associate sample data while creating the storage service in Kony Fabric, The sample data gets attached to the object service. By doing this, you can save time by uploading the sample data to storage object services in the run-time server.
- Other users can reuse the service with sample data.
- While publishing the app with a storage service, the sample data gets published to the storage database. So now, Users can access their application with sample data on the first launch itself.

24.14.1 Workflow for Generating Sample Data Template and Attaching Sample Data for Storage Object Services



24.14.2 Generating Sample Data Template for Storage Object Services based on a Data Model

The following section helps you to generate sample data model with sample data, which you can import to a storage object services while creating the service.

To create an object service for storage with sample data, follow these steps:

1. **Create a Data Model.**

   a. While creating an object service, select **Storage** from the **Endpoint Type** field.

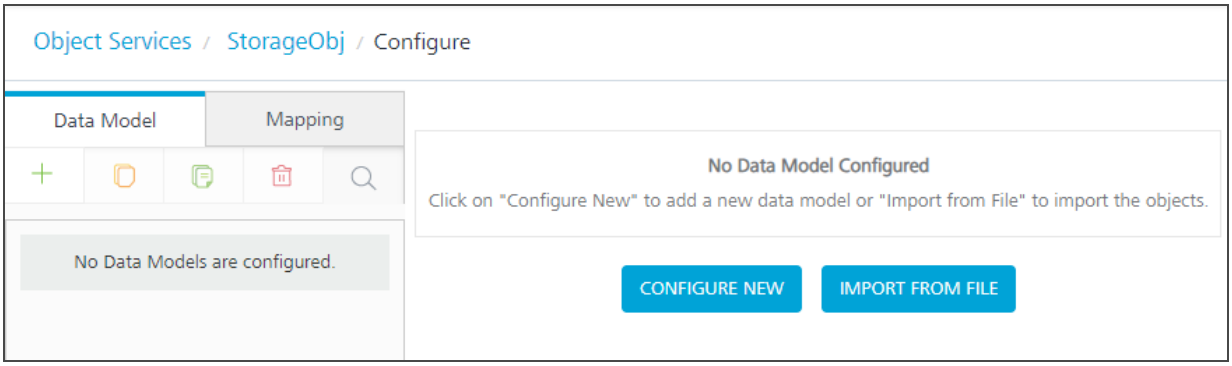
   The **Sample Data** and **Download Template** buttons are displayed.

   > ***Note:*** The **Delete Strategy** feature in Storage services helps you to enable the **Delete** operation to delete the selected data from the database. Refer to **Delete Strategy**.

   b. Click **Save and Configure**. The **Data Model** configure screen for the object service appears.

   c. **Create a data model as follows**:

   You can create your data model or upload it by using an excel file.



   - **Configuring a new data model:**

     a. Click **CONFIGURE NEW**. An object is created with the default name. You can change the name of the object under the **Name** text box and description.

     b. Click **SAVE**.

c. In the **Data Model** tab of the navigation pane, click the plus button next to the object.

   Fields and Relationships nodes appear under the selected object.

d. Click **Fields**.

   The metadata details of the field is created by default and appears the same in the Configure screen.

e. Add fields to the object based on your requirements.
   For more information on how to add fields, refer to steps in the Creating Objects' Definition and Map to Back-end Objects Manually section.

f. Save the object. You can create more objects, if required.

   > ***Important:*** Primary Key is mandatory for a data model. Please specify at least one field as a primary key in the data model.

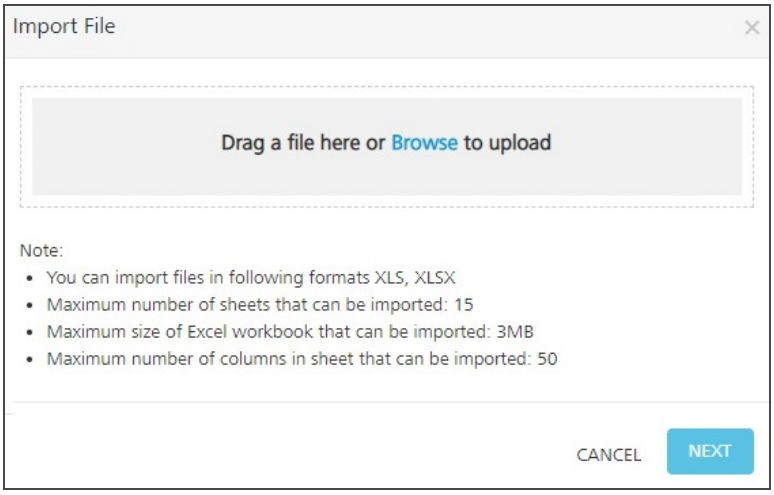
g. Click **SAVE** to save the changes to the data model. Now you have created the data model.

- Importing a data model:

  You can import data model in these file formats XLS and XLSX.
  For more information on requirements to create a valid data model using an XLSX file format, refer to Data Model Template in XLSX file.

  a. Click **IMPORT FROM FILE**.

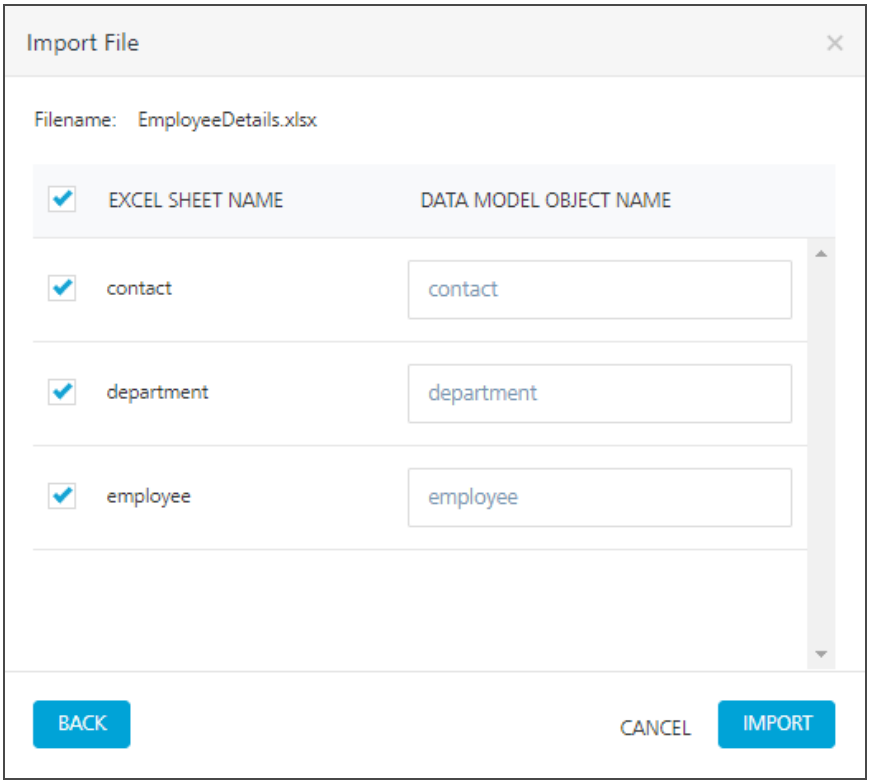
  b. In the **Import File** dialog box, upload the data model by drag-and-drop the valid XLS/XLSX file.



c. After the file is uploaded successfully, click **NEXT**.

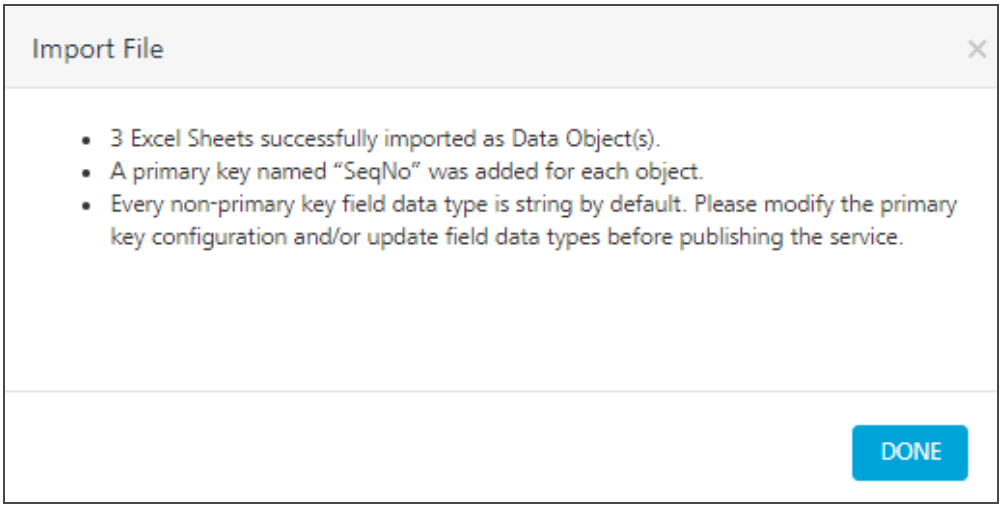
In the next screen, the objects are created under the **DATA MODEL OBJECT NAME** based on the **EXCEL SHEET NAME** column details uploaded from the excel file. These objects are selected by default.

- You can modify an object name, if required.
- You can clear the check box for an object to not to be included it in the data model, if required.



d. Click **IMPORT**.

The **Import File** dialog box displays the additional information such as: number of objects imported, primary key field name, and default data types for the fields.



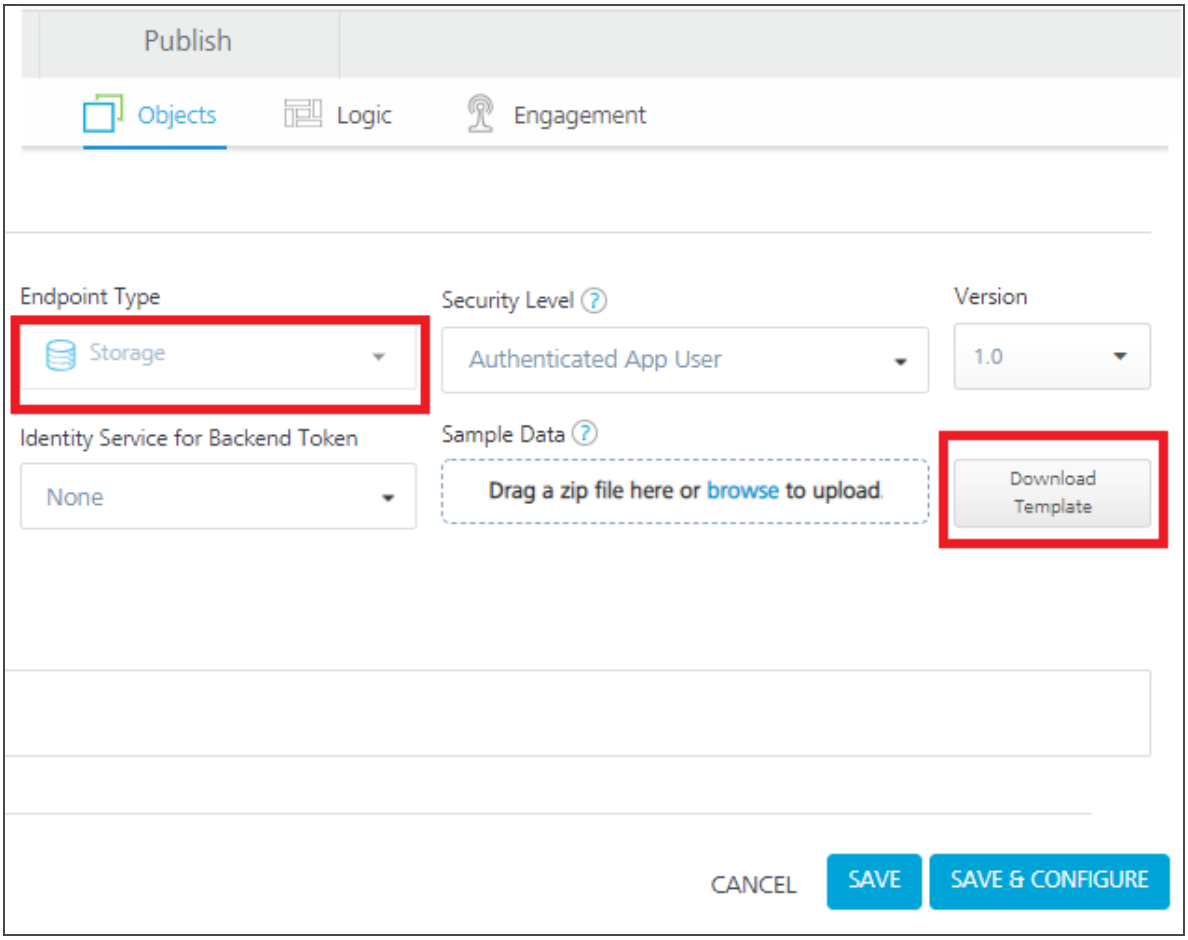
e. Click **DONE**.

> ***Important:*** Primary Key is mandatory for an object. By default, a primary key named **SeqNo** is added for each object. You can modify the primary key configuration field as per your requirements.

> ***Important:*** Every non-primary key field data type is string by default. You must update field data types before publishing the service.

Now you have created the objects in the data model.

2. **Generate Sample Data template** based on the data model as follows:
    a. **Downloading the Template:**
        i. Go to the **Objects Service Definition** page.



        ii. Click the **Download Template** button to auto-generate the **Template** based on the data model. The sample data template will be downloaded in a .ZIP file into your local system.

> ***Note:*** You can add more data to the existing .CSV files in the .ZIP

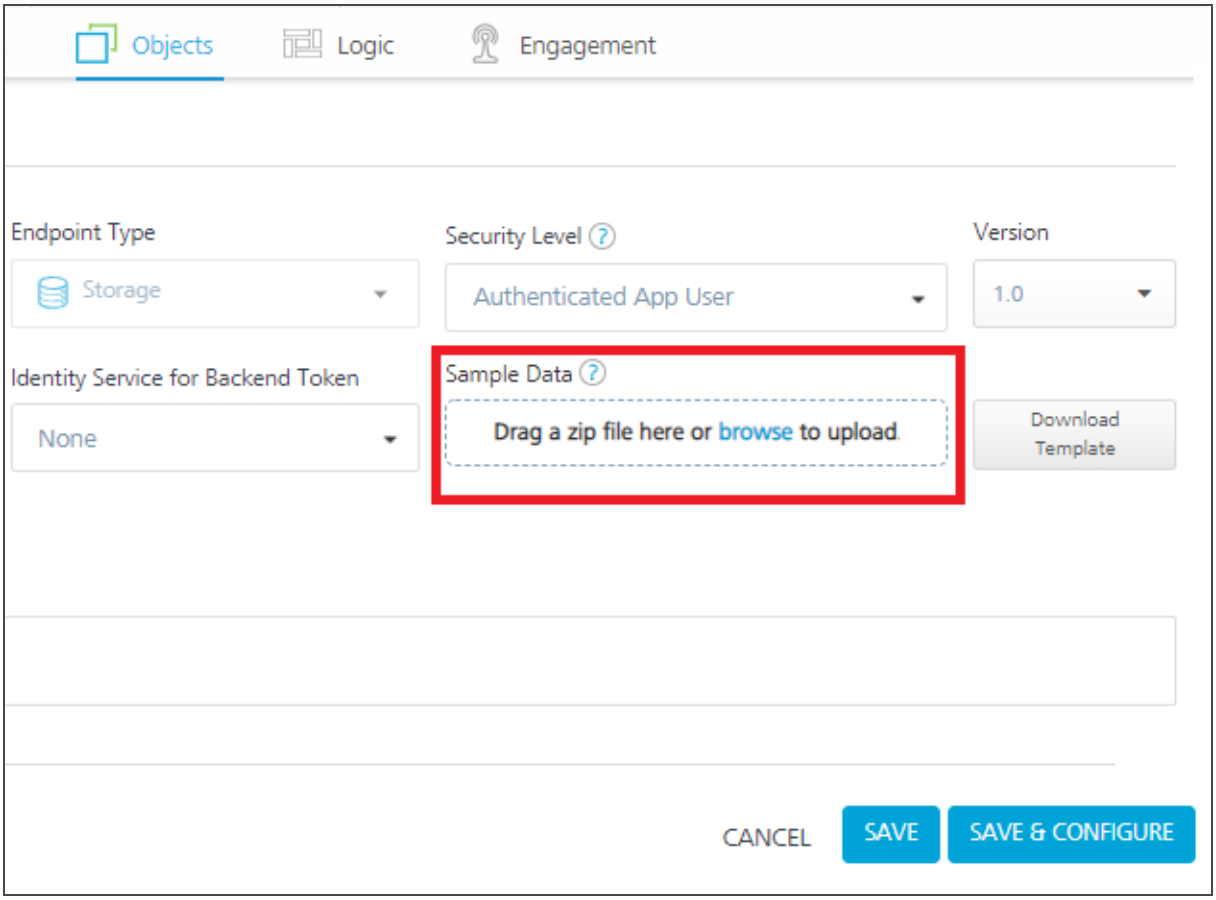
Now you have created the sample data template in a .ZIP file. The sample data template .ZIP file contains .CSV files for each object.

b. **Uploading the Template with sample data:**

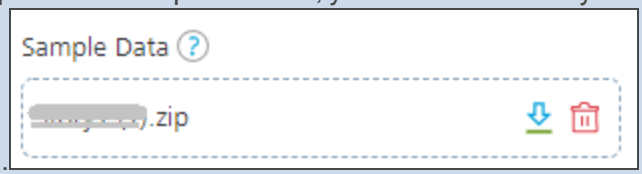
i. Add data to the fields to your sample data objects in .CSV files in the template .ZIP file.

ii. Zip all the updated the .CSV files.

iii. Upload the zip file by drag-and-drop to the **Sample Data**. In the **Sample Data > Drag a zip file here or browse to upload**, drag-and-drop the zip file. The sample data file is uploaded to the service.



> ***Note:*** You can upload the sample data .ZIP file by clicking the **browse** button.

> ***Note:*** After you upload the sample data file, you can download by click the **Download** button.
>
> 
>
>
> To delete the uploaded sample data .zip file, click the **Delete** button.
>
> Refer How to Manage Sample Data to Storage Object Service using Kony Fabric

iv. Click **SAVE** to save the service.

3. **Publish the Sample Data with the App.**
   - To publish the sample data for the first time, do the following:
     a. In the **Service and Web Client** tab, select the Environment.
     b. Click **Publish**. The sample data will be published along with the app.

        After you publish the app, the app users can access the sample data on devices. You can modify and manage the sample data for Object services in the App Services console.

24.14.3 How to Manage Sample Data to Storage Object Service using Kony Fabric

While publishing the sample data for the first time, the data is auto-published. If you want to publish a new version of sample data, you must select the **Include Sample Data** in Publish Reconfiguration.

Data in the sample set act as an addition to the existing data. If there is a conflict, existing data will be overwritten with the new sample data.

1. Upload the new Sample Data .zip file to your Object. Refer Generating Sample Data for Storage Object Services based on a Data Model
2. Click the **Publish** tab.
3. In the **Service and Web Client** tab, select the Environment.
4. Click **CONFIGURE & PUBLISH**. The **Service Reconfigure** page appears.
5. Navigate to the **Object Service Configuration** tab.
6. Select the **Include Sample Data** check box.
7. Click **SAVE & PUBLISH**.

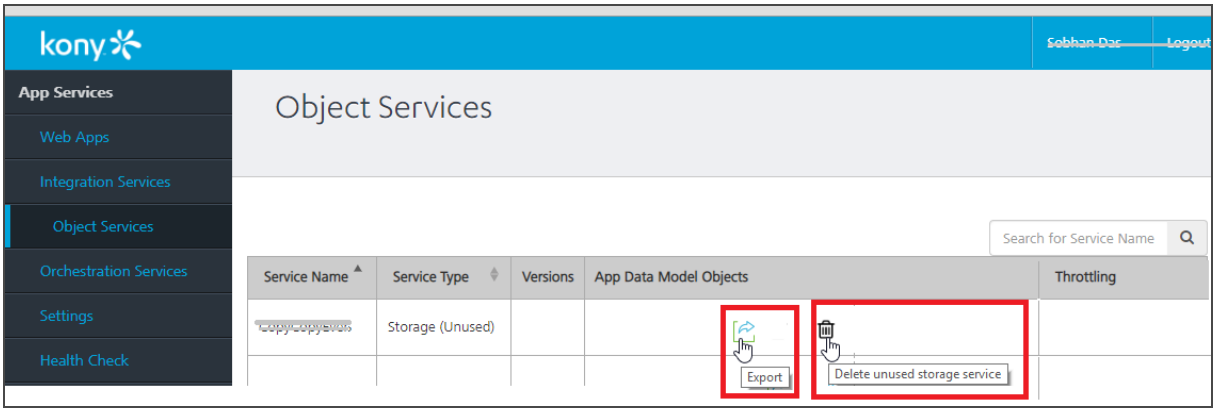
24.14.4 How to Delete Unused Data for Storage Object Services using the App Services Console

Storage object services occupy the space of your environment. You can unpublish unused storage services to gain the space.

When you unpublish an app with a storage object service from Kony Fabric Console, the data that is associated with the storage object service is not deleted. You must clear the data permanently from the database by using the App Services Console.

The following steps helps to delete orphan data for storage object services:

1. Log in to Kony Fabric Console.
2. Navigate to an app that has configured with storage object service with data.
3. Unpublish the app.
4. Log in to App Services Console.
5. Navigate to the Objects Services section. All the unused schemas are stored as `Storage (Unused)`.



Now, you can view the **Delete** and **Export** buttons next to the object service in App Services Console.

6. Click **Delete** to delete the unused data.

> ***Note:*** Ensure that you export the data before you delete it permanently.

- When you click the **Export** button, the existing data is exported in a .CSV file.

24.15 Offline Sync (Offline objects)

An Offline Object is a solution to the data synchronization problem that Apps face when they must work offline. The Offline object is a new synchronization capability built on the top of Kony Fabric Object Services. Using the Offline Objects feature, apps can download data from the Kony Fabric Object Services to a mobile device. Apps can continue to use the downloaded data when the device does not have network connectivity. Data can be synced with the Kony Fabric Object Services back-end, once the device gets the network.

Let us learn more about offline objects in this video. This video gives information about:

- The definition of Offline Objects
- The architecture of Offline Object
- The configuration of the Offline Object Services

Note: For more hands-on approach on how to implement Offline Sync Services, import and preview the **Work Order Management (Offline Enabled)** app on to your Visualizer.

DOWNLOAD THE APP

For more information on Offline objects, refer to the following:

- Offline Object User Guide
- Offline Objects Getting Started Guide
- Offline Objects API Reference Guide
- Webinar Recording on Offline Objects
- Video tutorial on Offline Objects
- Sample App - Offline Enabled WorkOrder in Marketplace
- Offline Objects Windows 10 Support
- Offline Objects in Progressive Web App
- Offline Object Support for Mobile and Desktop Web Apps
- Reports for Object Services
- Comparison of Kony Fabric Sync vs Offline Objects
- How To Use Kony Legacy Sync in On-Premise V8 SP4 and Above

25. Offline Sync (Offline objects)

An Offline Object is a solution to the data synchronization problem that Apps face when they must work offline. The Offline object is a new synchronization capability built on the top of Kony Fabric Object Services. Using the Offline Objects feature, apps can download data from the Kony Fabric Object Services to a mobile device. Apps can continue to use the downloaded data when the device does not have network connectivity. Data can be synced with the Kony Fabric Object Services back-end, once the device gets the network.

Let us learn more about offline objects in this video. This video gives information about:

- The definition of Offline Objects
- The architecture of Offline Object
- The configuration of the Offline Object Services

Note: For more hands-on approach on how to implement Offline Sync Services, import and preview the **Work Order Management (Offline Enabled)** app on to your Visualizer.

DOWNLOAD THE APP

For more information on Offline objects, refer to the following:

- Offline Object User Guide
- Offline Objects Getting Started Guide
- Offline Objects API Reference Guide
- Webinar Recording on Offline Objects
- Video tutorial on Offline Objects
- Sample App - Offline Enabled WorkOrder in Marketplace
- Offline Objects Windows 10 Support
- Offline Objects in Progressive Web App

- Offline Object Support for Mobile and Desktop Web Apps
- Reports for Object Services
- Comparison of Kony Fabric Sync vs Offline Objects
- How To Use Kony Legacy Sync in On-Premise V8 SP4 and Above

26. Legacy Sync

> *Important:* Legacy Sync has been deprecated from Kony Fabric V8 SP4 and support for the same is not available. From Quantum Visualizer V9 onwards, you cannot create a legacy sync based client application. However, the existing legacy sync based apps will continue to work. The new applications that need offline and sync capabilities must use the Offline Objects feature.

Kony **Legacy Sync** is a comprehensive data synchronization platform that enables developers to add synchronization capabilities to mobile applications. Fundamental to Sync Framework is the ability to support offline access for applications, services, and collaboration of data between devices and the backend systems.

To enable synchronization capability for an app, you need to define a Sync Configuration file.

26.1 Sync Configuration file

A Sync Configuration captures details of the data synchronization characteristics of an application. These details are captured in the file typically referred to SyncConfig.xml that adheres to the SyncConfig.xsd schema. A SyncConfig.xml represents the below structure.

The two most important elements of the schema are:

- Sync Scope
- Sync Object

26.1.1 Sync Scope

A Sync Scope groups the Sync Objects that share a common synchronization characteristic like Sync Strategy and synchronization characteristics.

A Sync Configuration can have multiple Sync Scopes. It is not possible to define relationships between Sync Objects that belong to belonging to different Sync Scopes.

26.1.2 Sync Object

You can consider a Sync Object as a business object that has some public attributes and some methods. The public attributes correspond to the fields visible to client devices, and they are used for synchronization. The methods correspond to the CRUD operations that map to the backend services exposed to the object. The parameter values methods /operations based on both public attribute values.

A Sync Object is meta-data:

- Defining the business object model of an application.
- Defining the way data is exchanged between mobile devices and backend.

A Sync Object is data:

Sync Object data is a business object instance exchanged between client and server.

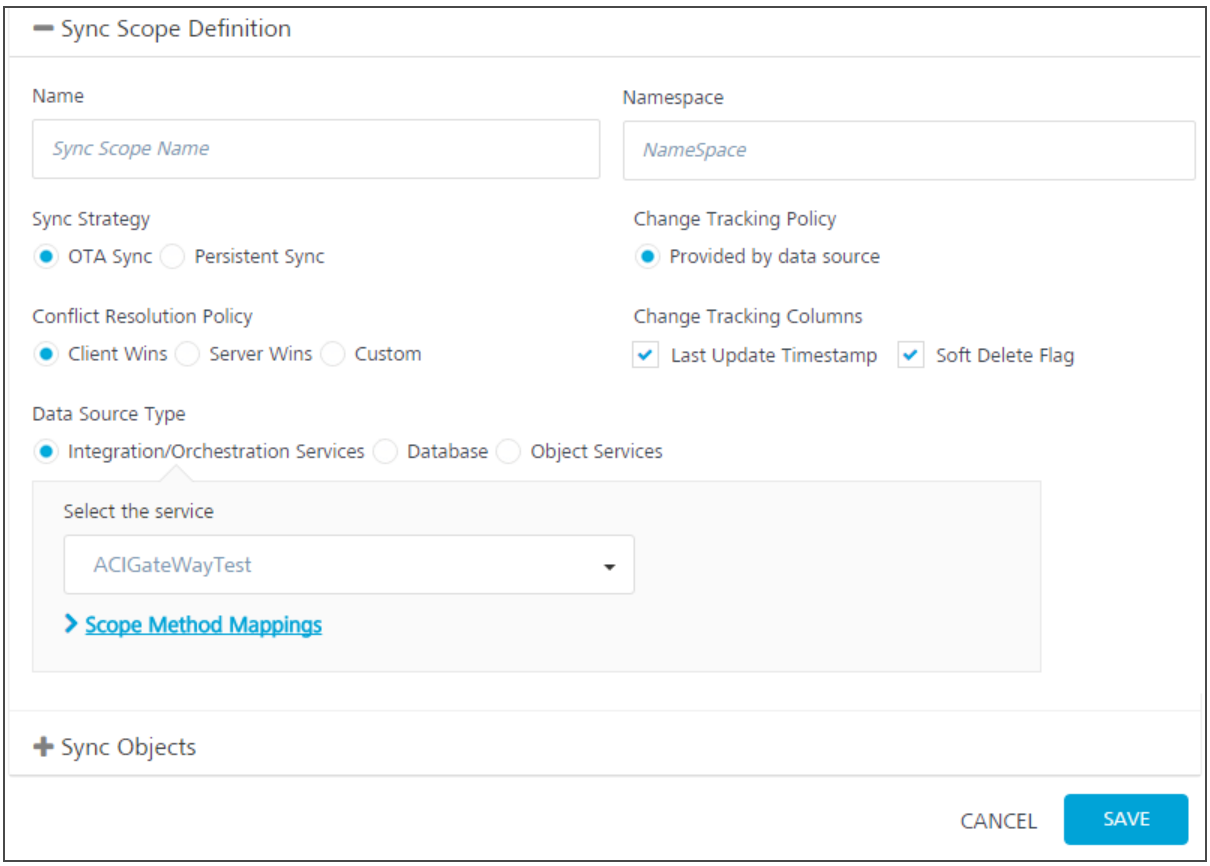
26.2 Adding a New Synchronization Scope

> ***Note:*** The following section explains setting up a Sync scope for a Salesforce account. The Sync scope defines the rules for how data is managed by the sync process. All objects under this Sync scope are bound to these rules.

To add a new Synchronization scope, follow these steps:

1. After you create an application, in the **Configure Services** tab, click the **Offline sync** service tab. The **Offline sync** page appears.
2. Click **CONFIGURE NEW**.
3. Provide a name for the new Sync scope, such as *FSSync.*

4. Under **Sync Scope Definition**, provide the following details:



   a. Specify a **Namespace** for the Sync scope. The Namespace should follow a prescribed format such as *com.kony*. It is not mandatory that you give a namespace for a Sync Scope.

   b. Select a **Sync Strategy**. The available options are **OTA Sync** and **Persistent Sync**.

      A persistent sync strategy is not supported for a Sync scope that uses an object service as the data source type. You must use an OTA Sync strategy for a Sync scope that maps to an object service. Use Integration/Orchestration Services or Database as the data source type for a persistent sync strategy.

      > ***Note:*** To understand which strategy to use for your sync scope, refer section Appendix - Sync Strategy.

c. Select a Change Tracking Policy (CTP) if you want to track the changes happening in the server database. Select Provided by data source if you have a provision to track changes in the data source. For example, a timestamp column in a database updates with any changes a row. Set CTP as Kony Sync Server if you want Sync Server to track the changes. The option is available only if you select Persistent Sync as the sync strategy.

d. In case of conflicts between the data at the client and server end, specify any of the following under **Conflict Resolution Policy**:

   - **Client Wins**: The changes on the client-side take precedent over the changes on the server side.
   - **Server Wins**: The changes on the server-side take precedent over the changes on the client side.
   - **Custom**: Enables you to upload an Interceptor class, which comprises the logic or policy for conflict resolution.

e. In the **Change Tracking Columns**:

   i. Select the **Last Updated Timestamp** when you have a column that represents the latest edited values. The column must belong to the timestamp data type.

   ii. Select the **Soft Delete Flag** check box when the database has the column that represents soft deletes. The soft delete field in a record represents that a particular record is deleted by changing the status to deleted. This record will exist in the database. Kony Fabric Sync does not sync records with the statuses set as deleted. Enterprise systems typically do not permanently delete data; an enterprise system will soft delete data by setting a Boolean field, such as isDeleted, to indicate that the record is deleted.

> ***Note:*** At run-time, a default filter is added to achieve delta sync when Last Update Time Stamp is selected in the Change Tracking column. The server appends the custom filter provided by the client to the default filter with an AND operator.
>
> In order of precedence, the custom filter must include brackets.
> For example: If a customer provides the custom filter as `$filter = (reviewerid eq a or revieweeid eq b)`.
>
> At run-time, the custom filter is appended to the default filter as `$filter =LastUpdateTime gt 2018-09-21 and (reviewerid eq a or revieweeid eq b)`.

5. Under **Data Source Type**, select one of the following:

   - **Integration/Orchestration Services**: If you select a service that does not have an identity service, set the scope method mappings for the Sync Scope. If you select a service that has an Identity service, specify the user ID and password.

     To use an Integration service or Orchestration service as the data source, follow these steps:

     - Click in the Select the service field. A drop-down menu appears. Select the service from the menu.

   - **Database**: Use this option if you want the Synchronization service to connect directly with the data source without going through an Integration service. This option is typically used for a persistent sync strategy.

     Sync Supports different databases as backend datasource such as Mysql, Oracle, Microsoft Server, and PostgreSQL.

To use a database as the data source, specify the following connection details of the backend database:

- **Database Type**
- **Database Connection URL**
- **User ID**
- **Password**

Click **Test Connection** to verify the connection to the database.

- **Object Services**: This is a Sync scope mapped to an object service. An object service has all the information to auto-generate the Sync scope, including objects, relationships, change tracking, and life-cycle methods. You need only provide the scope-specific data, such as sync strategy and filters; the rest of the Sync scope is inferred. If the object service changes, the scope is refactored to incorporate those changes.

  Note that a persistent sync strategy is not supported for a Sync scope that uses an object service as the data source type.

  To use an object service as the data source:

  - Click in the Select the service field. A drop-down menu appears. Select the object service from the menu.

  > ***Note:*** You can now select the storage object service as a data source type from the drop-down menu in the Select the service field. The storage object service must be created in **Object Services** tab. For more information on creating a new storage object service, refer to How to Create a Storage Object Service.

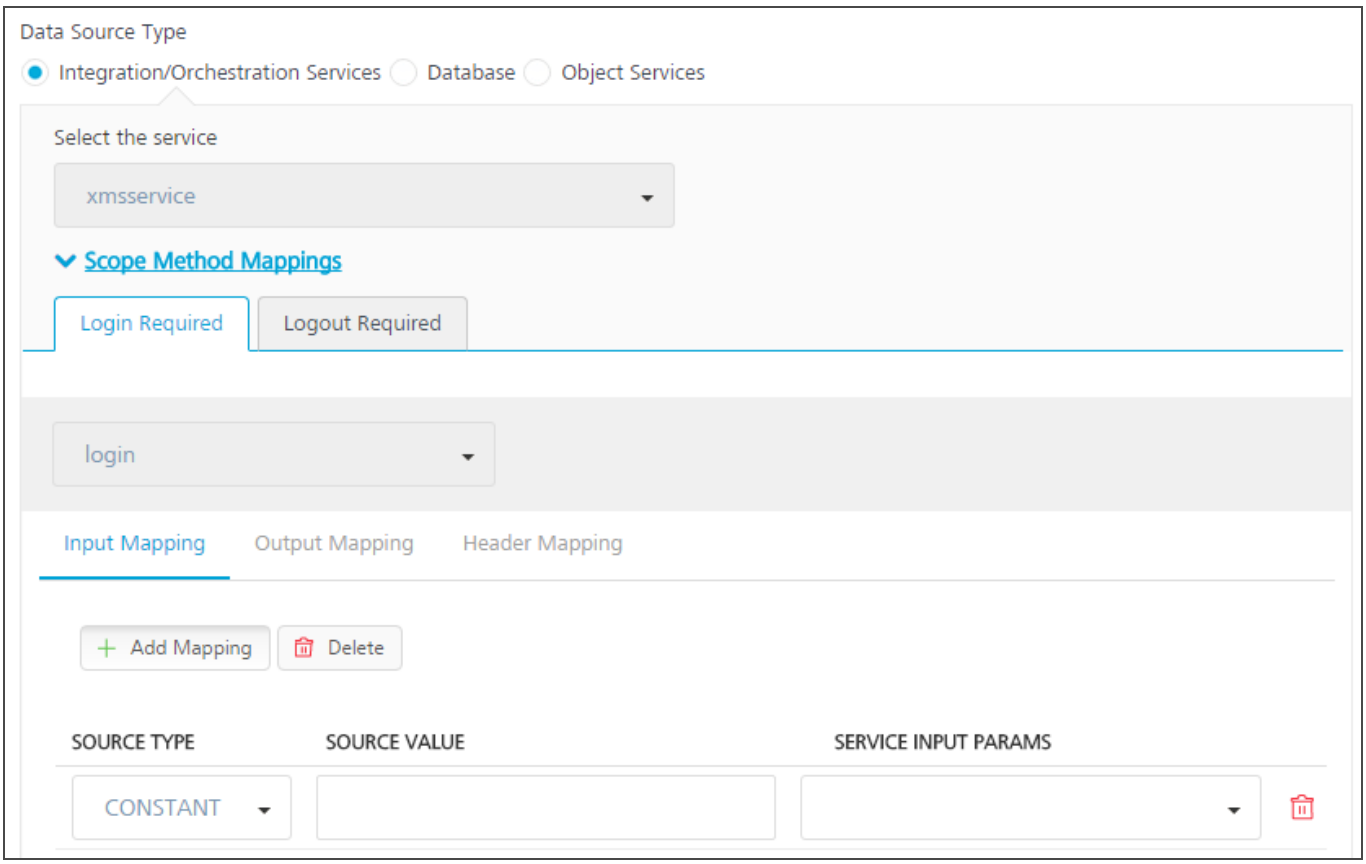
26.2.1 Authentication for the Datasource

For a Persistent sync, Kony Fabric Sync synchronizes the client device with the back end in a two-phase, asynchronous process. Sync first synchronizes the client device with the Kony Sync Server, and then synchronizes the Kony Sync Server with the back-end data source for a mobile app. Because the call to the back-end is not a device call, the Kony Sync Server must log in to the back-end data source.

For a Persistent sync, you must provide the log-in operation details for the back-end database in the Sync Scope definition. The persistent sync service starts immediately after you publish the sync configuration to the persistent database of the Sync Server. Then, when the client device connects to the Sync Server, the data from persistent database is downloaded to the client.

26.2.2 Scope Method Mapping

Use method mapping to map authentication operations for the Sync Scope. Mapping operations at the Sync Scope level are used primarily for Persistent Sync, but you can also map operations for OTA Sync.



To map methods for the Sync Scope, follow these steps:

1. In the sync scope Definition window, click **Scope Method Mappings**. The Login Required and Logout Required tabs appear in the Scope Method Mappings area.
2. Click in the field labeled None. A drop-down menu appears. Select an operation from the menu.

   The operations that are available are based on the Integration service that you specify.
3. Click **Input Mapping**, **Output Mapping**, or **Header Mapping**.

   Input Mapping captures how the input parameters of the log-in operation are populated. Output Mapping defines the response to the log-in method. Header Mapping defines how headers are stamped when the log-in method is invoked.

4. Click **Add Mapping**.

   The **Source Type**, **Source Value**, and **Service Input Params** fields appear. For output mapping, the service Output Params field appears. For header mapping, the service Header Params field appears.

5. Click in the **Source Type** field. A drop-down menu appears. Select the type of source from the menu to map to the Service Input Parameter, Service Output Parameter, or Service Header Parameter. The following types of source are available to map to the Service Input Parameter, Service Output Parameter, or Service Header Parameter.

   - **Constant**: Maps a constant value to the service parameter that you specify.
   - **Context**: Maps context from the device call to the service parameter that you specify. Context is content that you are tracking on the device side, and you use attributes from that context as service parameters.
   - **Template**: Maps a template to the service parameter that you specify. For example, a Velocity template that populates the service parameter.

6. Under **Source Value**, enter the value of the source to map.

7. Under **Service Input Params**, **Service Output Params**, or **Service Header Params**, select the parameter.

26.3 Defining Sync Objects

1. Click **Sync Objects**.

   The Sync Objects area appears.

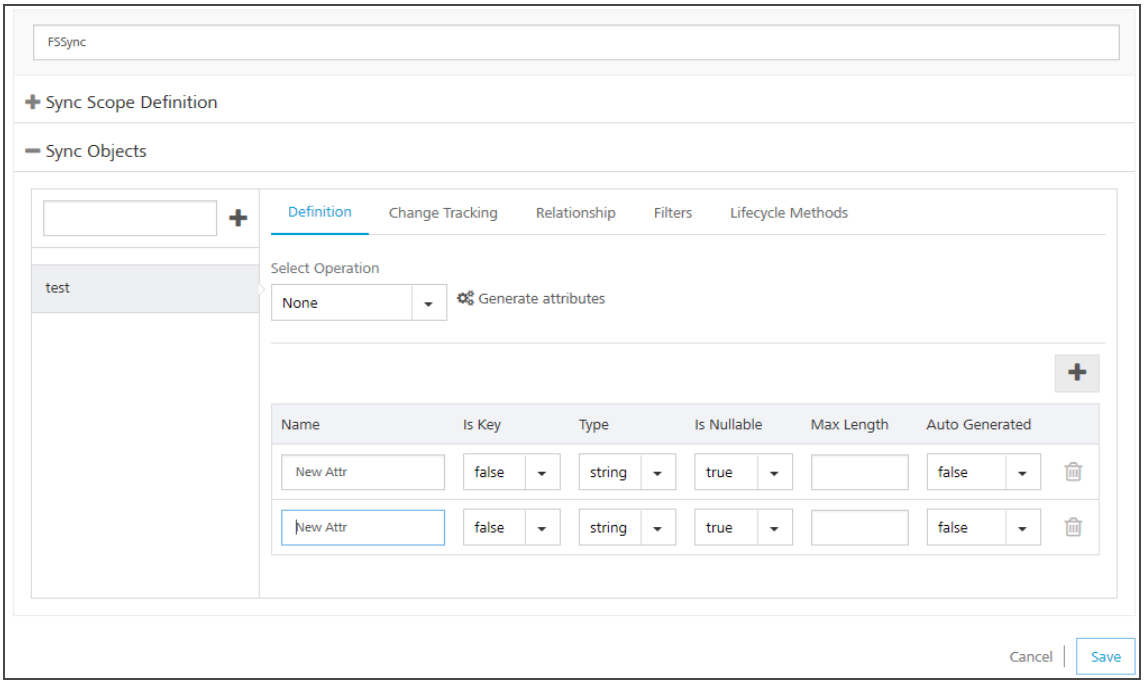
2. Click in the blank text box, and then enter a name for your Sync object.

3. Click the **Plus** button.

4. On the **Definition** tab of the new Sync object, click in the Select Operation field. A drop-down menu appears. Select an operation from the menu. Select the operation that will generate the maximum attributes for the object.

5. Click **Generate attributes**.

> ***Note:*** The list of operations available for a new Sync object depends on the Integration Service selected in the Sync Scope.



Set one of the generated attributes as primary key. For the primary key attribute, select true under **IS KEY**. If the attribute you set as primary key is an automatically generated column, select **true** under **Auto Generated**.

6. Under **Sync Objects**, provide the following details:

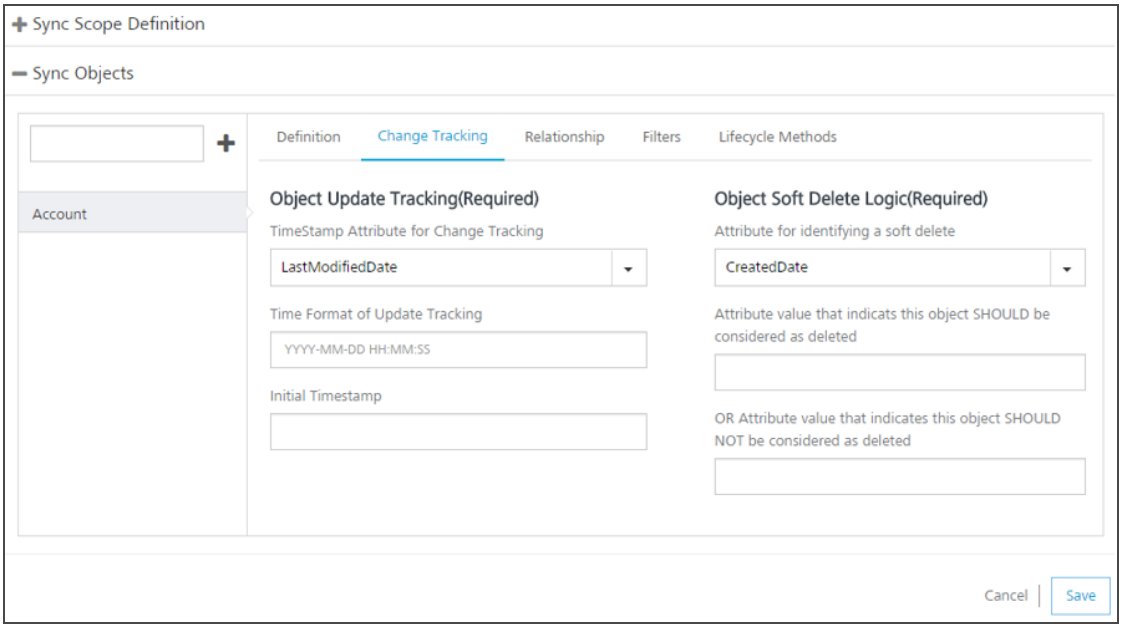
    a. On the left pane, provide a name for your Sync object, and then click the **Plus** button.

b. On the **Definition** tab of the new Sync object, select an operation from the **Select Operation** list, and click **Generate attributes**. Select the operation that will generate the maximum attributes for the object.

7. Click the **Change Tracking** tab. The Change Tracking fields appear. Do the following:
   a. Click the **TimeStamp Attribute for Change Tracking** list, select an attribute that denotes a particular record is modified.
   b. If required, change the **Time Format of Update Tracking**, if required. By default, Salesforce time format is yyyy-MM-dd HH:mm:ss.SSS
   c. Click the **Attribute for Identifying a soft deleted** list, select an attribute that denotes a soft delete.

> ***Note:*** You need to select **TimeStamp Attribute for Change Tracking**, only if you have selected **Last Update Timestamp** check box under the **Change Tracking Columns** respectively.

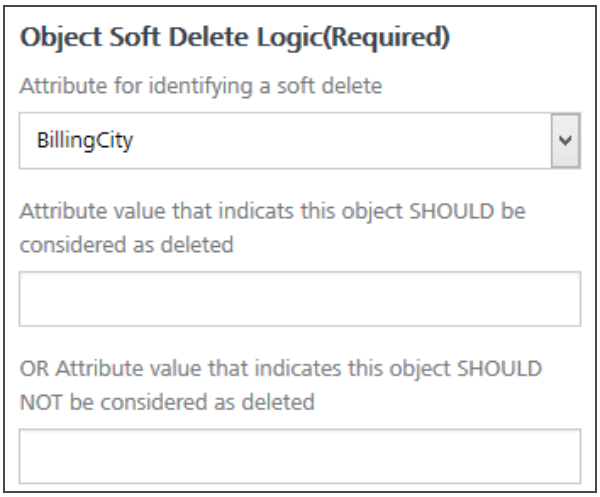
> ***Note:*** You need to select **Attribute for Identifying a soft delete** only if you have selected **Soft Delete Flag** check box under the **Change Tracking Columns**.



For non-Boolean attributes, enter additional values that will be considered for soft deleting. For example, from the list if you select **BillingCity**, the system displays the following fields.

- **Attribute value that indicates this object SHOULD be considered as deleted**: if this value matches with the main attribute, the system considers this row as a deleted row.
- **OR Attribute value that indicates this object SHOULD NOT be considered as deleted**: if this value matches with the main attribute, the system does not delete

this attribute.



d. In **Initial Timestamp** box, enter the date from which the records are to fetched.

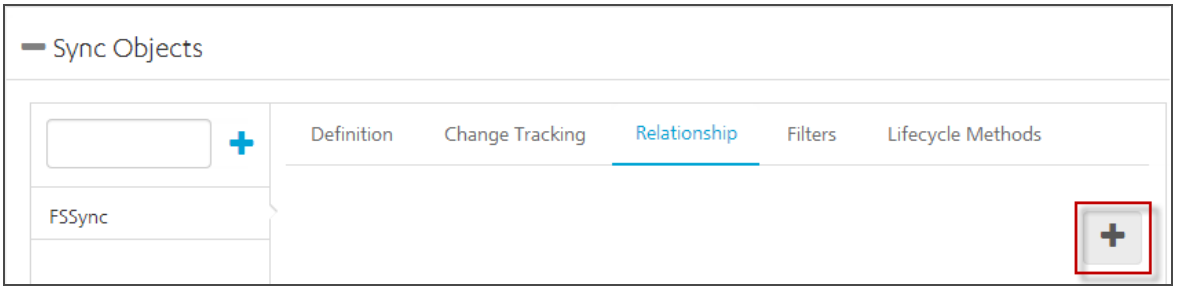
e. Click **Save**.

8. Click the **Relationship** tab, and then click the **Plus** button to open **Relationship** dialog.

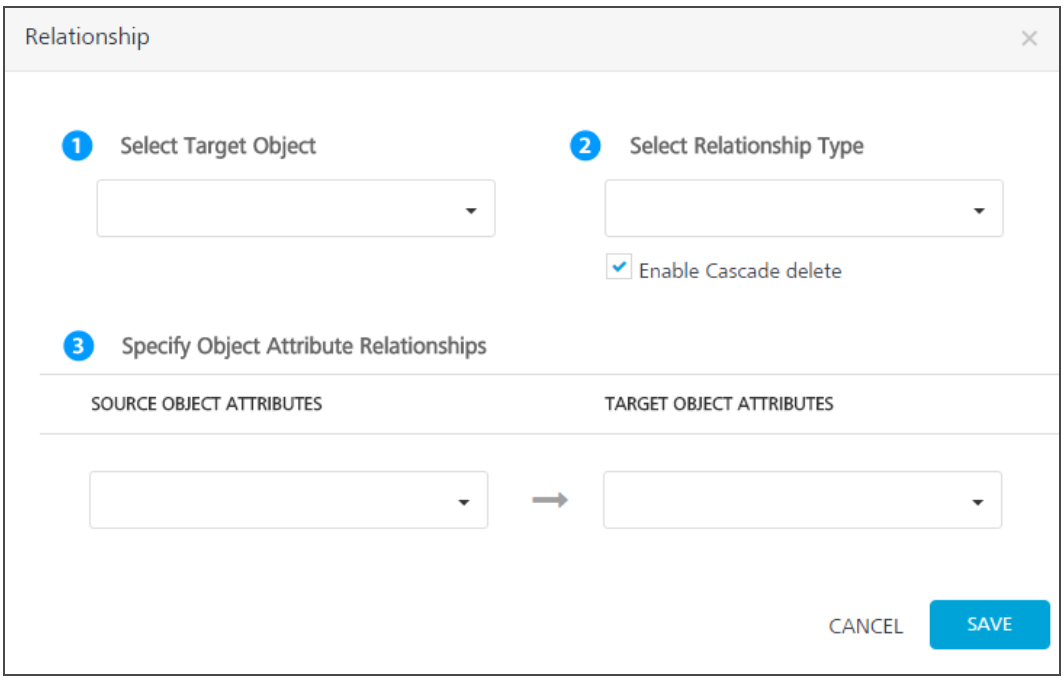
   The relationship that you define between two sync objects can consist of multiple columns. A foreign key that is composed of a collection of columns is called a composite foreign key.

   In a One to Many relationships, the target object attributes that you select constitute the foreign key in the table that is identified as the target object. The source object attributes that you select constitute the primary key in the table that is identified as the source object. The foreign key in the target object uniquely defines a row in the source object.

   In a Many to One relationship, the source object attributes that you select constitute the foreign key in the table that is identified as the source object. The target object attributes that you select constitute the primary key in the table that is identified as the target object. The foreign key in the source object uniquely defines a row in the target object.



a. Provide the following details:



i. Select the required object from the **Select Target Object** list.

ii. Select the required attribute from the **Target Object Attributes** list.

iii. Select the required attribute from the **Source Object Attributes** list.

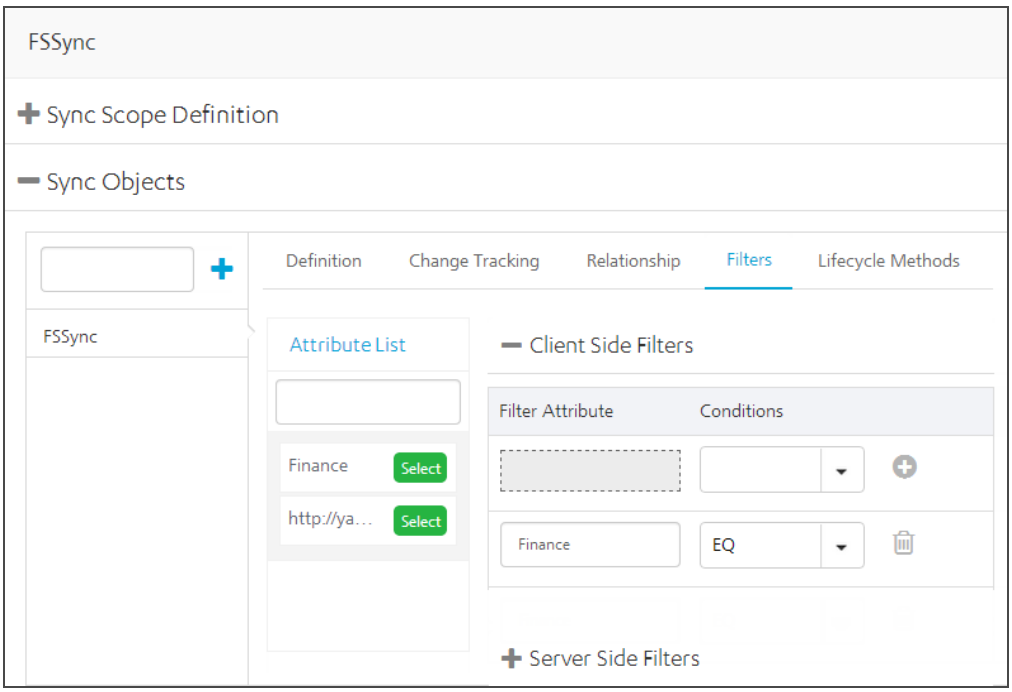
iv. Select the type of relation between Source attribute and target attribute from the **Select Relationship Type** list.

v. Select **Enable Cascade delete** if you want to delete a record in the parent table and its child tables.

When the relationship that you define between two sync objects uses a composite foreign key, an XML tag named "RelationshipAttribute" is added to the sync configuration file. For example:

```
<Relationships>
   <OneToMany TargetObject="Order" Cascade="false">
      <RelationshipAttribute
TargetObjectAttribute="CategoryID">
                        <SourceObjectAttribute="CategoryID"/>
      <RelationshipAttribute TargetObjectAttribute="OrderID">
                        <SourceObjectAttribute="OrderID"/>
   </OneToMany>
</Relationships>
```

9. On the **Filters** tab, provide the following details:



> ***Note:*** OAuth 2.0 provides ability to retrieve and save user attributes in Kony Fabric Identity Sessions after a successful login response and uses them as client filters during Legacy Sync calls. For example, User Role (one of the attributes of the user profile) received as part of User Profile after a successful OAuth 2.0 login can be used as client-side filter for Legacy Sync. For more details, refer to Synchronization > client-side filters.

a. In the **Client Side Filters**, from the **Attribute List**, select an attribute.

b. Select a scope for the client-side filter:

- **Request**: Filters data for the attribute by using the condition that you specify. Client Filter values can be set from the client app when the type is Request.
- **Constant**: Filters data for the attribute based on the constant value and the

condition that you specify.

- **Context**: Filters the attribute based on the context value and the condition that you specify. Context is content that you are tracking on the device side. You filter the attribute by the context that you specify.
- **Identity**: Filters profile data for the attribute based on the value and the condition that you specify. When you create a user profile for an Identity service, you get some additional data as part of the profile information, which you can use to filter additional attributes. The device owner cannot modify these attributes.

c. For the selected attribute, provide a condition.

d. To save the current filter and add another filter, click the **Plus** button.

e. In the **Server Side Filters**, from the **Attribute List**, select an attribute.

f. Under **Conditions**, apply a condition for the selected attribute.

g. For the selected attribute, provide a condition.

h. To save the current filter and add another filter, click the **Plus** button.

If you apply the Expression condition on client-side filters or server-side filters, it does not matter on which attribute the expression is applied. The Expression condition alerts Kony Fabric that the value must be evaluated before returning the data to the mobile application.

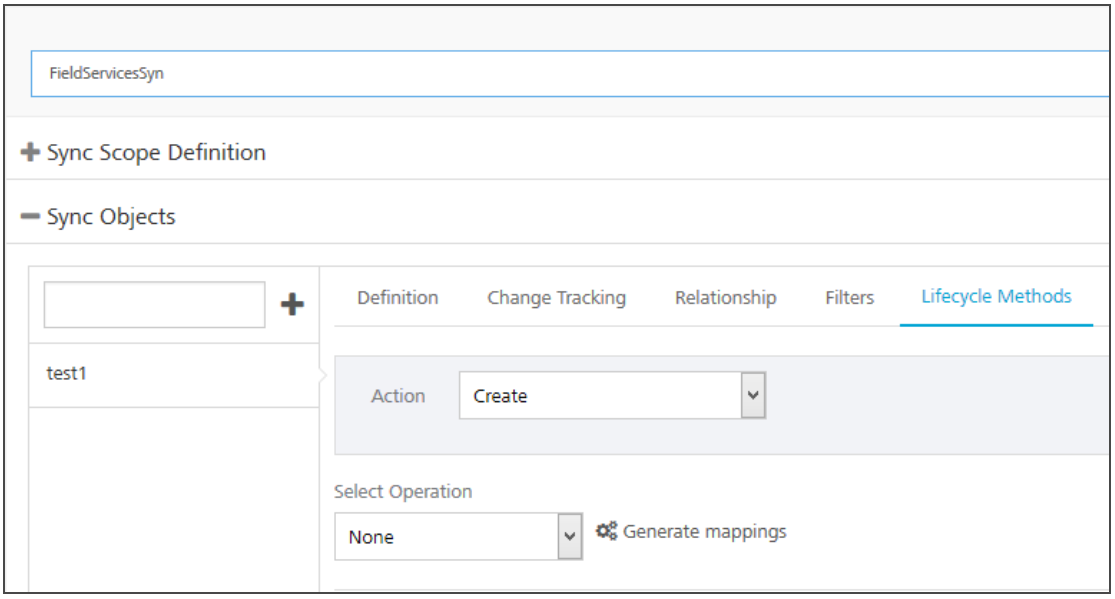
10. On the **Lifecycle Methods** tab, provide the following details:

> ***Note:*** The Lifecycle Methods are available only if the data source for the Sync service is an Integration service or an Orchestration service.

a. From the **Action** list, select an action.

b. From the **Select Operation** list, select an operation.

c. Click **Generate Mappings**.



To enable sync capabilities to an app, a developer must map sync actions with proper integration service operations.

Synchronization of data happens from a client to a server when the Sync Object calls the `startsession()` method. For example, `sync.syncstartsession()` if sync is returned by invoking `getSyncService()`. For more details, refer to SDKs.

- To synchronize data only from the server to the client, defining the `getUpdated` operation alone is sufficient.
- To enable bi-directional sync, a developer must define the `create`, `update`, `delete`, and `getUpdated` operations. Also, define the getoperation.

Sync Purpose	Sync actions	Integration service operations
To create a new account from a device (client) to a server. ***Note:*** As an example, for Salesforce apps, the **Account** object is used as a Sync Object for configuring the sync service.	Create	createAccount
To modify data from a device (client) to a server	Update	updateAccount
To delete a record from a device (client) to a server	Delete	deleteAccount
To get sync data from a server to a device (client) ***Note:*** This is specifically used for clearing the conflict policies.	Get	getAccount

Sync Purpose	Sync actions	Integration service operations
For Salesforce apps, to send a query request from a device (client), add the **Sync Service Input parameter** as **LAST_SYNC_TIMESTAMP** to getUpdated sync action. It is not mandatory to add the input parameter LAST_SYNC_TIMESTAMP. Sync Service creates a CONTEXT object to hold certain data about that sync session. You can use the CONTEXT attribute LAST_SYNC_TIMESTAMP in your input mapping query. For example, add a query to get any changed records since the last time the device did a sync. Create an input that maps to the **select** input parameter, of source type **Template**, and with a **Source Value** of: Select Id, AccountNumber, Name, Type, Rating, Phone, Fax, IsDeleted, LastModifiedDate from Account Where $CONTEXT.LAST_SYNC_TIMESTAMP and Id = '$FILTER.get('Account.Id')' If the sync scope type is **OTA** or **Persistent** and change tracking policy is provided by data source, **getUpdated** is called. If the sync scope type is **Persistent** and change tracking policy is **Kony Server**, **getAll** is called. For more details, refer to http://docs.kony.com/konylibrary/studio/Studio_User_Guide/Content/Creating Sync Configuration using RESTful XML DataSource	getUpdated	queryAllAccount

Sync Purpose	Sync actions	Integration service operations
To get the deleted records from the enterprise to the application if the getUpdated doesn't return the deleted records. You can add the queryAll operation and modify the name of the operation to avoid conflict, and modify the input and output to match the getDeleted functionality.	getDeleted	queryAll
To download any new or changed records in batches from the backend to the local device.	getBatch	queryMoreAccount
Similar to getUpdated, getAllPKs sync action is called to get the primary keys from server. This life cycle method is used to support Data Reconciliation feature on Offline Sync (Legacy Sync).	getAllPKs	queryAllAccount
To download any new or changed primary keys in batches from the backend to the local device. This life cycle method is used to support Data Reconciliation feature on Offline Sync (Legacy Sync).	getBatchPKs	queryMoreAccount

> ***Note:*** Input mapping is generated only for *Create*, *Update* and *Delete* operations.

> ***Note:*** Output mapping is generated for all the operations: *Create*, *Update*, *Delete*, *get, getUpdated*, *getDeleted* and *getBatch*.

> ***Note:*** Header Mapping needs to be added manually.

d. Add Input parameters from the **Input Mapping** by clicking the **Plus** button. Provide the following details:

a. From the **Source Type** list, select the type of the source.

b. From the **Source Value** list, select a value.

c. From the **Service Input Param** list, select an input parameter.

d. Click **Save**.

e. Add Input parameters from the **Output Mapping** by clicking the **Plus** button. Provide the following details:

a. From the **Source Type** list, select the type of the source.

b. From the **Source Value** list, select a value.

c. From the **Service Output Param** list, select an input parameter.

d. Click **Save**.

f. Add Input parameters from the **Header Mapping** by clicking the **Plus** button. Provide the following details:

a. From the **Source Type** list, select the type of the source.

b. From the **Source Value** list, select a value.

c. From the **Service Header Param** list, select an input parameter.

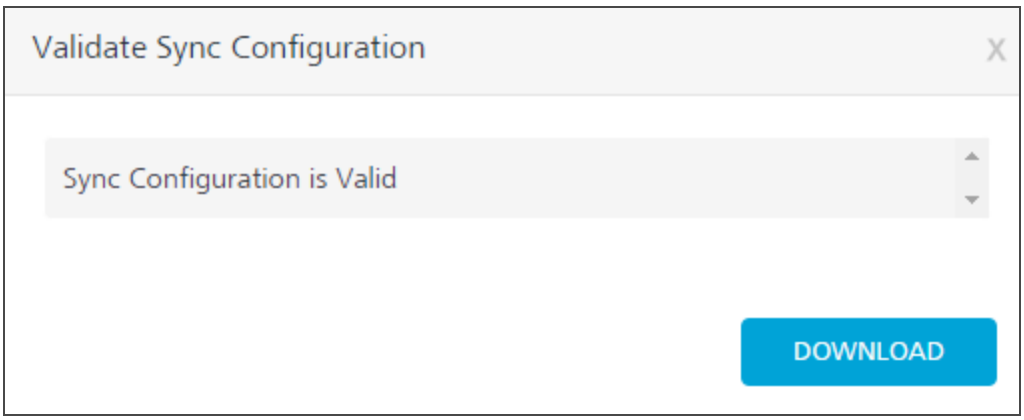
d. Click **Save**.

26.4 Validate Sync Configuration

You can use Kony Fabric to validate a Sync configuration. Validate a sync configuration before use the scope in your application.

To validate your Sync configuration, on the **Offline sync** page, click the **Settings** button and then click **Validate all**.

You receive the following message if your scope is valid:



To download the file, click **Download**. This file is useful when the Sync Scope is invalid, and you want to know the details of the errors encountered while validating the Sync Scope.

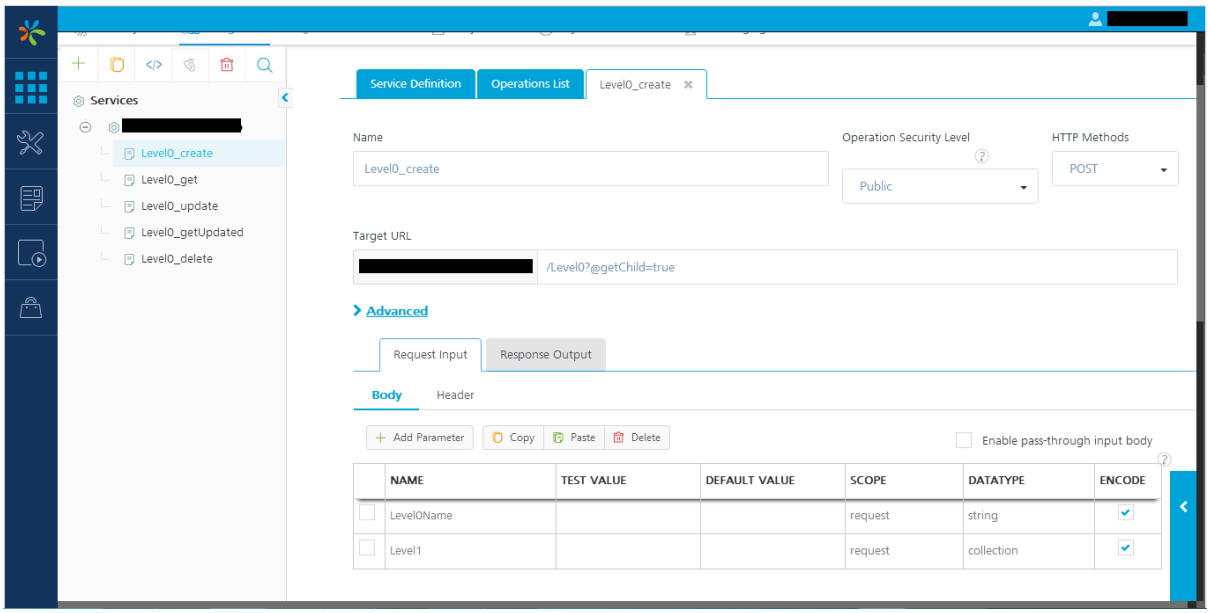
26.5 Use Parent-Child Upload

Parent-Child Upload is a feature of the Synchronization SDK that allows you to include all changes to parents and children to be transmitted to the backend in a single complex data structure. This increases system efficiency by permitting all information to be transmitted at once, minimizing network calls and processing. Parent-Child Upload also enables communication with a Sky/SAP backend, which requires such structured data.

26.5.1 Creating a Sync Configuration with Parent-Child Upload support

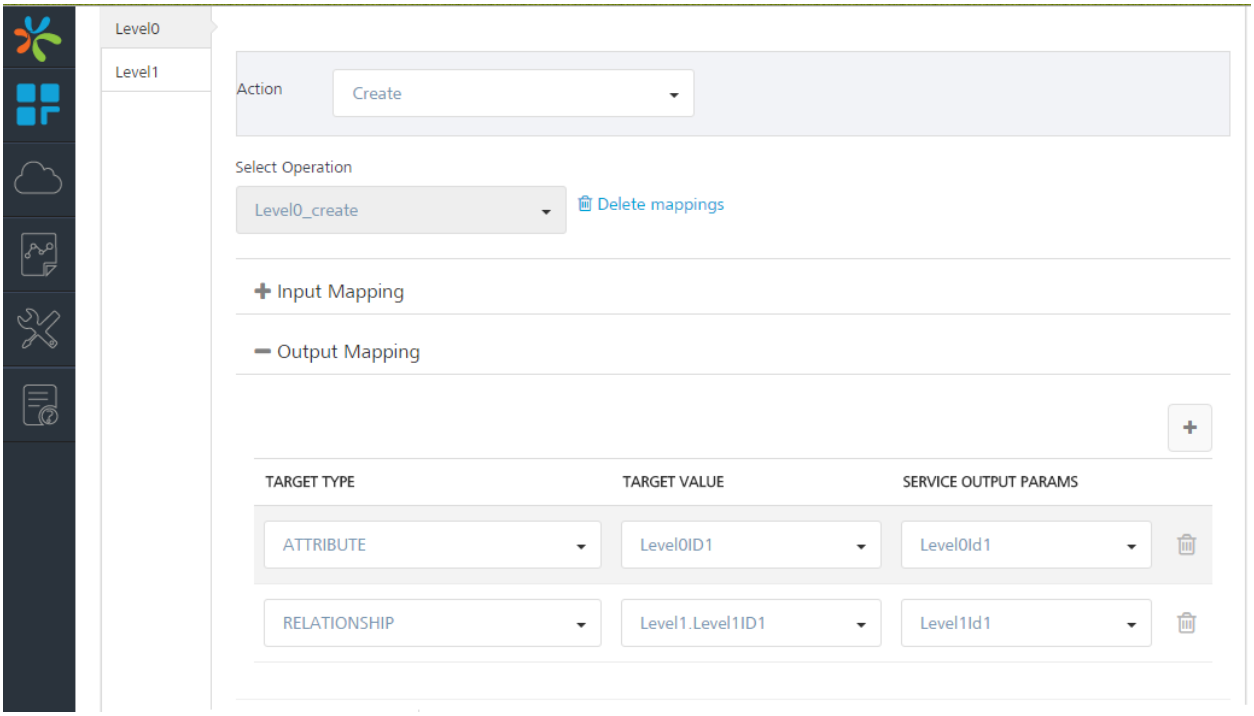
The current version of Kony Fabric allows configuration of the parent-child upload in the input mapping of the Sync Configuration under the Synchronization tab using template parameters. This is done through the Kony Fabric console and by modifying the request template.

1. Using the Console, nest the parameters, making the child parameters of datatype **Collection**.



To access the child object in the parent object's input mapping, use source type collection. That is, if the backend service expects the parent-child data in a Create request to be formatted as a nested object, then the child must be mapped as a collection under the parent's input mapping.

2. Configure the sync objects and mapping the parent-child relationships.



To access the child object in the parent object's output mapping, use target type relationship. That is, if the back-end service responds even with child data for a service at parent level then the child attributes can be mapped using the target type relationship.

While building a sync app with SAP back-end through Kony Fabric Console, the above two points need not be followed as the above flow is taken care of internally.

Note that the upload batch size used by the client SDK takes a value closer to that mentioned in the app for parent-child feature. For example, if the developer specifies the upload batch size as 1 in the app and then updates N child records on the device side, the upload batch size will be taken as 2 to include both the child and the parent .If the developer now performs a sync, the data will go in N batches, each batch containing parent and child.

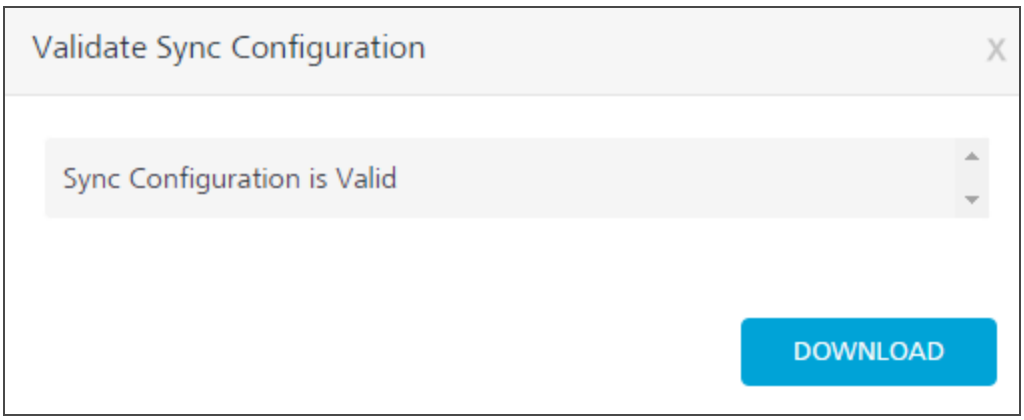
26.5.2 Feature Limitations and Behavior - Parent-Child

- Error message mapping is not supported at the child level from User interface. User has to manually edit the app and add error message mapping at object level.
- To support propagation of parent-child with different actions, currently only the first action of each child is propagated along with a parent and successive actions of a child will be applied as individual operations of child. In this propagation note that delete actions on child are not propagated along with parent. They are applied individually on child. So if delete is the first action on child then that delete and successive actions are applied individually on child but not propagated.
- If Cascade is set to true and the developer configures a Delete operation to send a parent and child in same batch, both the parent and child will be sent. This is unnecessary; deleting the Parent will automatically delete the Child when Cascade is set to true. (If the backend has Cascade set to true, child data such as the primary key is not required for Delete.) This may decrease performance of the backend when a parent with many children is deleted, because a separate delete will be attempted for every child, despite the children having already been deleted.

  Workaround: When deleting a parent with children and Cascade is set to true, send only the Parent.

26.6 Download the Sync Configuration

To download the Sync configuration file `Syncconfig.xml` file on your computer, click on the **Synchronization** page, click the Settings button, and then click **Export all**.



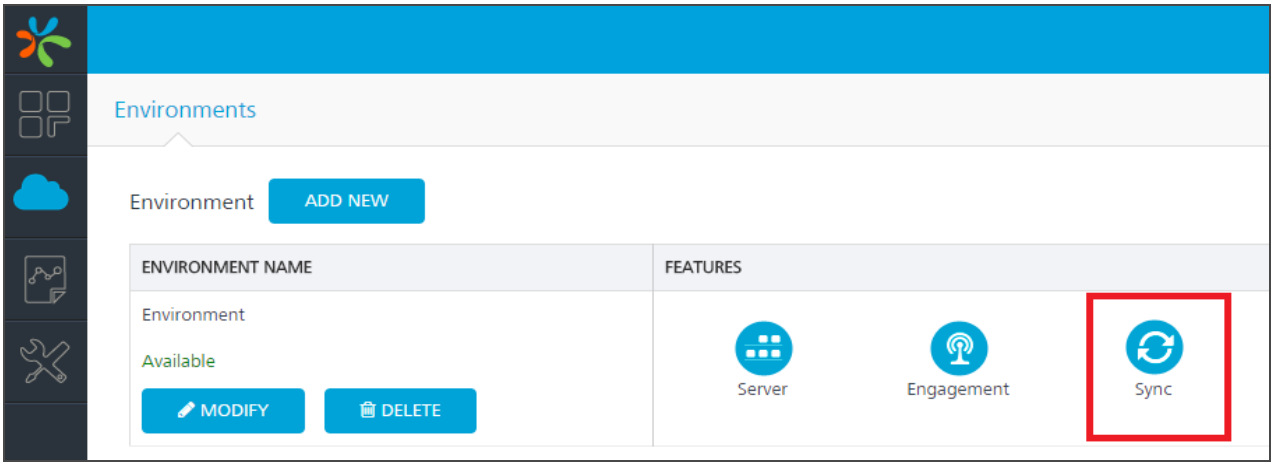
For more details on Sync Console, refer to the following document:
http://docs.kony.com/konylibrary/sync/kony_sync_console_user_guide/Default.htm

26.7 Kony Fabric Sync Console

> ***Note:*** The details of your sync scope will be available in Sync Services after you *publish* the app.

Kony Fabric Sync Management Console provides a single point of control for monitoring and configuring the Kony Fabric Sync console creation process.

To view your Sync Console, click **Sync Services** from your Environments.



For more details on Sync Console, refer to the following document:
http://docs.kony.com/konylibrary/sync/kony_sync_console_user_guide/Default.htm

27. Kony Developer Portal

Kony Fabric Developer Portal feature lets you create a Portal for exposing APIs created using Kony Fabric. Developers from internal and external partner teams can access the portal created to explore and test the APIs.

To configure a Dev Portal, you will need to ensure that you have Kony account credentials (through manage.kony.com). For more information, or if you need to register for a Kony account, see the topic Accessing the Kony Fabric Console.

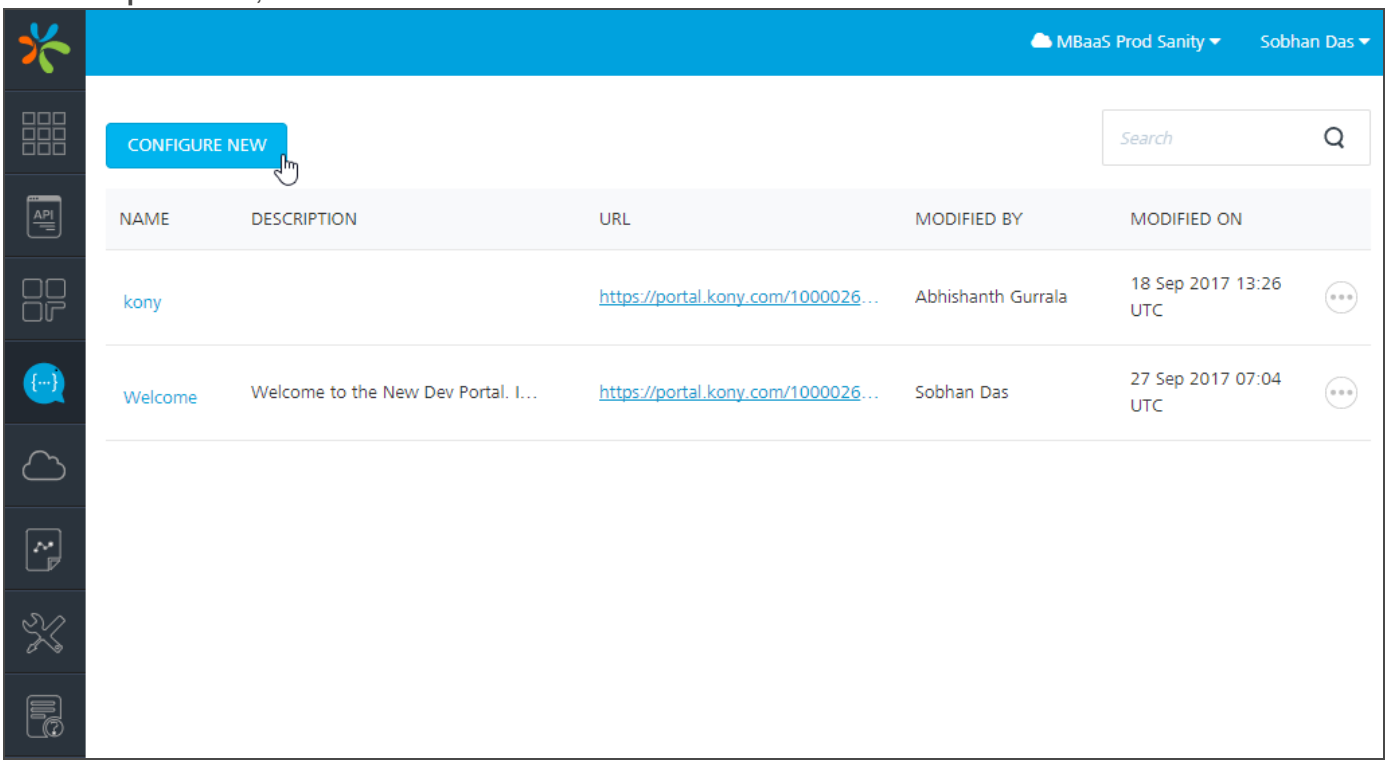
27.1 Creating a New Developer Portal

You as an Admin or owner can create, update, or modify Developer Portal for your partners to access the service APIs of your app. A member can only view Developer Portal.

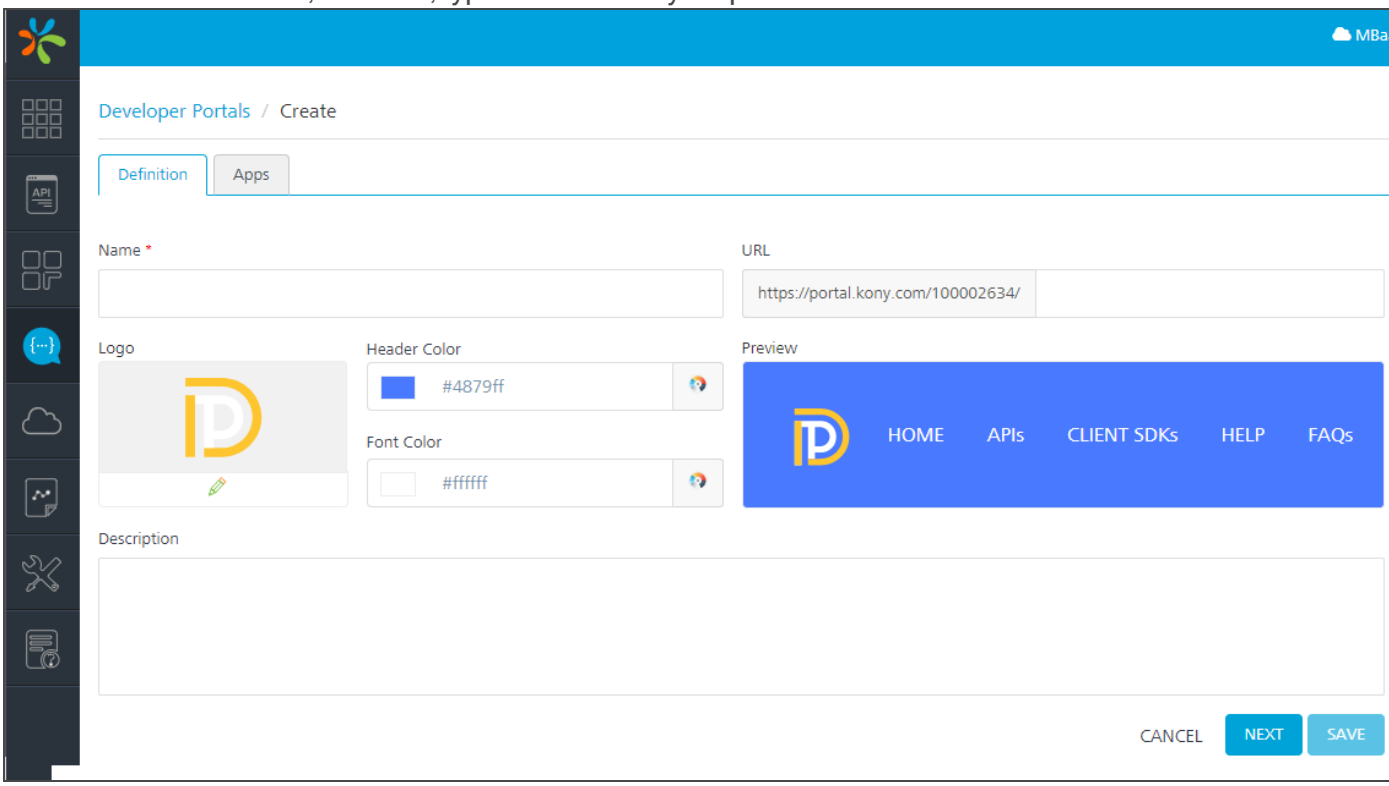
Each Dev Portal has a default logo, and **Header** menu with pages corresponding to the portal home page, APIs, and more. The **HOME**, **HELP**, and **FAQs** pages can be modified to suit your needs. The basic information is covered in the following procedure, while more details about fully modifying your portal can be found in the section Customizing a Developer Portal later in this topic.

To create a new developer portal, do the following:

1. In **Developer Portal**, click **CONFIGURE NEW**.



2. Under the **Definition** tab, in **Name**, type the name for your portal.



> ***Note:*** As you type a name for your portal in the **Name** field, the **URL** field populates the name to create the URL for the new portal, by default. You can edit the URL while creating the portal. Once the portal is created you cannot change the URL. Any special characters in the Name field will be ignored for the URL.
> While editing the Dev portal, you cannot change the URL.

3. If you want a custom logo for your Dev Portal, click on **Logo**, select the image file containing your logo, and then click **Open**.

4. If you want your Dev Portal to have a different font color or header color, click **Header Color** and **Font Color**.

> ***Note:*** The logo, header color, and font color can all be changed after the Dev Portal is created. For more information, see the Customizing a Developer Portal section later in this topic.

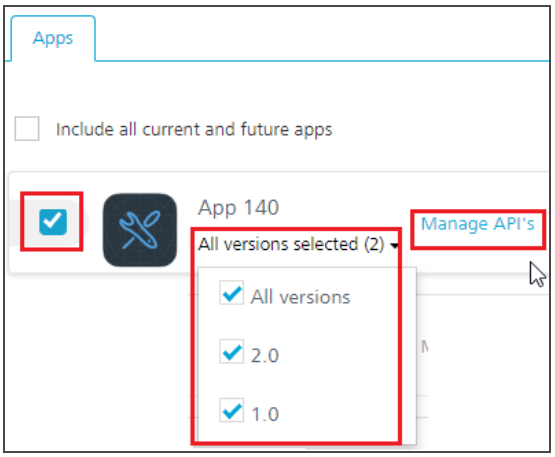
5. In Description, type a short description of your Developer Portal.
6. Click **Next**.

   By default all the apps are linked to Developer portal.
7. To **add specific apps** to Developer portal, do the following:
   a. Go to the **Apps** tab.
   b. Select the required check boxes for **the apps** that you want to add to the Dev Portal.

      > ***Note:*** To select **all apps** that you want to add to the Dev Portal, select the **Include all current and future apps** check box.

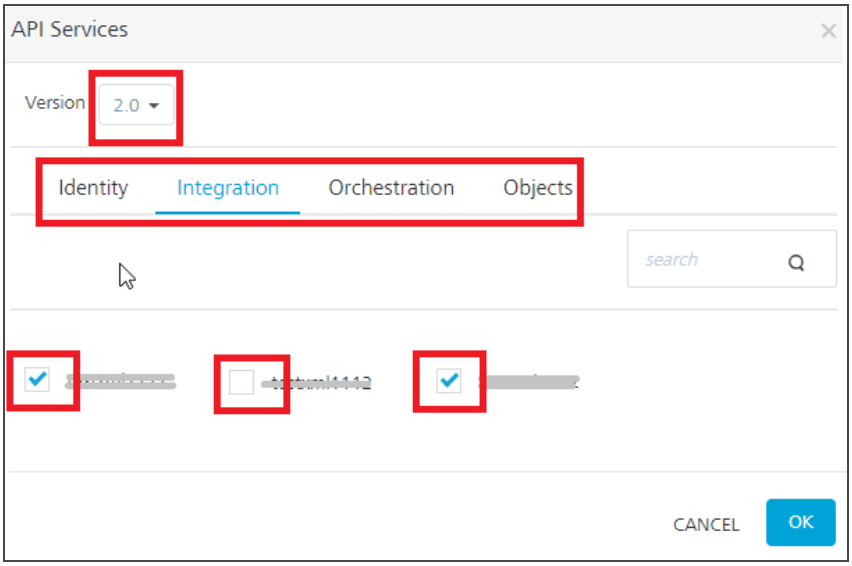
   c. click **SAVE** to add the selected apps with services to Developer portal.
8. (From Kony Fabric V8 SP4 onwards) To **add specific services** of an app to Developer portal, do the following:
   a. Go to the **Apps** tab.
      i. Select the required check box for **the app**. If you want to add **multiple versions of the app**, you can select the available app versions from the **Version** list. The

Manage APIs link is enabled for the selected app.



i. Click the **Manage APIs** link. The **API Services** dialog appears and displays services tabs such as Identity, Integration, Orchestration and Objects tabs for the selected app version. By default all the configured services for each app version are selected in each service tab.

   You can view the services from each version by selecting the required version.



i. Select the app version from the **Version** list.

i. Click the required services tab and clear the check boxes for required services.

> *Important:* Repeat this step if you want to add specific services from different versions of the app.

ii. Click **OK** to save the settings and close the **API Services** dialog.

b. In the **Apps** tab, click **SAVE** to add app with services to Developer portal.

> *Note:* You can also search for specific apps by entering an app name in the **Search** box.

Now you can log into your Developer portal account and can view the add apps with services.

The selected services of an app are listed in the swagger file of the app. You can click API Documentation in Developer Portal to download services details.

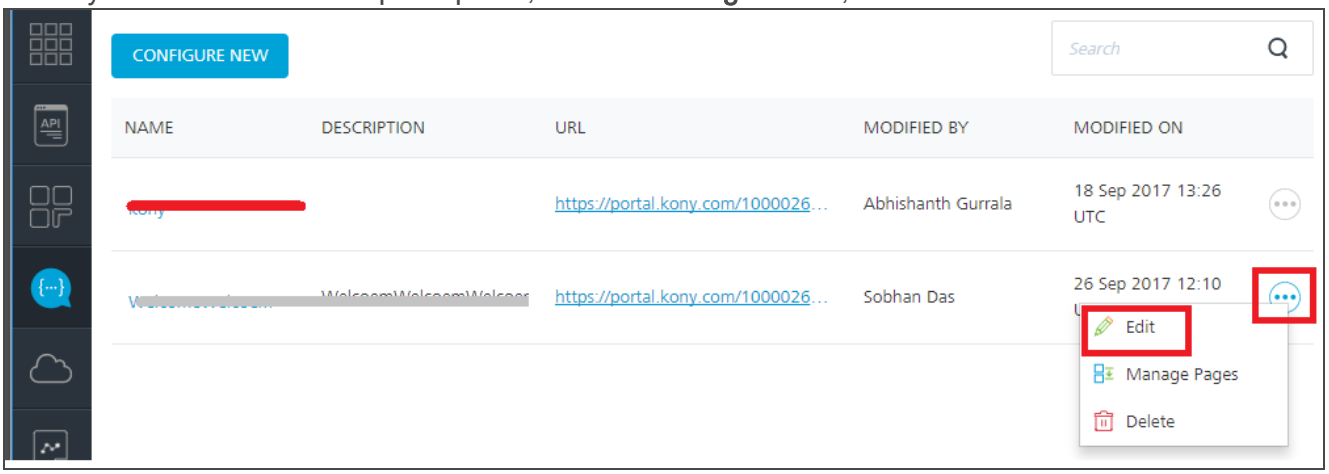
27.2 Customizing a Developer Portal

After you have created a Dev Portal, you can modify or update the information on the portal. For example, you might want to add additional frequently asked questions, or give more details on what the apps in the portal can do. You can also modify the look and feel of the portal pages, and add or remove pages, if necessary. The procedures in this section describe how to edit your portals and how to manage the pages of the portal.

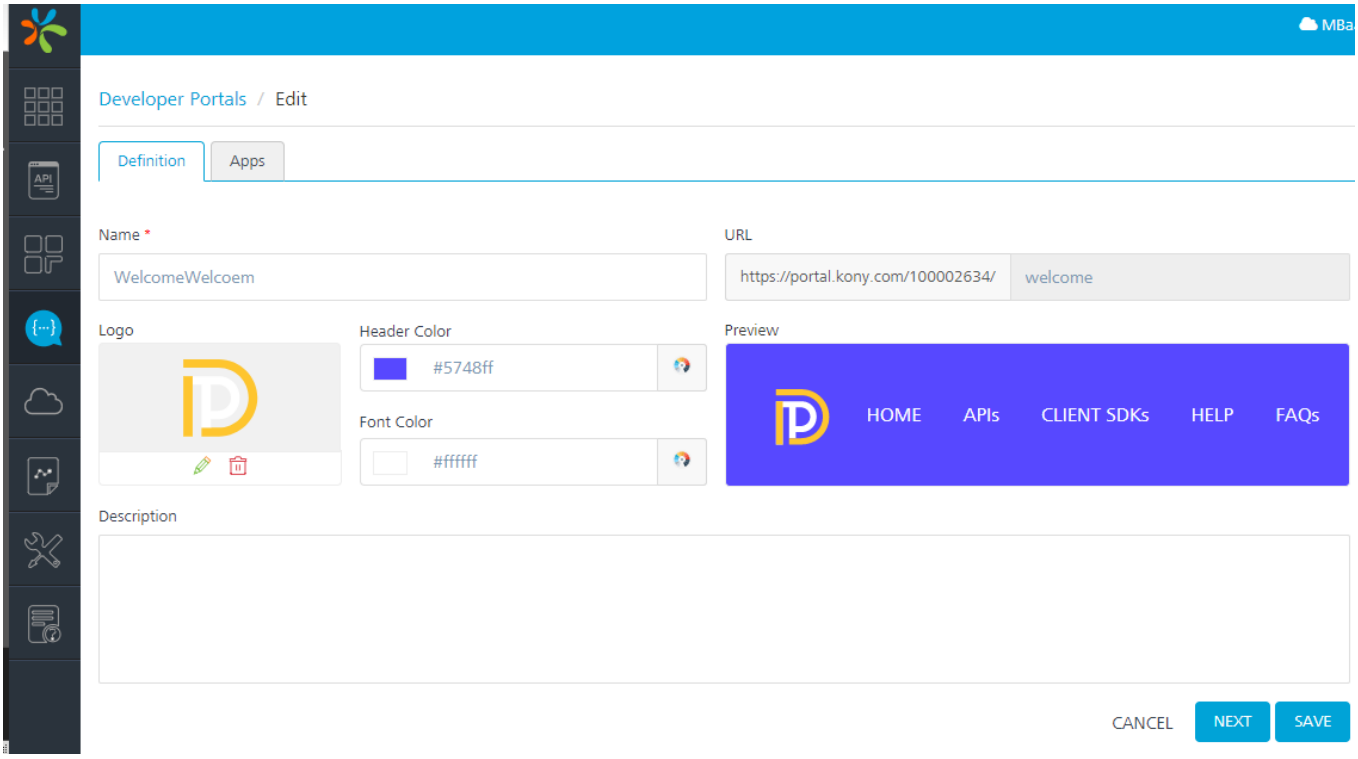
27.2.1 Editing a Developer Portal

To edit a Dev Portal, do the following:

1. Hover your cursor over the required portal, click the **Settings** button, and then click **Edit**.



2. Under the **Definition** tab, in **Name**, type the name for your portal.



3. If you want to add a custom logo to your Dev Portal, click on **Logo**, select the image file containing your logo, and then click **Open**.

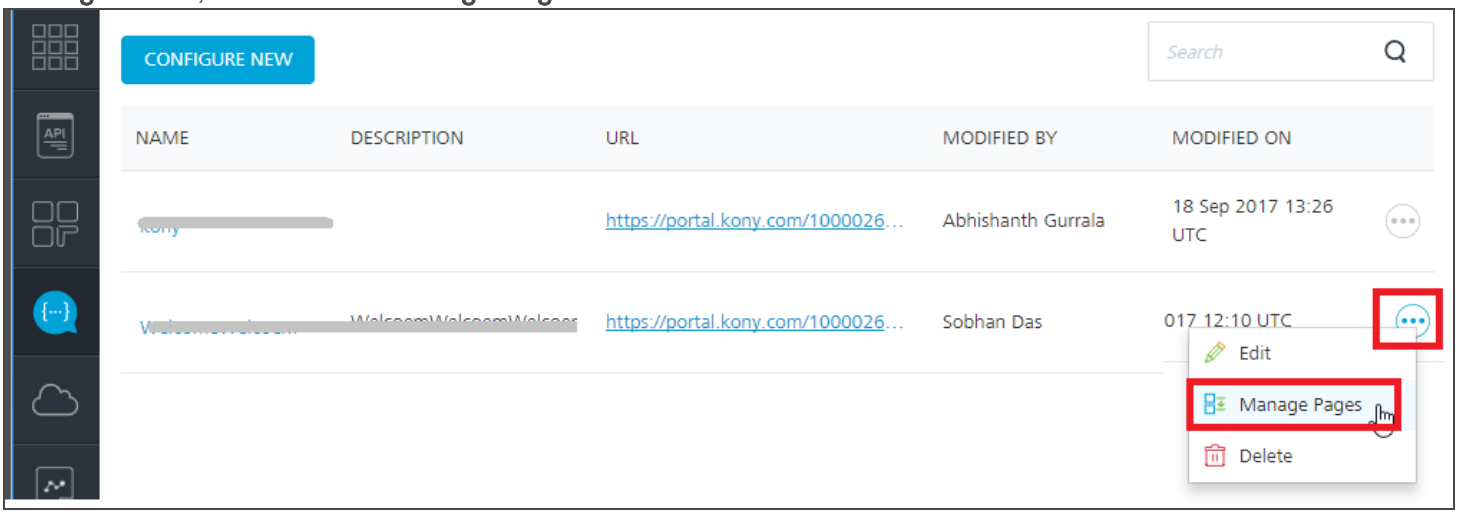
4. If you want your Dev Portal to have a different font color or header color, click **Header Color** and **Font Color**.

> ***Note:*** You can view the header color and font color in the preview section before saving.
>
> The logo, header color, and font color can all be changed after the Dev Portal is created.

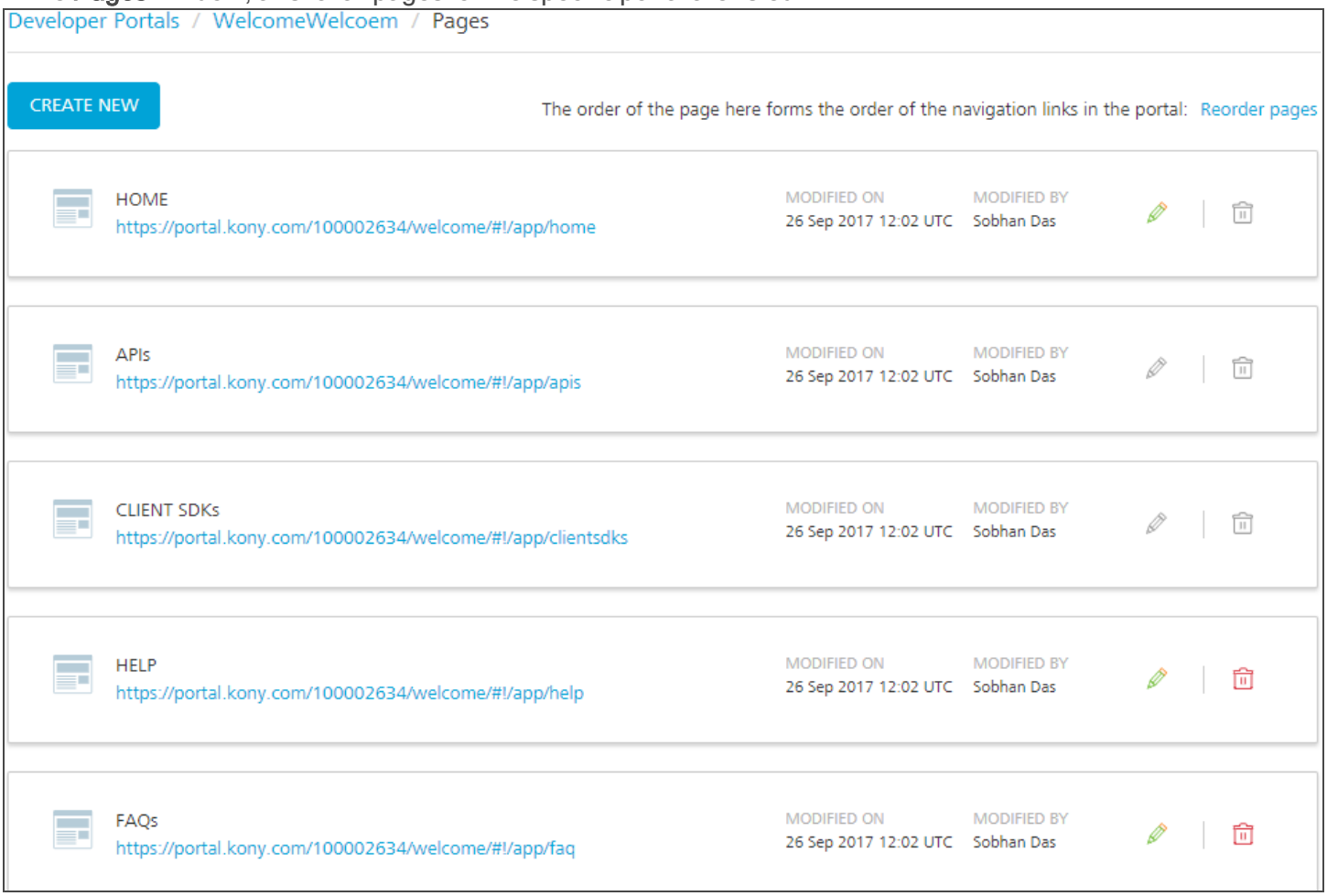
5. If you want to modify the description of the portal, type a description in the **Description** field.

27.2.2 Managing Dev Portal Pages

To manage the pages available in the Dev Portal, hover your cursor over the required portal, click the **Settings** button, and then click **Manage Pages**.



In the **Pages** window, a list of all pages for the specific portal are listed.



The order of the pages in the list determines the order of the navigation links that are displayed in the portal header. The default order of pages are:

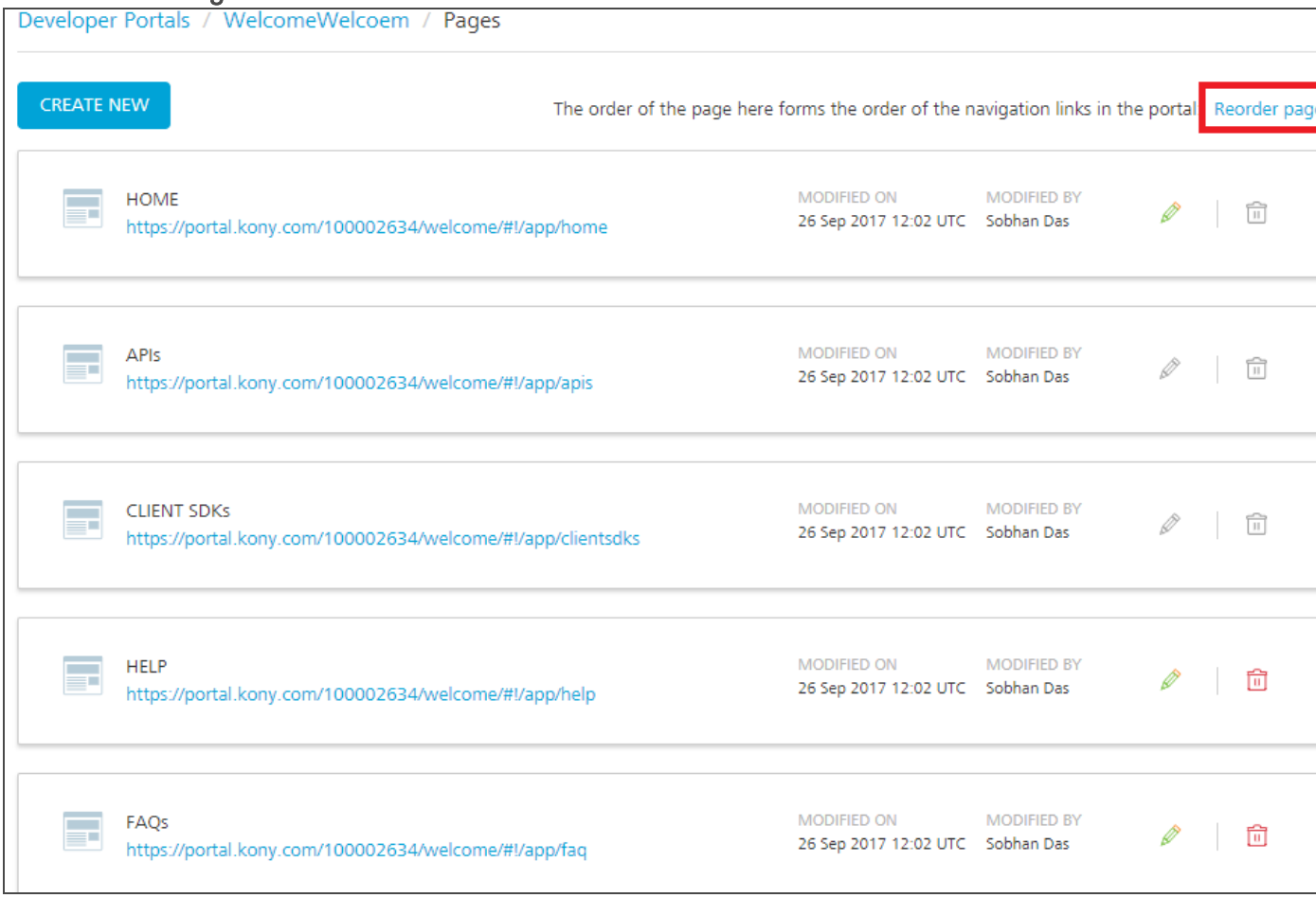
- **HOME** - The home page for the portal.
- **APIs** - A page listing all apps available through the portal, along with their corresponding APIs.
- **CLIENT SDKs** - Links for users to download the appropriate SDKs for using the portal (such as iOS or Android).
- **HELP** - A page with a brief description on how to use the portal.
- **FAQs** - A page for Frequently Asked Questions.

All of the pages are pre-populated with general information by default. However, the **HOME**, **HELP**, and **FAQs** pages can be modified as needed, or additional pages can be added.

27.2.2.1 Modifying the Page Order

To modify the order of the pages, do the following:

1. Click **Reorder Pages**



2. Drag-and-drop the pages into the order you want. When you are finished, click **SAVE PAGE ORDER**. The **HOME** page is not draggable.

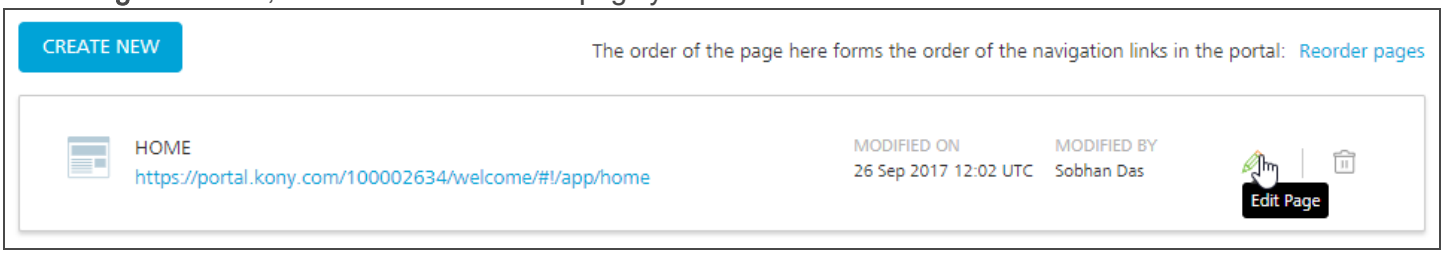
27.2.2.2 Editing a Specific Page

If you have selected Manage Pages for a portal, you can edit the specific pages of your portal. For example, to provide new or updated information, or to add new FAQs.

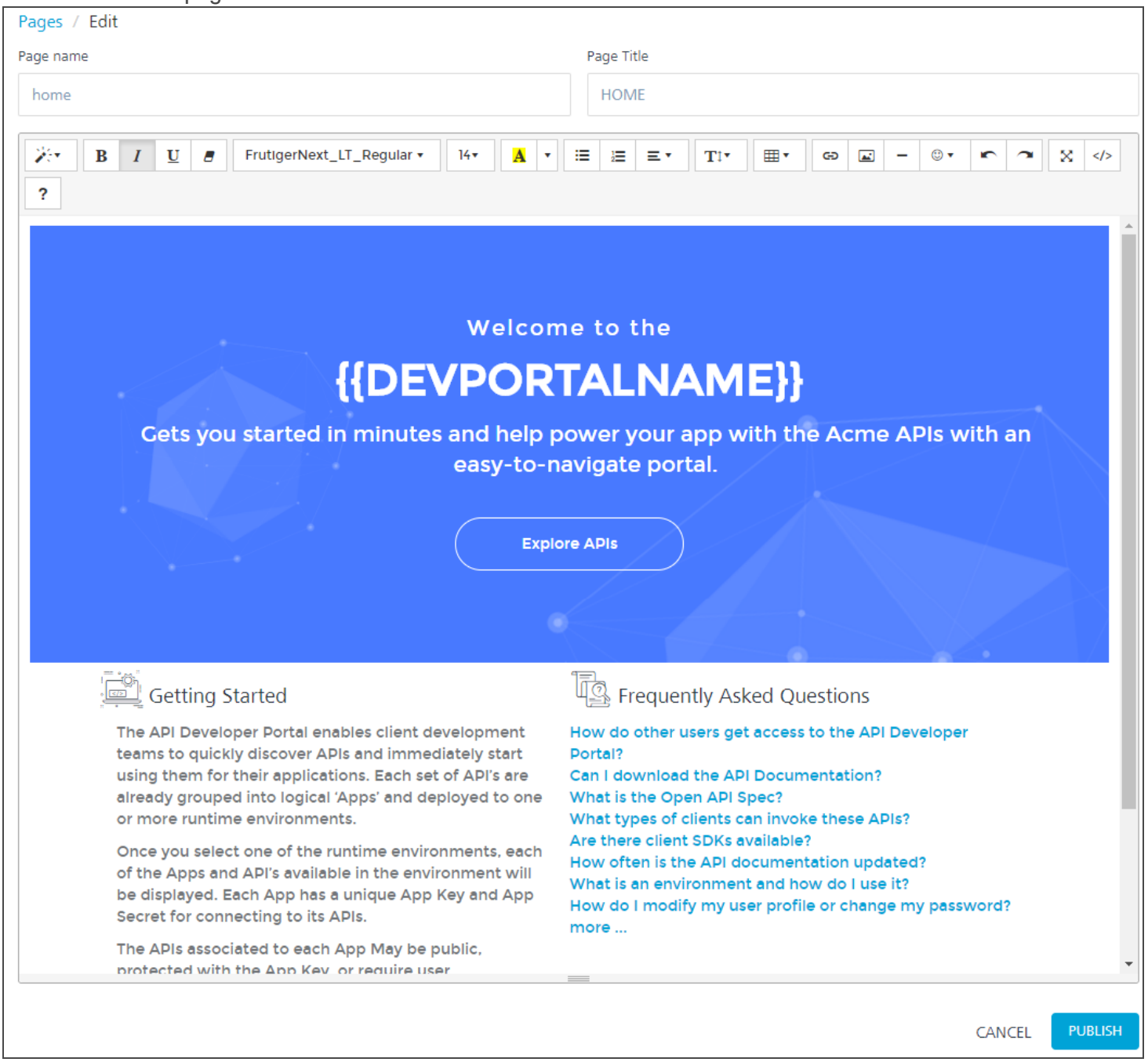
> ***Note:*** You cannot edit the **APIs** or **Client SDKs** pages.

To edit a specific page, do the following:

In the **Pages** window, click the **Edit** icon of the page you want to edit.



The page appears with a graphical interface that you can use to directly add, modify, or delete information on the page.



The menu at the top of the page provides functionality for changing fonts, changing styles, adding links, inserting images, switching to code view for the page, and more. For information about each button, hover the mouse over the button and a brief description of the button's functionality will appear.

In the interface, you can click anywhere on the page and add or modify text.

If there are links on the page, such as **Explore APIs** in the previous image, you can click on it and do one of the following:

- Select **Edit** to change the name of the link and the destination of the link.

Or

- Select **Unlink** to remove the link.

When you have finished editing the page, click **PUBLISH** to save the updated page, or click **CANCEL** to leave the page without saving updates.

27.2.2.3 Deleting a Specific Page

To delete a page, click the **Trash Can** icon associated with the page.

> ***Note:*** By default, you cannot delete the **HOME**, **APIs**, or **CLIENT SDKs** pages.

27.2.2.4 Adding a New Page

To add a new page to your portal, click the **CREATE NEW** button, and, use the interface to design your page.

When the page is complete, click **Save**.

27.3 Granting Developer Portal Access

After the Dev Portal is configured, you can provide users access to the portal by inviting them using the Kony Fabric Console. There is a specific role added to the user account that has been added to Kony Fabric called **Developer Portal Only**. This provides access to specific Dev Portals only and does not provide access to the Kony Fabric Console.

27.3.1 Inviting a new user for Developer Portal only access

> ***Note:*** The new user should not be part of the Dev portal.

To invite a new user to access the Developer Portal, do the following:

1. In **Settings**, and click the **INVITE** button.
2. In the Invite User window, type the user's email address, and under **Account Role**, click the drop-down box and select **Developer Portal Only**.
3. In **Environment Permissions**, select the environment or environments to grant the user access to.

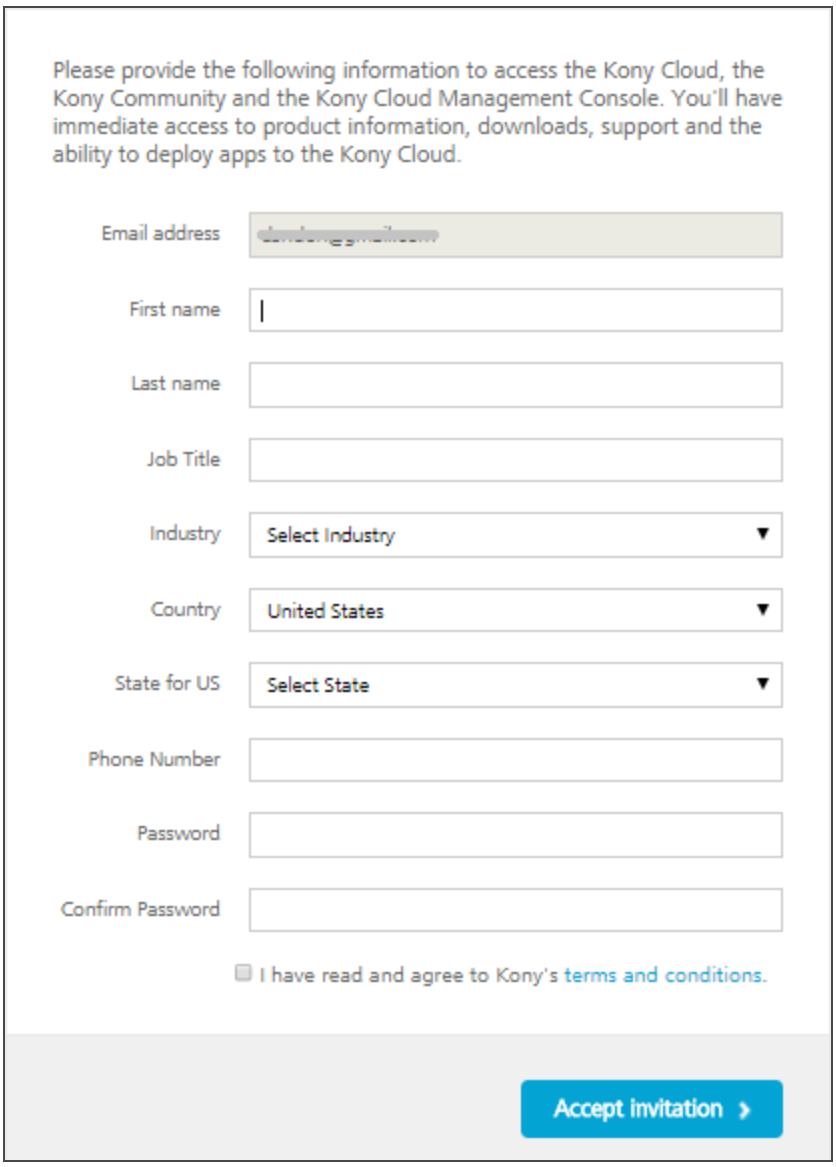
   > ***Note:*** The user can access all Dev Portals in Kony account that they have been granted access.

4. Click **INVITE**.

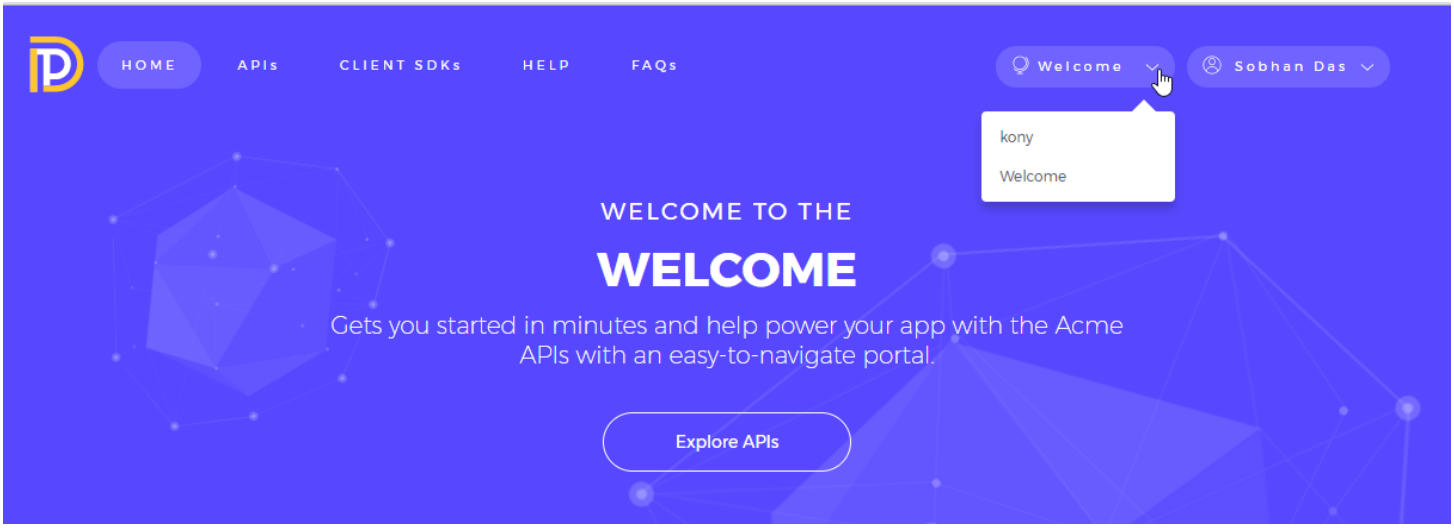
The user will receive an email notification of access, along with a link to the Dev Portal.

27.4 Using the Developer Portal

After a user has been added, they can click the link in their email invitation to use the portal. Enter the required details and click **Accept Invitation**.



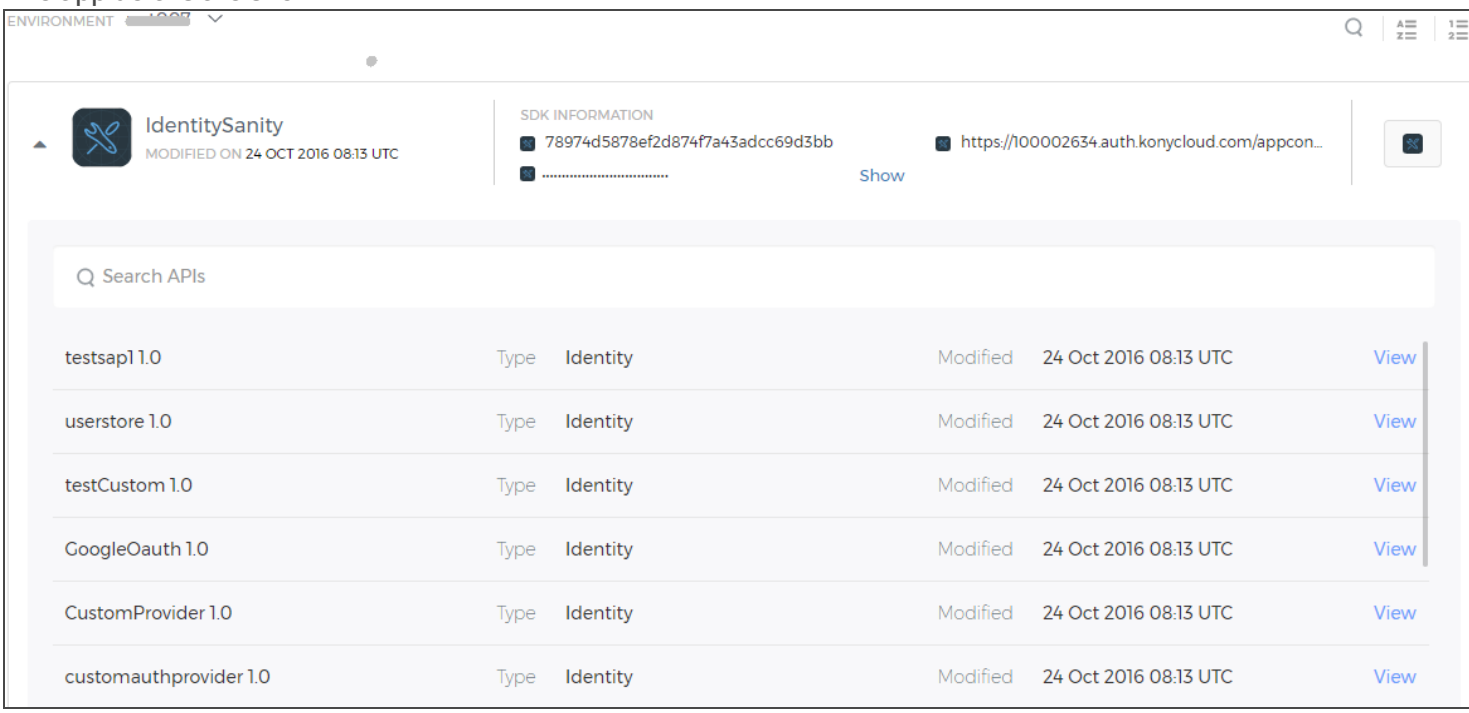
Users can have access to multiple Dev Portals. A user can switch from one Dev Portal to another by clicking the Dev Portal drop-down box and selecting the portal they want to switch to.



Apps are published per environment, so you must ensure that you have the correct environment selected in the portal. To change the environment, click the **Environment** drop-down box and select the correct environment where the app is located.

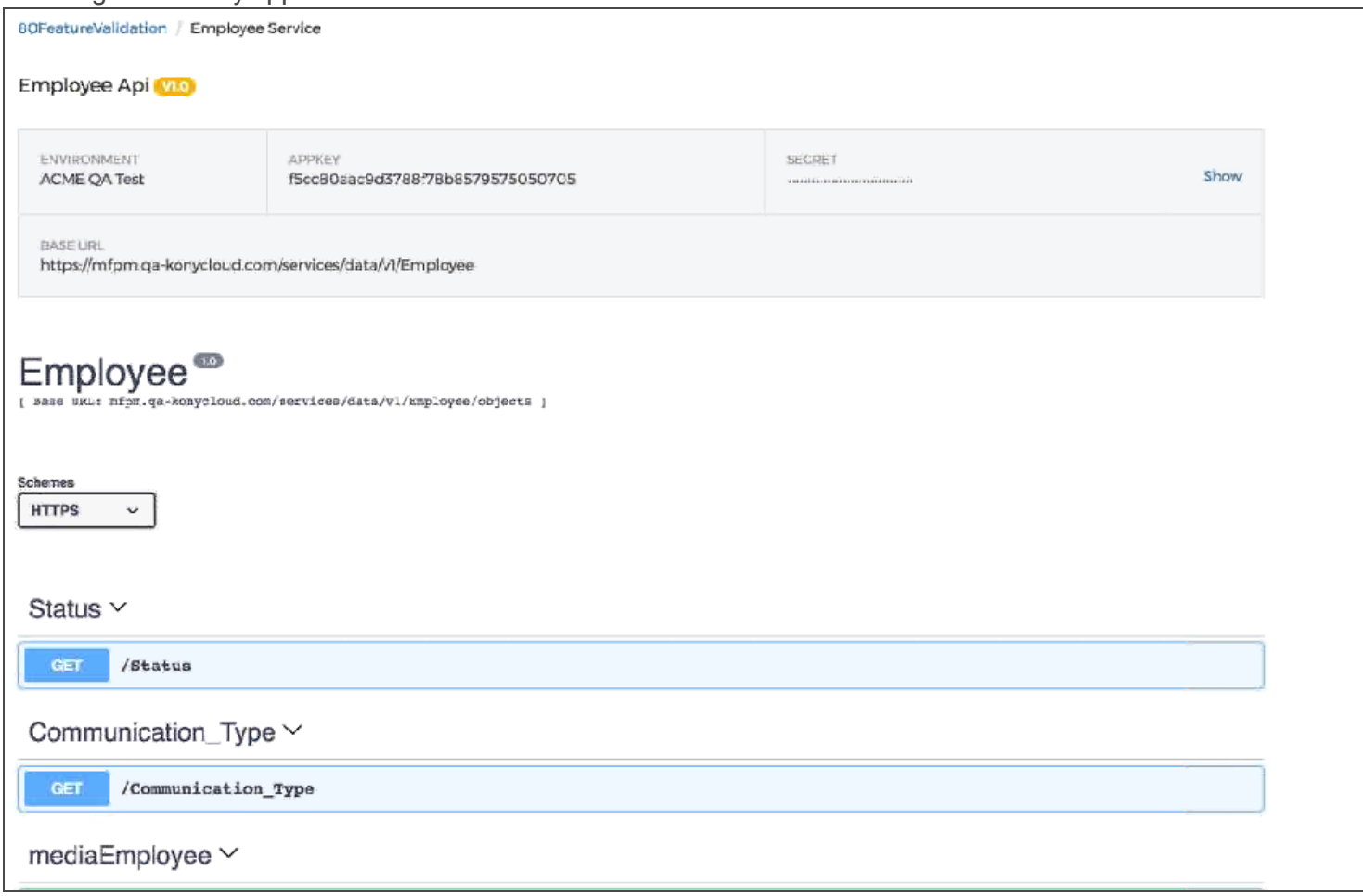
You can view details of a particular app by clicking on it.

The app details are shown.



To view a specific service, click **View**.

After a service is selected, the portal retrieves details about all exposed APIs for that service. The following screen only appears for V8 or above Clouds.

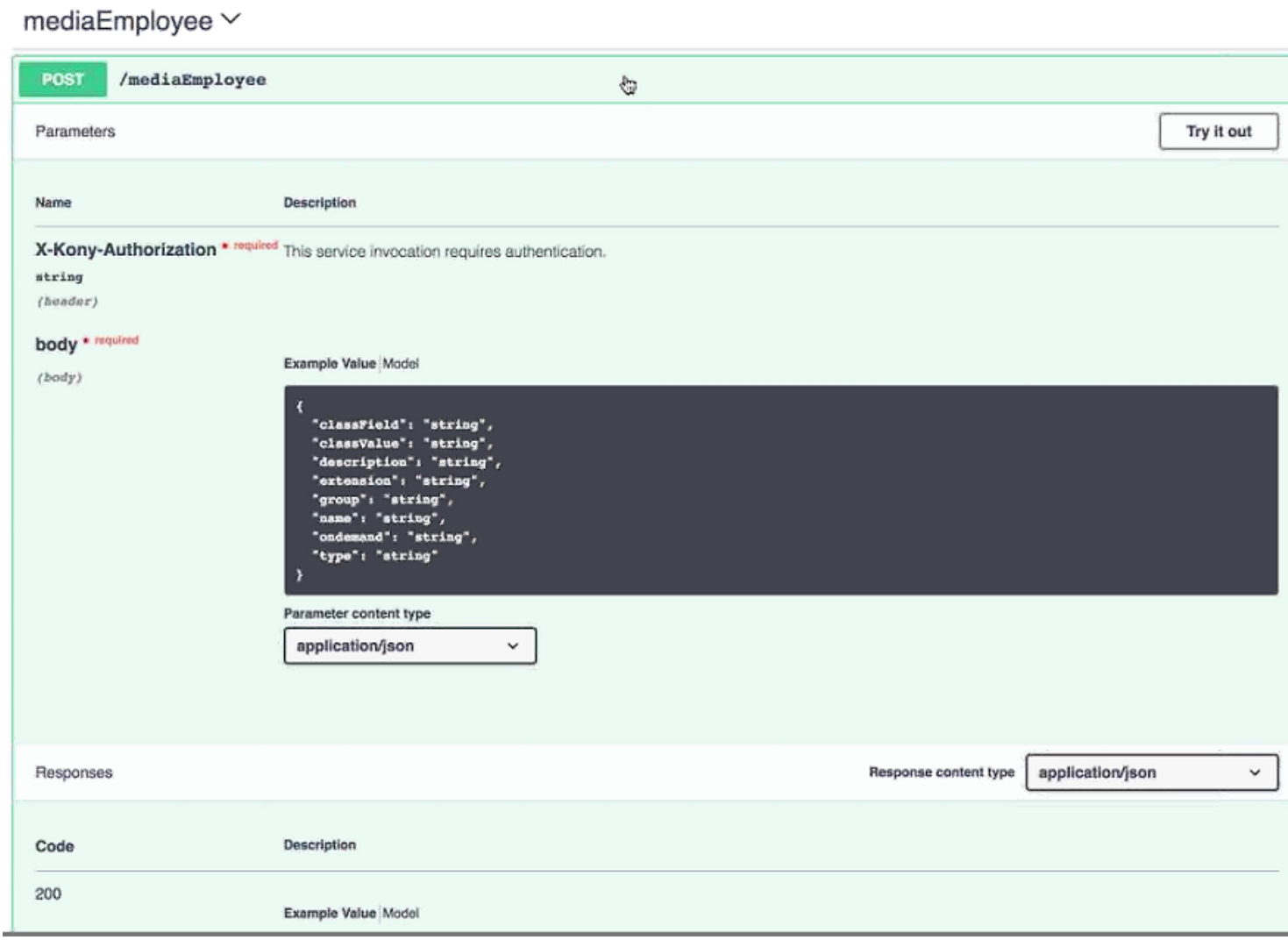


APIs enabled with Custom Front-end URLs

From Kony Fabric V8 SP3 onwards, if you have enabled a front-end URL in Kony Fabric and if the front-end URL has a value, the value is displayed as path (as per the swagger) in Developer Portal, as shown in the following table:

Front-end configured with Resource HTTP method	APIs displays the following additional details in parameters section
GET method	Parameters • **Path** for the configured front-end URL verb, which is one of the request parameters. • **Query** parameters of the request parameters. Responses
POST/ PUT methods	Parameters • **Path** for the configured front-end URL verb, which is one of the request parameters. • **Body** contains request input parameters in a JSON format Responses
DELETE method	Parameters • **Path** for the configured front-end URL verb, which is one of the request parameters. Responses

You can click on an API to get more detailed information.



Each API lists its corresponding parameters, and example values, and response codes for the API.

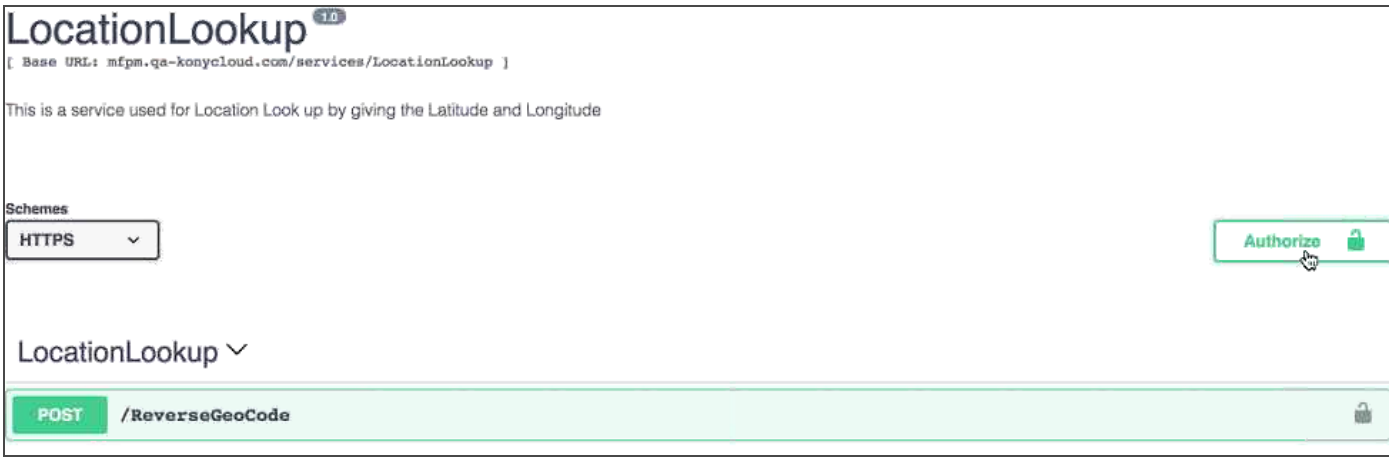
To use the API, click **Try It Out**. You can then modify the values for the parameters as needed. When you want to test the API, click **Execute**.

27.4.0.1 Securing an API using an Identity Service (Invoking an API protected by identity service)

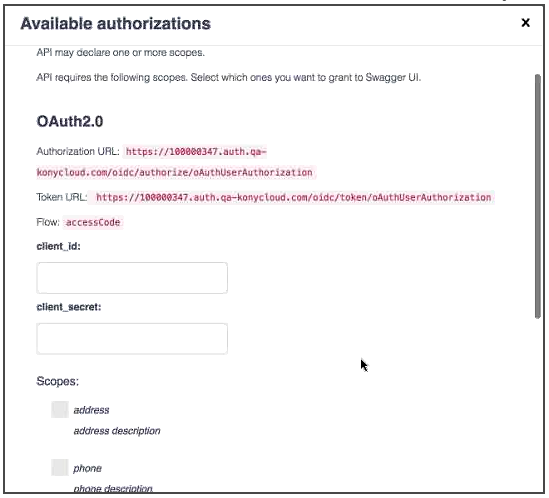
Some APIs require authorization through an identity service to use and test. These correspond to service that has its Operation Security Level set to **Authenticated App User**.

To test an API , which uses an identity service of type OAuth 2.0, do the following:

1. Click the **Authorize** button.

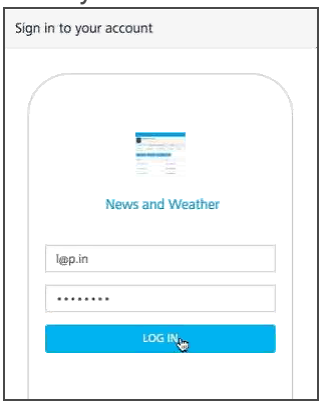


2. In the Available Authorizations window, provide the **client_id** and **client_secret** information.



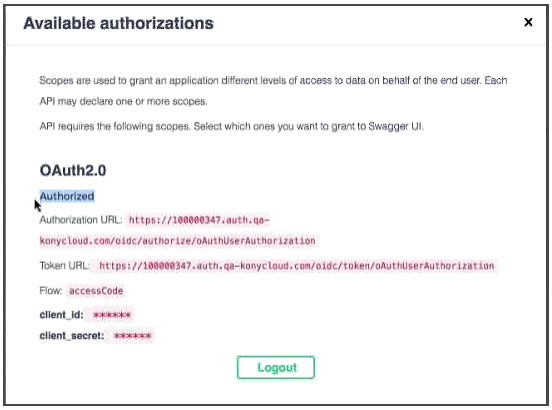
These correspond to the **AppKey** and **Secret** fields listed at the top of the API page.

3. Click the **Authorize** button.
4. Enter your credentials that correspond to the identity provider, and click **Log In**.



5. In the **Request for Authorization** window, click **Authorize**.

6. Confirm in the **Available Authorizations** window that your identity provider says **Authorized**.



28. How to Integrate Node.js Services to Kony Fabric Applications

For more details, refer to How to Integrate Node.js Services into Kony Fabric Apps.

29. Rules as a Service

Kony Fabric provides the ability to write business rules in a simplified text form by using the Rules-as-a-Service feature. The Rules service makes defining business logic closer to human language and is built using MVEL. Instead of embedding rules within Fabric integration services, with Rules as a first-class service, the business logic and conditions are externalized and can be managed separately. For example, let's say in a Banking app there could be certain business rules defined to control whether a customer is eligible for a loan. However, these business rules for loan eligibility (like credit score and income level) might vary from time to time based on market conditions, bank's promotional offers, and regulatory changes. In such cases, Fabric Rules-as-a-Service provides the ideal design to abstract these business rules from other application logic and manage them separately. This also provides Rules-as-a-Service all the capability of any Fabric service including versioning, export, import across Fabric apps, API management.

You can use the **Rules** service tab to define and store your business logic as a set of rules. A collection of rules defined in a Rules service are stored in a Ruleset. For example, a Loan Ruleset can have multiple rules such as Home Loan Rule, Education Loan Rule, Vehicle Loan Rule and so on. Similar to how any Fabric service operation can be protected by Operation security levels, rules in a Ruleset can also be protected by Rule Security Level. These are: Authenticated App Users, Anonymous App Users, Public, and Private.[1]

How Rules as Service Works

---

[1]- Authenticated App Users restricts access to clients who have successfully authenticated using an Identity Service associated with the app. - Anonymous App Users allows access from trusted clients that have the required App Key and App Secret. - Authentication via an Identity Service is not required.Public (All Users) allows any client to invoke this rule without any authentication. This setting provides no security for invoking this rule and should be avoided if possible. - Private (Internal Server only) blocks access to this rule from any external client. It allows invocations only from within the same runtime environment either from an Orchestration/Object Service, or from custom code.

Rules in a Rules service are not dependent on any back-end system and are executed within the Kony App Server. So, rules defined as a service are dependent between a client app and a Fabric app. The rules can be changed for a particular app and republish the app as required. Based on request input parameters in a request sent to a Fabric app, a set of conditions in a particular rule will be evaluated and the corresponding action will be performed whenever the condition matches. And, then Kony App Server sends the response to the client.

Why and When to Use Rules-as-a-service

When you are creating an application that has a significant portion of business logic/rules that need to be managed/modified from time to time, define those business logic as separate rules service in your Fabric application. The following are some of the sample use cases where you can use Rules-as-a-service:

- For Banks and financial institutions when they want to develop lending apps - to abstract Rules determining Loan eligibility
- For Retail and e-commerce institutions when they want to separate out promotional campaigns - which can then be modified seasonally
- For Insurance providers when they evaluate if a potential new customer meets eligibility requirements

Use Case:

A bank app with a set of rules defined as a Rules service in a Fabric app is published. A user of this client app sends request to the app to check loan eligibility. The request may contain input params, which can be used to evaluate the condition in Rules, for example, the condition is: `Credit Rating between 300 and 669, credit length less than 5`. Kony App Server executes the logic defined in the rules service in the app based on the input params (`creditRating`and `creditLengthInYears`), and then sends the desired response.

The following table details client requests and Fabric Server responses executed based on a sample rule.

<table>
<tr><th>Step 1

Client App

A user sends requests with Input Params to the published Fabric app as follows:</th><th colspan="2">Step 2

Fabric App with Rules Service published to Kony App Server</th></tr>
<tr><th></th><th>a. Kony App Server executes the logic defined in the Rules service</th><th>b. Kony App Server sends the Response to the client app</th></tr>
<tr><td>creditRating = 300 or 669
creditLengthInYears = 5</td><td>Sample rule
<pre>name: "Credit Rating between 300 and 669, credit length less than 5 "
description: "Credit Rating between 300 and 669, credit length less than 5 "
condition: "creditRating >= 300 && creditRating <= 669 && creditLengthInYears <= 5"
actions:
- "results.addParam(\"status\", \"Reject\")"
---
name: "Credit Rating between 300 and 669, credit greater than 5 "
description: "Credit Rating between 300 and 669, credit greater than 5 "
condition: "creditRating >= 300 && creditRating <= 669 &&</pre></td><td><pre>{
"opstatus":0,"status" : "Reject"
}</pre></td></tr>
</table>

<table>
<tr><td>Step 1

Client App

A user sends requests with Input Params to the published Fabric app as follows:</td><td colspan="2">Step 2

Fabric App with Rules Service published to Kony App Server</td></tr>
<tr><td></td><td>a. Kony App Server executes the logic defined in the Rules service</td><td>b. Kony App Server sends the Response to the client app</td></tr>
<tr><td>creditRating = 300 or 669
creditLengthInYears = 6</td><td rowspan="3"></td><td><pre>{
"opstatus":0,"statu
s" : "Review"
}</pre></td></tr>
<tr><td>creditRating = 670 or 740
creditLengthInYears = 5 or 6</td><td><pre>{
"interestRate": "10"
"opstatus":0,"statu
s" : "Approve"
}</pre></td></tr>
<tr><td>creditRating = 750 or 880
creditLengthInYears = 5 or 6</td><td><pre>{
"interestRate": "5"
"opstatus":0,"statu
s" : "Approve"
}</pre></td></tr>
</table>

What is the Structure of Rules: Rules have a structure in the form of statements, as shown in the following table:

Sample Rules Structure

```
name: "<Name of the rule>"
description: "<Description of the rule>"
priority: <Priority of the rule>
condition: "<Condition to evaluate>"
actions:
   - "<Set of actions to execute>"
```

Description of Rules Structure

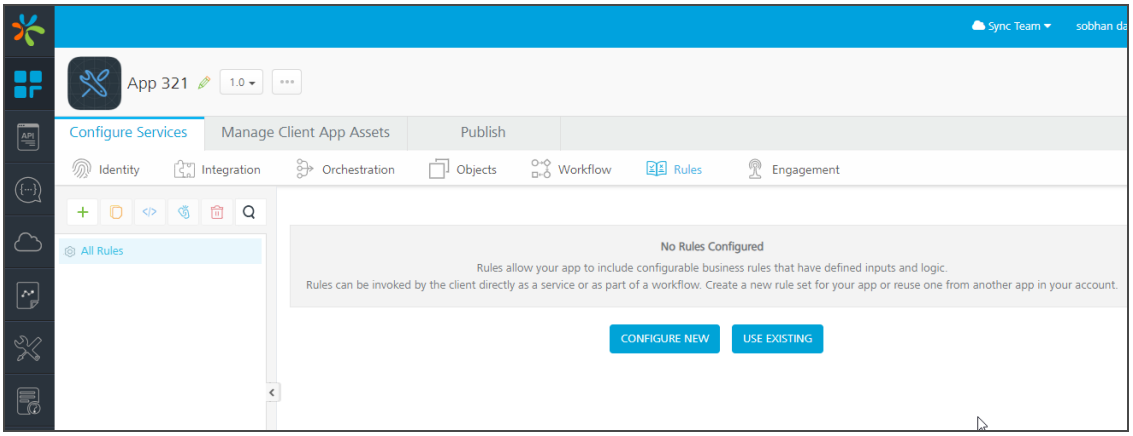
- **Name:** A unique name of the rule. This is a mandatory field.
- **Description:** A description for the rule.
- **Priority**: An integer value that represents the order to execute the rule. The bigger the value, the higher the priority.
- **Condition**: An expression that is evaluated by the Rules engine. When the condition evaluates to True, the engine executes a set of actions. This is a mandatory field.
  For example, `response != null` can be used to check whether the back-end response is empty.
- **Action**: A set of statements that are executed when the condition evaluates to True. This is a mandatory field.
  For example, `statusCode = 200` sets status code to 200.

29.1 How to Create a Rules Service

To go to the **Rules** service tab from the Kony Fabric Console dashboard, click **Add New** or select any existing Kony Fabric app, and click the **Rules** tab. The Rules tab landing page appears. Creating a rules service involves two stages, a ruleset and rules. Rules defined in a Rules service are stored in a rules-set. A Ruleset is a collection of rules.

1. Create the**Service definition** for a **Rules** service.

    a. Click **CONFIGURE NEW** to create a ruleset. The following details are displayed in the Rules service designer.



    b. In the **Name** field, provide a unique name for your ruleset.

    c. For additional configuration of your ruleset definition, provide the following details in the **Advanced** section:

Field	Description
Throttling	API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, it will not return the service response. • **To specify throttling in Kony Fabric Console, follow these steps:** i. In the **Total Rate Limit** text box, enter a required value. With this value, you can limit the number of requests configured in your Kony Fabric console in terms of Total Rate Limit. ii. In the **Rate Limit Per IP** text box, enter a required value. With this value, you can limit the number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit. • **To override throttling in App Services Console, refer to** Override API Throttling Configuration.

> ***Note:*** All options in the Advanced section are optional.

d. Click **SAVE & ADD RULE** to save the rules definition (rules-set). A **Rule List** tab appears.

2. Create **Rules** in a Ruleset.
    a. In the **Rules List**, click **ADD RULE** to add a new, if required.
    b. Provide the following details to create a Rule:

Field	Description
Name	The rule name appears in the Name field. You can modify the name, if required.

Field	Description
Rule Security Level	The Rule Security Level specifies how the client must authenticate for invoking this rule Select one of the following security operations in the **Rule Security Level** field. • **Authenticated App Users** restricts access to clients who have successfully authenticated using an Identity Service associated with the app. • **Anonymous App Users** allows access from trusted clients that have the required App Key and App Secret. Authentication via an Identity Service is not required. • **Public (All Users)** allows any client to invoke this rule without any authentication. This setting provides no security for invoking this rule and should be avoided if possible. • **Private (Internal Server only)** blocks access to this rule from any external client. It allows invocations only from within the same runtime environment either from an Orchestration/Object Service, or from custom code. ***Note:*** The field is set to Authenticated App User, by default.

c. Configure your business rules logic in the **Rule Logic** text field. This field contains sample MVEL rule logic. You can modify as per your business requirement.

d. For additional configuration of request (or) response rules, provide the following details in the **Advanced** section.

Additional Configuration Properties	Additional Configuration Properties allows you to configure service call time out cache response. For information on different types of configuration properties, refer Properties.
Front-end API	Front-end API allows you map your endpoint (or) backend URL of an operation to a front-end URL. For detailed information, refer Custom Front-end URL.
Server Events	Using Server Events you can configure this service to trigger or process server side events. For detailed information, refer Server Events.

> ***Note:*** All options in the **Advanced** section for operations are optional.

e. Enter the **Description** for the operation.

f. Click **SAVE RULE**. Now, you can configure input request and response output for your rule. The following sections detail how to create input and output operations for rules.

29.2 Configure Request for a Rule

1. In the **Request Input > Body** tab, do the following:
   a. Click **Add Parameter** button to create new entries for the input.

> ***Note:*** - To make duplicate entries, select the check box for the entry, click Copy, and then click Paste.
>
> - To delete an entry, select the check box for an entry, and then click the Delete button.

b. Configure parameters in the client's body, do the following:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.
Value	Select Request or Session. It is set to **Request** by default. i. **Request** indicates that the value must be retrieved from the HTTP request received from the mobile device. ii. **Session** indicates that the value must be retrieved from the HTTP session stored on Kony Fabric. iii. **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services.
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.

Field	Description
Datatype	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit • **Date** - • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Encode	Select the check box to enable encoding of an input parameter. For example, the name New York Times would be encoded as *New%20York%20Times* when the encoding is set to True. The encoding must also adhere to the HTML URL encoding standards.
Description	Provide a suitable description.

2. In the **Request Input > Header** tab, do the following:
    a. Click **Add Parameter** button to create new entries for the input.

> ***Note:*** - To make duplicate entries, select the check box for the entry, click Copy, and then click Paste.
>
> - To delete an entry, select the check box for an entry, and then click the Delete button.

b. Configure parameters in the client's header, do the following:

Field	Description
Name	It Contains a Unique Identifier. Change the name if required.

Field	Description
Value	Select Request or Session. It is set to **Request** by default. By default, this field is set to **Request.** Five different options are available in Kony Fabric under **Request Input > Headers** > **VALUE** during configuration of any operation. When you start editing this field, dependent identity services are auto populated. These options primarily determine the source of the value of the header**.** • **Request**: If this option is selected, the Integration Server picks the value pairs from the client's request during run time and forwards the same to the back-end. User has the option to configure the default value. This default value is taken if the request does not have the header. • **Session**: If this option is selected, the value of header is picked from session context based on the user configuration. • **Constant**: Constant is used to configure the value that is picked and sent to back-end by the Integration Server during the run-time. • **Expression**: Select this option to configure the velocity template expressions for the header values. You cannot edit the default value for expression. • **Identity**: If this is selected, you can filter the request parameters based on the response from the identity provider. For more details to configure identity filters, refer to Enhanced Identity Filters - Integration Services. ***Note:*** If the header value is scoped as a **Request** (or) **Session** and the same header is accessed under the **Expression** header value, then the expression must be represented as $request.header (or) $session.header. **Example**: If a header 1 value is a request and header 2 value is an expression, then the value of the expression must be $Request.header1.

Field	Description
TEST VALUE	Enter a value. A test value is used for testing the service.
DEFAULT VALUE	Enter the value, if required. The default value will be used if the test value is empty.
Datatype	Select one of the following data types. • **String** - A combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit. • **Boolean** - A value that can be true or false. • **Number** - An integer or a floating number. • **Collection** - A group of data, also referred as data set.
Description	Provide a suitable description.

3. Click **SAVE RULE** to save the rule. The system updates the rule's definition.

29.3 Create Response for a Rule

1. Click the **Response Output** tab, and enter the values for required fields such as name, path, scope, data type, collection ID, record ID, format, format value, and description.

> ***Note:*** If you define parameters inside a record as the session, the session scope will not get reflected for the parameters.

2. Click **SAVE RULE** to save the rule. The system updates the rule definition.

   If you click **Cancel**, the **Edit Service Parameters** window will close without saving any information.

> ***Note:*** You can view the service in the Data Panel feature of Kony Visualizer. By using the Data Panel, you can link back-end data services to your application UI elements seamlessly with low-code to no code. For more information on Data Panel, click here.

> ***Note:*** By using Visualizer SDKs, you can invoke the Rules in a Ruleset, similar to integration services. For more details on Fabric Integration Service SDKs, refer to Visualizer SDK > Invoking an Integration Service

29.4 Built-in Objects

The following objects help you to write rules in Kony Fabric.

Objects	Description
"configurationParameters"	Used to access the Server and Client App parameters that are set by the developer in the App Services console. This is equivalent to using ConfigurableParameters in Java.
"customMetrics"	This is used to access custom metrics. For more details to create custom reports and Metrics, refer Custom Reporting - Metrics, Reports, and Dashboard Guide
“deviceHeadersMap”	Used to set headers that are passed to the client and is equivalent to using setDeviceHeaders in Java.

Objects	Description
"headerMap"	Used to access the header map of a request. A client can directly access the header map or the individual key-value pairs of the header map.
“identityHandler"	Used to access the identity attributes when a service is protected by an identity service.
"inputMap"	Used to access the input map of a request. A client can directly access the input map or the individual key-value pairs of the input map.
"logger"	Used to add a log statement with the appropriate level.
"response"	Used to modify the response body and is equivalent to using setResponse in Java.
"results"	Used to modify the results. The Result is a collection of Params, Data-sets, and Records. For more details, refer Result.
"resultCache"	Used to perfom read/write in the cache. This is equivalent to using ResultCache in Java.
"servicesManager"	Used to invoke any service with the specified service id, operation id and version.
"session"	Used to modify the session that is associated with the request. For more details, refer Session A client can access values from the session and the individual attributes of the session.
“statusCode”	Used to set the status code of the response and is equivalent to using setStatusCode in Java.
"ua"	Used to access the User Agent Header of the request.

29.5 Built-in Functions

The following functions help you to write rules in Kony Fabric.

Functions	Description

"Check.isWithin"	Checks if an element is in a specified range. It will return true if the element present in the specified range, otherwise false.
	• Signature: `isWithin(double fromInclusive, double toInclusive, double elementToFind)`
	• Example
	`Check.isWithin(100, 300, 250) = true`
	`Check.isWithin(100, 300, 350) = false`

"Check.isEmpty"	Checks if a CharSequence is empty ("") or null.
	• Signature: `isEmpty(final CharSequence cs)`
	• Example
	`Check.isEmpty(null) = true`
	`Check.isEmpty("") = true`
	`Check.isEmpty(" ") = false`
	`Check.isEmpty("xyz") = false`
	`Check.isEmpty(" abc ") = false`

Functions	Description

"Check.isNotEmpty"	Checks if a CharSequence is not empty ("") and not null.
	• Signature: `isNotEmpty(final CharSequence cs)`
	• Example
	`Check.isNotEmpty(null) = false`
	`Check.isNotEmpty("") = false`
	`Check.isNotEmpty(" ") = true`
	`Check.isNotEmpty("xyz") = true`
	`Check.isNotEmpty(" abc ") = true`

"Check.isBlank"	Checks if a CharSequence is empty (""), null or white-space only.
	• Signature: `isBlank(final CharSequence cs)`
	• Example
	`Check.isBlank(null) = true`
	`Check.isBlank("") = true`
	`Check.isBlank(" ") = true`
	`Check.isBlank("xyz") = false`
	`Check.isBlank(" abc ") = false`

Functions	Description

"Check.isNotBlank"	Checks if a CharSequence is not empty (""), not null and not white-space only.
	• Signature: `isNotBlank(final CharSequence cs)`
	• Example
	`Check.isNotBlank(null) = false`
	`Check.isNotBlank("") = false`
	`Check.isNotBlank(" ") = false`
	`Check.isNotBlank("xyz") = true`
	`Check.isNotBlank(" abc ") = true`

"Check.isEqualTo"	Compares two CharSequences, returning true if they represent equal sequences of characters.
	• Signature: `isEqualTo(final CharSequence cs1, final CharSequence cs2)`
	• Example
	`Check.isEqualTo(null, null) = true`
	`Check.isEqualTo(null, "abc") = false`
	`Check.isEqualTo("abc", null) = false`
	`Check.isEqualTo("abc", "abc") = true`
	`Check.isEqualTo("abc", "ABC") = false`

Functions	Description

"Check.isEqualToIgnoringCase"	Compares two CharSequences, returning true if they represent equal sequences of characters, ignoring case.
	• Signature: `isEqualToIgnoringCase(final CharSequence str1, final CharSequence str2)`
	• Example
	`Check.isEqualToIgnoringCase(null, null) = true`
	`Check.isEqualToIgnoringCase(null, "abc") = false`
	`Check.isEqualToIgnoringCase("abc", null) = false`
	`Check.isEqualToIgnoringCase("abc", "abc") = true`
	`Check.isEqualToIgnoringCase("abc", "ABC") = true`

29.6 Sample Rules

Modify request input

Use Case	Changing request input before evaluating the rules. For example, you can map the account type received from the request to an account code.

Rule	`name: "Convert account type to account code in pre-processor"`
	`description: "Rule to convert account type to account code"`
	`condition: "AccountType == \"Loan Account\""`
	`actions:`
	`- "inputMap.AccountCode = 1"`
	`- "inputMap.Message = \"This is a loan account\""`
	
	The given sample rules above checks the account type, if the account type is Loan Account, then the associated account code is set to 1.
	
	The `inputMap` object is used to access the parameters in the request that comes from the device.

Modify result

Use Case	Modifying the result of an operation. For example, you can add the account type in the result depending upon the account code.

Rule	`name: "Add parameter in result"`
	`description: "Add a parameter in result for a specific account type"`
	`condition: "AccountCode == 1"`
	`actions:`
	`- "results.addParam(\"AccountType\", \"Loan Account\")"`
	
	The given sample rule checks the account code, if the account code is equal to 1, then the account type parameter is set as Loan Account.
	
	The `results` object is used to modify the result of an operation.

Modify response, status code and headers sent to a device

Use Case	Changing the response body, status code, and headers that are sent to the device. For example, you can categorize the customer based on the quarterly average balance and send specific headers to the device to render appropriate UX of client application.

Rule	`name: "Modify response in rules."`
	`description: "Set response message, status code and headers sent to device."`
	`condition: "quarterlyAvgBalance >= 100000"`
	`actions:`
	`- "response = \"{\\\"category\\\": \\\"Preferred customer\\\"}\""`
	`- "statusCode = 200"`
	`- "deviceHeadersMap.put(\"X-Kony-Preferred-Customer\", \"true\")"`
	
	The given sample rule checks the quarterlyAvgBalance parameter. If the parameter is greater than or equal to 100000, then the response body, status code and headers sent to the device are changed.
	
	The `response` object is used to modify the response body.
	
	The `statusCode` object is used to set the status code.
	
	The `deviceHeadersMap` object is used to modify headers that are sent to the device.

Access cache

Use case	Accessing data from the cache. For example, you can populate a country code in the cache if it is not present.

Rule	``` name: "Access cache in rules" description: "Checks if country code for India is present in cache, if not it will store in the cache" priority: 1 condition: "resultCache.retrieveFromCache(\"india\") == null" actions: - "resultCache.insertIntoCache(\"india \", \"+91\")" ``` The given sample rule checks the stored value in the cache. If the cache is empty, then the country code is added to the cache. The `resultCache` object is used to access the cache.

Invoke service

Use case	Invoking any service inside a Rule. For example, you can invoke an SMS or email service based on a request parameter.

Rule	```
	--- 	
	name: "Invoke send email integration service"
	description: "Execute a service to send email if sendEmail is true in input map."
	priority: 1
	condition: "inputMap.get(\"sendEmail\") == \"true\""
	actions:
	    - servicesManager.invokeIntegration ("RulesIntegrationService", "SendEmail")
	
	--- 	
	name: "Invoke send SMS integration service"
	description: "Execute integration service to send SMS if sendSms is true in input map"
	priority: 1
	condition: "inputMap.get(\"sendSms\") == \"true\""
	actions:
	    - servicesManager.invokeIntegration ("RulesIntegrationService", "SendSms")
	``
	
	In the sample, based on the parameters sent from the device, we are invoking services either to send an SMS or email or both.
	
	The `servicesManager` object is used to invoke any service from a rule.

Access Identity data

Use case	Accessing the Identity data. For example, you can access the Identity data such as the app id and the user profile for the request.

Rule	```
	name: "Access Identity Info"
	description: "Accesses identity info like first name, last name, email id, and app id in rules"
	priority: 1
	condition: "setIdentityDetails == true"
	actions:
	    - "results.addParam(\"FirstName\", identityHandler.getUserProfile().getFirstName())"
	    - "results.addParam(\"LastName\", identityHandler.getUserProfile().getLastName())"
	    - "results.addParam(\"Email\", identityHandler.getUserProfile().getEmailId())"
	    - "results.addParam(\"AppId\", identityHandler.getAppId())"
	``
	The given sample rule accesses the identity information such as first name, last name, email id, and app id and sends it to the device.
	The `identityHandler` object is used to access identity information.

Access session

Use case	Accessing data from the session. For example, user can check user type from the value inserted in session if user is preferred one and set interest rate accordingly.

Rule	`name: "Access session data"`
	`description: "Check if authorization token is present in session then set the same in header"`
	`priority: 1`
	`condition: "session.containsKey ("\isPreferredBankingSession\")"`
	`actions:`
	`- "results.addParam(\"interestRate\", \"2\");`
	
	The given sample rule checks for if isPreferredBankingSession attribute is available in session and sets interest rate accordingly.
	
	The `session` object is used to access the session data.

Access Configurable Parameters defined in App Services

Use case	Accessing the Configurable Parameters defined in App Services.

Rule	`name: "Access configuration properties"`
	`description: "Check if encryption enabled in client properties then set encryption key in input map."`
	`priority: 1`
	`condition: "configurationParameters.getClientAppProperty (\"encrypt\") == \"true\""`
	`actions:`
	`- "inputMap.encryptionKey = configurationParameters.getServerProperty (\"encryptionKey\")"`
	
	The given sample rule checks the encrypt property. If the encrypt property is enabled in the client properties, then the code fetches the encryption key from server properties and adds the key to the request.
	
	The `configurationParameters` object is used to access the properties that are defined in App Services.

Access custom metrics

Use case	Accessing custom metrics.

Rule	<pre>name: "Access custom metrics in rules"
	description: "Set custom metrics of product if enabled in request."
	priority: 1
	condition: "enableCustomMetrics == true"
	actions:
	- "KonyCustomMetricsDataSet metricsDataset = new KonyCustomMetricsDataSet();"
	- "metricsDataset.setMetricsString(\"Product Name\", \"Kony Quantum\");"
	- "metricsDataset.setMetricsBoolean(\"Is Released\", true);"
	- "metricsDataset.setMetricsTimestamp(\"Release date\", \"2019-03-12\", \"yyyy-MM-dd\");"
	- "customMetrics.addCustomMetrics(metricsDataset);"</pre>
	The given sample rule assigns values to the custom metrics.
	
	The `customMetrics` object is used to access custom metrics.

Multiple rules in same Rule

Use case	You can write multiple rules in the same rule by using `--- (three hyphens)` as the separator. ***Note:*** Please do not give separator after last rule.

Rule	``` --- name: "Convert account code to type for loan account" description: "Rule for loan account to convert account type to code and add message in result" condition: "AccountCode == 1" actions: - "results.addParam(\"AccountType\", \"Loan Account\")" - "results.addParam(\"Message\", \"This is a Loan Account\")" --- name: "Convert account code to type for saving account" description: "Rule for saving account to convert account type to code and add message in result" condition: "AccountCode == 2" actions: - "results.addParam(\"AccountType\", \"Saving Account\")" - "results.addParam(\"Message\", \"This is a Saving Account\")" --- name: "Convert account code to type for current account" description: "Rule for current account to convert account type to code and add message in result" condition: "AccountCode == 3" actions: - "results.addParam(\"AccountType\", \"Current Account\")" - "results.addParam(\"Message\", \"This is a Current Account\")" ``` The given sample rule invokes multiple rules.

Iterate over a set of values

Use case	Iterating over a set of values.

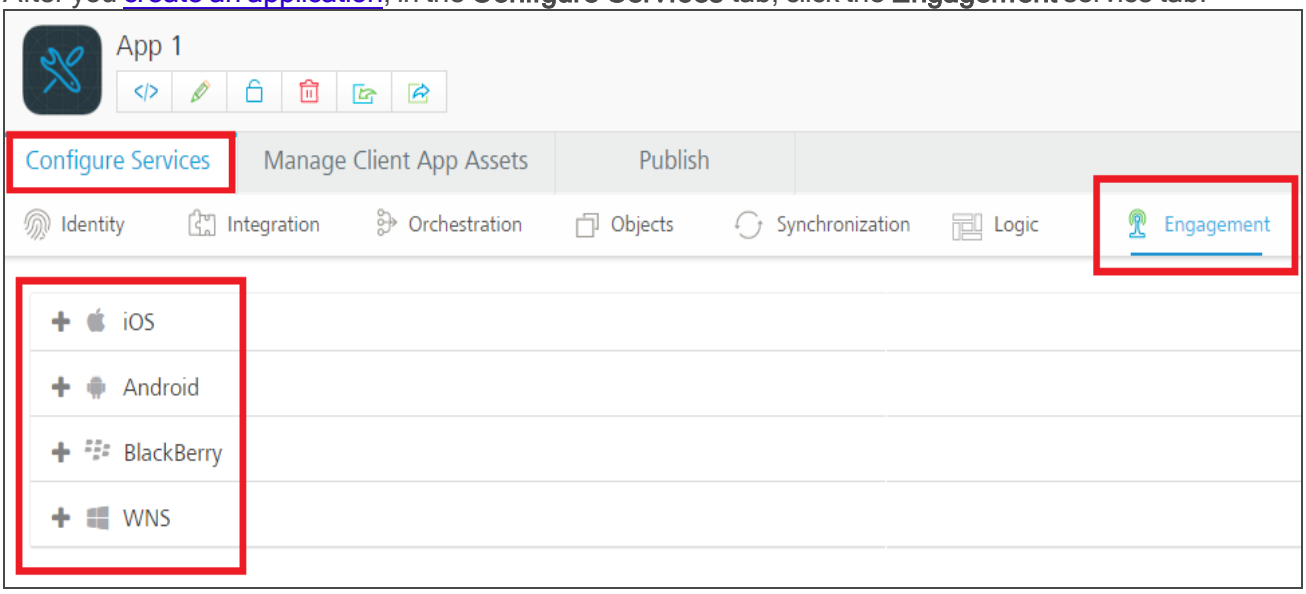
Rules	<pre>name: "Iterate over records using for loop in rules."
	description: "Add 80% of price as discounted price of book."
	condition: "\"giveDiscount\" == true"
	actions:
	- "for (Record record : results.getDatasetById (\"books\").getAllRecords()) {
	Record bookRecord = record.getRecordById(\"book\");
	double price = Double.parseDouble (bookRecord.getParamValueByName(\"price\"));
	bookRecord.addParam(\"discountedPrice\", price * 0.8);
	}"</pre>
	The given sample rule iterates a books dataset and adds discounted prices for each book in the dataset.

30. Engagement

Engagement service allows you to upload push certificates for iOS, Android,and Windows 8 RT platforms.

Menu path:

After you create an application, in the **Configure Services** tab, click the **Engagement** service tab.



For sending messages, follow these steps:

1. Add Push Certificates
2. Access Engagement Console
3. Send a Push Message

30.1 Add Push Certificates

Kony Fabric Engagement supports the following platforms:

1. iOS
2. Android
3. WNS

This section details the process for adding push certificates to your application.

30.1.1 iOS

> ***Note:*** Refer to the following section for creating a push certificate: Engagement Services Console User Guide > Applications

To add iOS Push Certificates for your app, follow these steps:

1. Expand **iOS**. A list of configurable items appear.
2. **Application Mode**: An appropriate application mode.
    - Production mode: When selected, production certificates and associated password details are entered while sending push notifications. Push notifications are delivered in real-time.
    - Development mode: When selected, you can still send push message notifications, but delivery of push notifications are not real-time.
3. **iPhone Push Certificate**: From here, you can upload, download, or delete a certificate.
    - Click **Browse** to upload an iPhone certificate.
    - Click **Download** to download an iPhone certificate.
    - Click **Delete** to delete an iPhone certificate.
4. **Certificate Password**: Enter the password for iPhone, and then click Save to complete the configuration process.

5. **iPad Push Certificate**: From here, you can upload, download, or delete a certificate.
    - Click **Browse** to upload an iPhone certificate.
    - Click **Download** to download an iPhone certificate.
    - Click **Delete** to delete an iPhone certificate.
6. Click **Save** to complete the configuration process for iOS platform.

30.1.2 Android

> ***Note:*** Refer to the following section for creating a push certificate: Engagement Services Console User Guide > Applications

To add Android (GCM Key/Sender ID) details or Jpush details (app key and master secret), follow these steps:

1. Expand **Android**. A list of configurable items appear.
2. Kony Engagement Services supports two options for sending push notifications to Android devices. The Google Cloud Messaging network is the recommended network, but for certain geographies such as China, JPush may be required to reliably deliver push notifications. You may select to use either network option or configure both and allow your app to specify at the time of subscription which network it will use.
    - Enter the **Google Cloud Messaging (GCM)** authorization key, and then click **Save** to complete the configuration process.

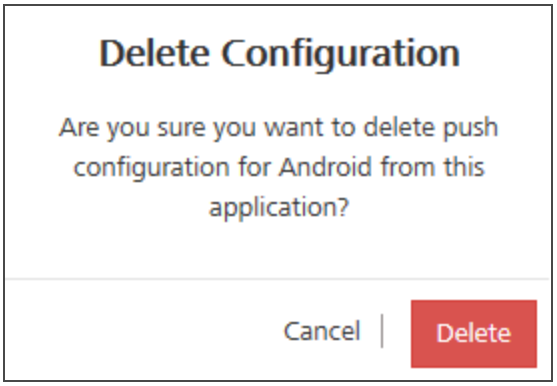
      > ***Note:*** Google Cloud Messaging for Android (GCM) is a service that helps you to send data from servers to Android applications on Android devices. This can be a lightweight message telling the Android application that there is new data to be fetched from the server (for example, a movie uploaded by a friend), or it can be a message containing up to 4kb of payload data (so apps like instant messaging can consume the message directly). The GCM service handles all aspects of queuing of

> messages and delivery to the target Android application running on the target device.
>
> Click the Create an Android GCM Key (Via Google Old Console) link for more details on how to subscribe for GCM Authorization Key.

- Enter the following details for **JPush**:
    a. **App Key**: Enter the app key.
    b. **Master Secret**: Enter the master secret.

  For more details , refer to Engagement Services Console User Guide > Android Platform > Create a JPush App Key and Master Secret

3. To delete push configuration for Android, click **Delete Configuration**.

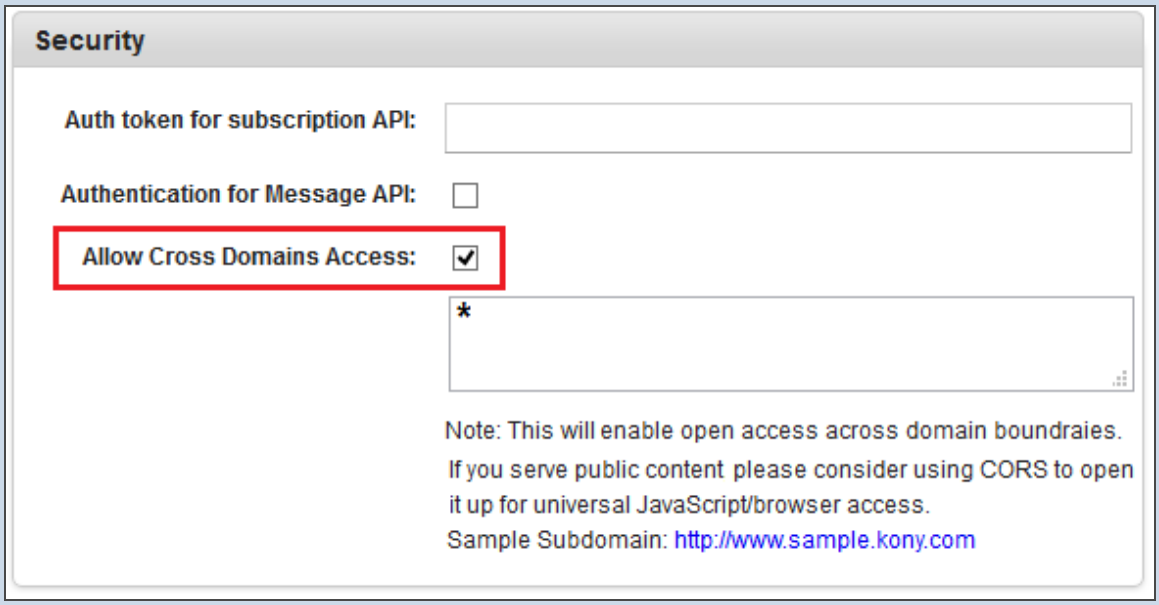


4. Click **Delete** to confirm.

> ***Important:*** From PhoneGap application, to use Engagement services (subscription, push messages and fetch messages), you must enable cross-origin resource sharing (CORS) in Kony Fabric Engagement console.
>
> To enable CORS, in **Kony Fabric Engagement Console > General > Settings > Security**, select the **Allow Cross Domains Access** check box. In Kony Fabric

Engagement Console, by default the check box is cleared.

For more details, refer to Kony Fabric Messaging Console > General > Settings > Security section.



30.1.3 WNS

Note: Refer to the following section for creating a push certificate: Engagement Services Console User Guide > Applications

Note: Windows push certificate is a purchased SSL certificate that is converted to correct format for uploading to Kony Fabric.

To add Windows Push Certificates for your app, follow these steps:

1. Expand **WNS**. A list of configurable items appear.
2. **Secret**: Enter the secret key details.

   > ***Note:*** Windows Secret is an associated secret key that contains strings used in authentication with Kony Fabric Messaging APIs. It is used in authentication on the client side during registration.

3. **SID**: Enter the SID details, and then click Save to complete the configuration process.

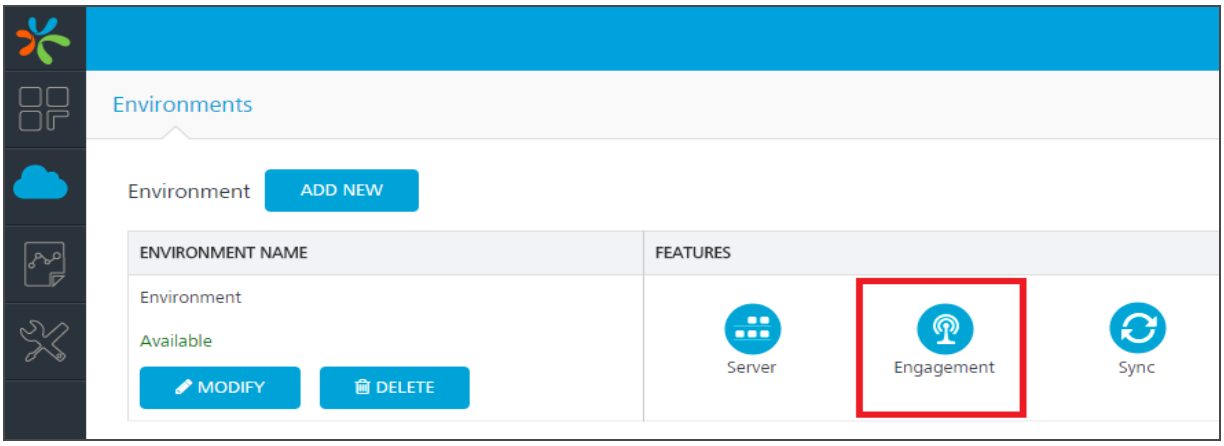
   > ***Note:*** Windows SID is a security identifier that is a unique, immutable identifier of a user, user group or other security principal. A security principal has a single SID for life, and all properties of the principal, including its name, are associated with the SID. This design allows a principal to be renamed (for example, from "John" to "Jane") without affecting the security attributes of objects that refer to the principal.

4. To delete push configuration for Windows, click **Delete Configuration**.

30.2 Accessing Engagement Services Console

Kony Engagement Services Console allows you to add and manage applications, view the stored certificates, and manage a subscribers list.

To view your Kony Engagement Services Console, click **Engagement** from your Environments.



Note: For more information on Kony Engagement Services Console, refer to the following guide: http://docs.kony.com/konylibrary/messaging/kms_console_user_guide/Default.htm.

31. Manage Client App Assets

With Manage Client App Assets (for example, Kony Management as a Service) functionality in Kony Fabric, Kony Fabric Users can now manage their client binaries through Kony Fabric Console such as creating mobile applications, publishing the apps to an EAS Environment or a Kony Management (EMM) Environment and Server. After the native client binaries are published to Kony Management environment, Kony Management admin can make the applications available to the users of an Enterprise. Currently, Kony Fabric supports more than one version of client binaries for different versions of platforms such as iOS Phone, for iOS Phone, iOS Tablet, Android Phone, Android Tablet, Windows Phone, and Web Client. The allowed version formats for binaries are `<One Digit>.<Upto 2 Digits>.<Upto 3 Digits>`. The dots and the digits after it in the version number are optional.
For example: The supported version formats for binaries are allowed for uploading such as `1`, `1.0`, `1.00.00`, `1.0.00`, and `1.23.456`.

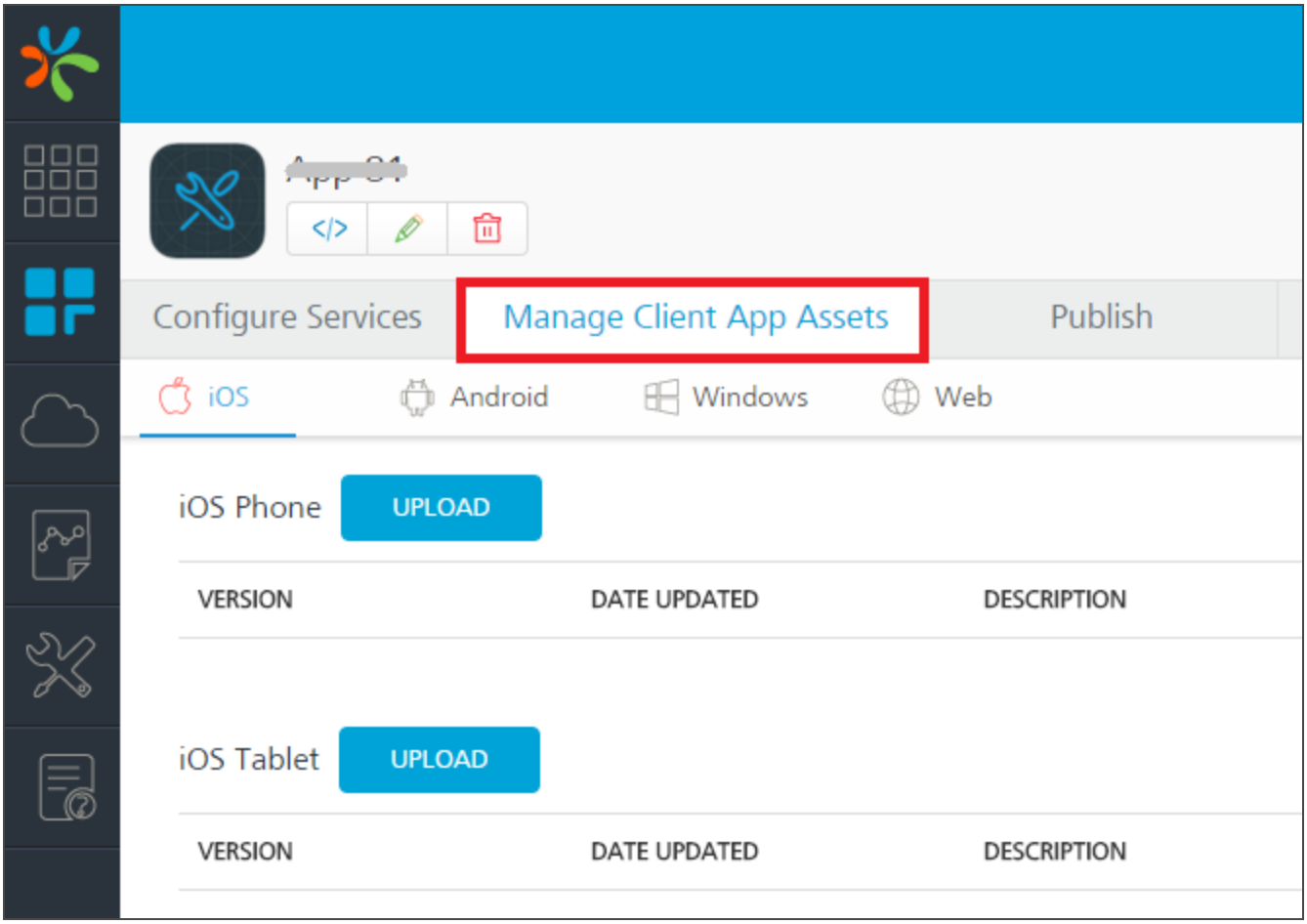
- In Kony Fabric Console, to publish native client binaries to a Kony Management Environment, first upload the required native client binaries for platforms under the **Manage Client App Assets** tab and publish these binaries to Management environment through **Publish > Native Client** tab.
- If you upload web client binaries (`.war`) for **Web** under the **Manage Client App Assets** tab, these web binaries will only be published to the server.

> ***Note:*** The database global variable `MAX_ALLOWED_PACKETS` size should be set to approximately twice the size of the binaries that you want to upload to Kony Fabric Console. Refer to FAQs > How I can increase the size limit of the Client binaries that I upload to MobileFaric Console.

> ***Note:*** You can upload maximum up to 10 Web Client binaries for an app.

Menu path for Managing Client App Assets service designer:

After you create an application, in the app configuration page, click the **Manage Client App Assets** tab to display the tabs for iOS, Android, Windows, and Web.



Managing Client App Assets involves four steps:

1. Uploading Client Binaries to Kony Fabric
2. Publishing Client Binaries from Kony Fabric
3. Publishing Native Client Binaries from Kony Management Console to Devices
4. Upgrading Native Client Binaries

31.1 Uploading Client Binaries to Kony Fabric

- Uploading Native Client Binaries to Kony Fabric
- Uploading Web Client Binaries to Kony Fabric

31.1.1 Uploading Native Client Binaries to Kony Fabric

To upload native client binaries to Kony Fabric, follow these steps:

1. In Kony Fabric Console, in the **Applications** page, click **Custom Apps** > **ADD NEW** to create an app. For more details, refer to How to Add Applications.
2. In the app configuration page, click the **Manage Client App Assets** tab.
3. For uploading native client binaries, click the required tabs for platforms such as iOS, Android, and Windows. By default, **iOS** tab is selected.
4. Click **UPLOAD** for each the device type you want upload binaries.
5. In the Upload dialog, drag the binary file or click **Browse** to locate the binary file.

   The following native client binary file formats are supported for platforms.

   - For uploading binaries for iOS Phone and iOS Tablet, select a `.ipa` file.
   - For uploading binaries for Android Phone and Android Tablet, select an `.apk` file.
   - For uploading binaries for Windows Phone, select a `.appx` or `.xap` file.
6. In the **Display Name** field, specify the name of the application. This field is optional.

   > ***Note:*** If the display name is not specified for the app binary, the published app is displayed with the default Kony Fabric app name.
7. In the **Version** field, enter the version for the binary file.

8. Upload a .png file for your app icon. The .png file scale is 120X120x. Also, you can upload screen shots for your app.
9. In the **Description** filed, enter the description for the binary file.
10. Click **UPLOAD** in the dialog. The selected binary files for each platform are uploaded to Kony Fabric Console.

    You can download and delete these binaries from the Console. To download the native client binary, click **Download**.
    To delete the uploaded binary file, click the **Delete** button. In the **Delete Client Binary** dialog, click **DELETE** to confirm the deletion.

- After you upload native client binaries in Kony Fabric Console, you can publish these binaries to an EAS Environment. For more details, refer to Publishing Native Client Binaries from Kony Fabric to Kony EAS.

- After you upload native client binaries in Kony Fabric Console, you can publish these binaries to an EMM Environment. For more details, refer to Publishing Native Client Binaries from Kony Fabric to EMM Environment.

31.1.2 Uploading Web Client Binaries to Kony Fabric

To upload Web client binaries to Kony Fabric, follow these steps:

1. In Kony Fabric Console, in the the **Fabric Apps** page, click **ADD NEW**. For more details, refer to How to Add Applications.
2. In the app configuration page, click the **Manage Client App Assets** tab.
3. Click the **Web** tab. By default, **iOS** tab is selected.
4. Click **UPLOAD**.
5. In the Upload dialog, drag the `.war` file or click **Browse** to locate the `.war` file.

6. In the **Display Name** field, specify the name of the application. This field is optional.

   > ***Note:*** If the display name is not specified for the app binary, the published app is displayed with the default Kony Fabric app name.

7. In the **Version** field, enter the version for the binary file.
8. In the **Description** filed, enter the description for the binary file.
9. Click **UPLOAD** in the dialog. The selected `.war` file is uploaded to Kony Fabric Console.

   You can download and delete these binaries from the Console. To download the native client binary, click **Download**.
   To delete the uploaded binary file, click the **Delete** button. In the **Delete Client Binary** dialog, click **DELETE** to confirm the deletion.

After you upload Web client binaries in Kony Fabric Console, you can publish these binaries to the Server. For more details, refer to How to Reconfigure an App.

31.2 Publishing Client Binaries from Kony Fabric

Using Kony Fabric Console, you can publish binaries such as native client binaries to Kony Management environment or Kony Enterprise Store and web client binaries to the Server.

The following sections help you publish client binaries:

- Publishing Native Client Binaries From Kony Fabric to EAS
- Publishing Native Client Binaries to Kony Management Environment
- Publishing Web Client Binaries to the Server

For more details on uploading client binaries, refer to Uploading Client Binaries to Kony Fabric Console.

> *Important:* Make sure that your Kony Fabric Console version used for creating apps is same as Kony Fabric runtime components' (Integration, Sync, and Messaging) versions. All the components of Kony Fabric must be upgraded to same versions.
>
> For example, if the Kony Fabric Console installed version is V8 for creating your apps, you must use the same Kony Fabric version for runtime components to publish your apps. If there is a version mismatch, Kony Fabric's Publish functionality and other runtime server components may not work as expected.

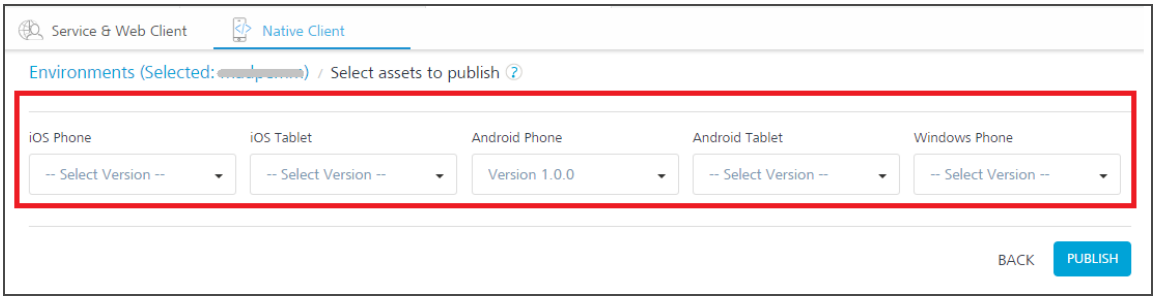
31.2.1 Publishing Native Client Binaries from Kony Fabric to Kony Management

This section details how you to publish native client binaries to Kony Enterprise Store using Kony Fabric Console. To publish native client binaries, you must create an app in Kony Fabric Console, upload the required native client binaries through the Manage Client App Assets tab, and then publish these binaries to Kony Management through **Publish** tab **> Native Client** tab.

> *Important:* The native client binaries uploaded for iOS, Android, and Windows platforms are only published to Kony Management. However native client binaries only available in Kony Enterprise Store if the publish workflow state is selected as **Active** and not Draft.
>
> In this document, Kony Management is referred to Kony Management Stage.

Limitations:

- After any version of a client binary is published, if you update the description of the client binary and re-publish the binary, the system does not update the description in Kony Management Console.
- When you select the `-- Select Version --` from the drop-down list for platforms, and click **Publish**, the system removes all versions of that client binaries if present in Kony Management runtime.



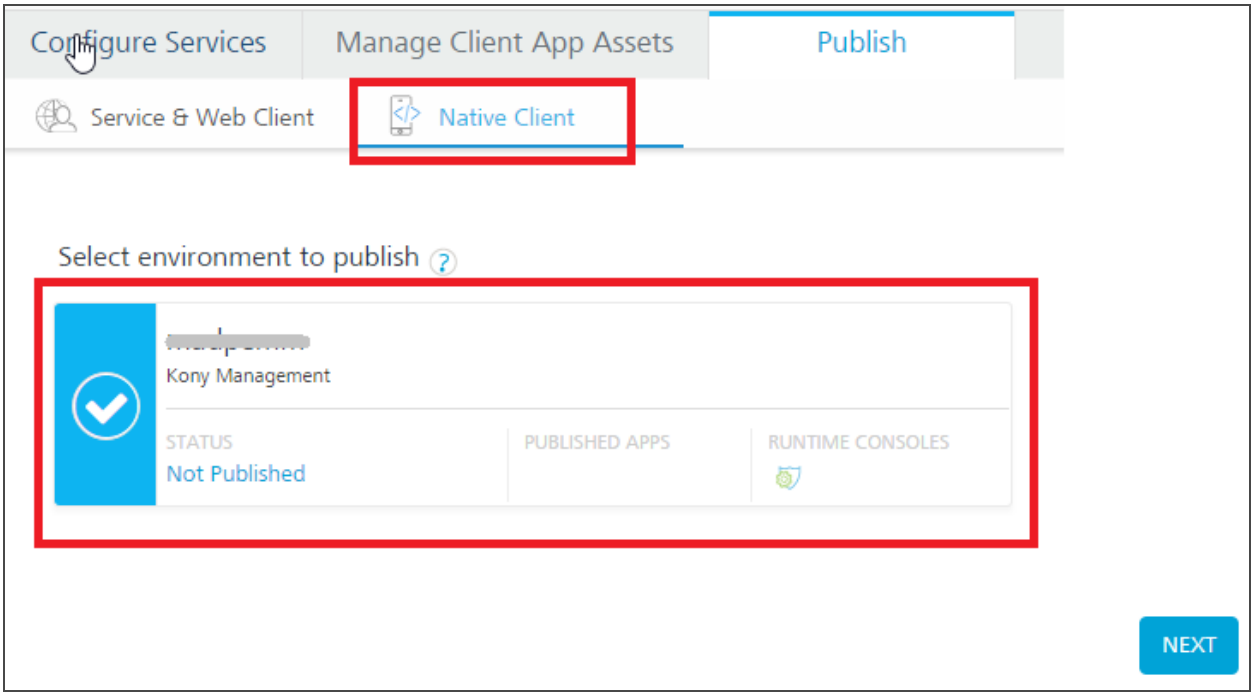
- Publish operation is cumulative. For example, if iOS Phone binaries are previously published, and now you want to publish iOS Tablet binaries, should not select the `-- Select Version --` from the iOS Phone drop-down list.
- If you delete any version of client binary after it is published, you can remove the version in Kony Management Console by one of the following two ways:
  - Delete older published versions of client binaries via Kony Management Console.
  - Select the `-- Select Version --` from platforms drop-down list and publish the app. This will remove all the versions of clients in the Kony Management runtime. Publish the versions against from lowest version.
- If a downgraded version is published to Kony Management, the same version does not get downgraded in Kony Management as Kony Fabric currently does not support downgrade of application.
- If any publish fails for native client binaries, the environment status changes to `Not Published`. Kony Fabric does not track the publish failures for native client binaries as of now.

To publish native client binaries to Kony Management, follow these steps:

1. In Kony Fabric Console, go the app for which you uploaded native client binaries.
2. Click the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected.
3. Click the **Native Client** tab. The **Native Client** tab lists clouds or environments configured for the Kony Fabric account. The list also displays the following app status for that cloud or environment.

- **Published**: An app is published to a cloud or environment. You can unpublish the app, if required.
- **Not Published**: An app is not published to a cloud or environment. You can publish the app, if required.
  An app is canceled while publishing or unpublishing. You can publish or unpublish the app, if required. Refer to Limitations:

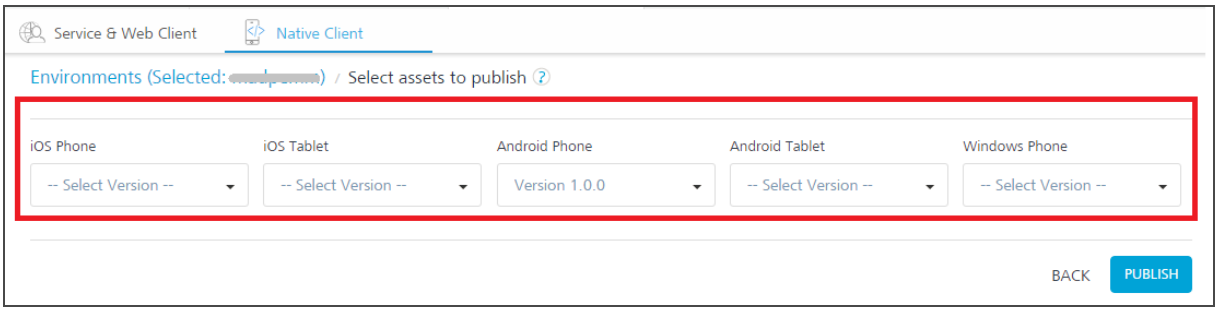
4. Under the **Select environment to publish**, click the appropriate Kony Management environment for publishing.



> ***Note:*** The **NEXT** button dims when you have not selected any environment. When an environment is selected, only then the **NEXT** button is available.

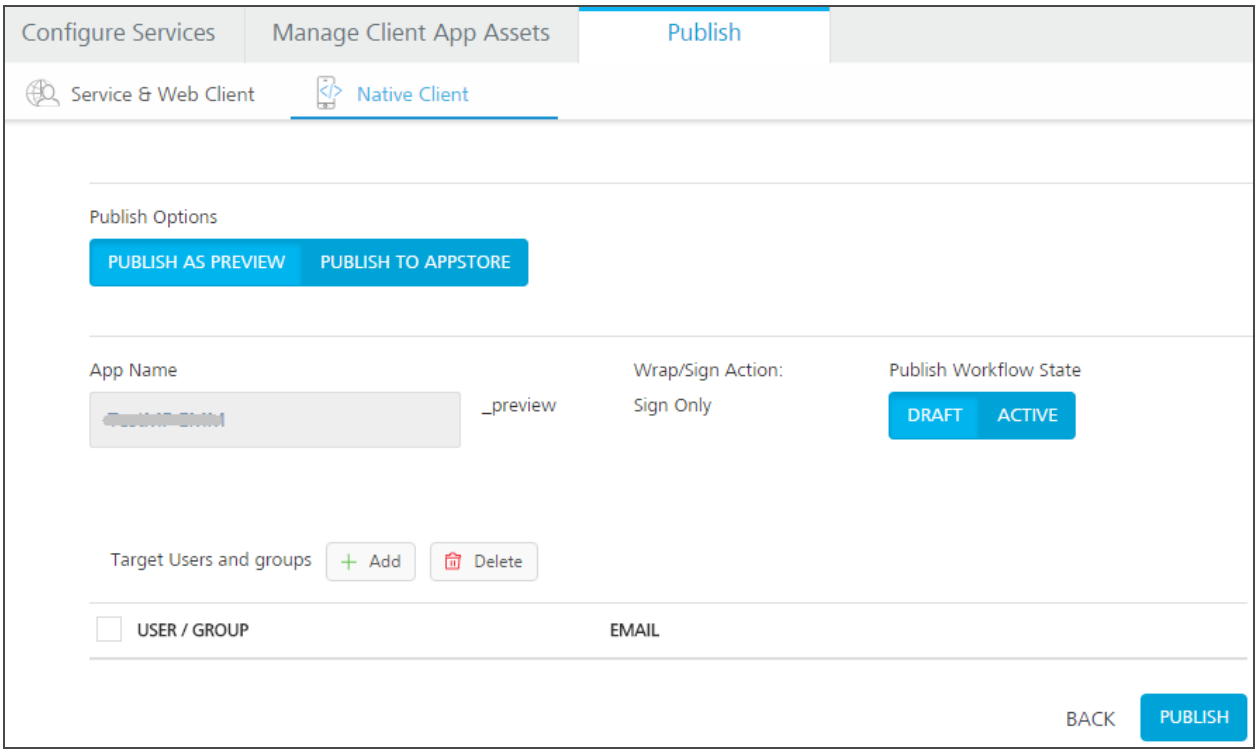
5. Click **NEXT** to select assets to publish.
   The uploaded binary versions are loaded in the drop-down list for each of the platforms.

6. Select the binary version from the platforms drop-down lists.



7. After you select binary versions for platforms, click **PUBLISH**.

   The **Publish Options** window appears. Kony Fabric Admin can choose **PUBLISH AS PREVIEW** or **PUBLISH TO APPSTORE**.



   You can publish the app binaries as preview to Kony Management directly.

> ***Important:*** The published app is available in Kony Management Console. Also the app is available in Kony Enterprise Store only if workflow state is selected as **ACTIVE**.

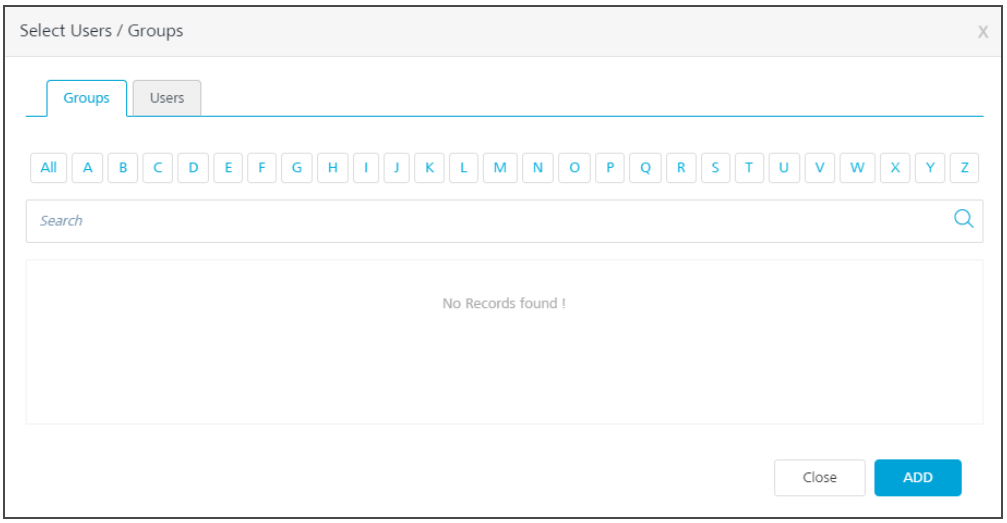
You can select Kony Management users or groups and target the native binaries to be published. When a Kony Management Admin updates these binaries and targets the uploaded versions of the app binaries, these binaries are available only to the selected users or groups. These users or groups will be notified about native app. Kony Fabric admin can target all Kony Management users or groups (Local users/groups, Microsoft Active Directory, and SAP) except cloud users. Kony Fabric admins can continuously push test versions of native apps to selected users or groups.

8. Choose one of the **publish options** as follows:
    a. To publish an app as preview, click **PUBLISH AS PREVIEW**. The system displays the current app with a suffix of `<Appname>_preview`. Preview apps are always **sign only** and can be installed on the same device alongside the production copy of the same app. These apps will be displayed as `<Appname>_preview` on user devices or spring board.
    b. To publish an app as a production copy, follow these steps:
        i. Click **PUBLISH TO APP STORE**.
        ii. Select one of the following options under **Under Wrap/Sign Action**:
            - **SIGN ONLY**: an app is set to sign only before publishing.
            - **WRAP & SIGN**: an app is wrapped and signed before publishing.

9. Select the publish state in **Publish Workflow State:**
    - **DRAFT**: If DRAFT is selected, the app is published to Kony Management Console.
    - **ACTIVE**:If ACTIVE is selected, the app is published to Kony Management Console and Kony Enterprise Store as well.

> ***Important:*** If **Publish Workflow State** is selected as **ACTIVE**, you must target users or groups before publishing. Else the publish fails.
>
> If **Publish Workflow State** is selected as **DRAFT**, Kony recommends that you target users or groups. It is an optional.

10. To targeting users or groups, follows these steps:
    a. Under **Target Users and groups**, click the **Add** button to display the **Select Users / Groups** window. The **Select Users / Groups** window displays names of users and groups from Kony Management.

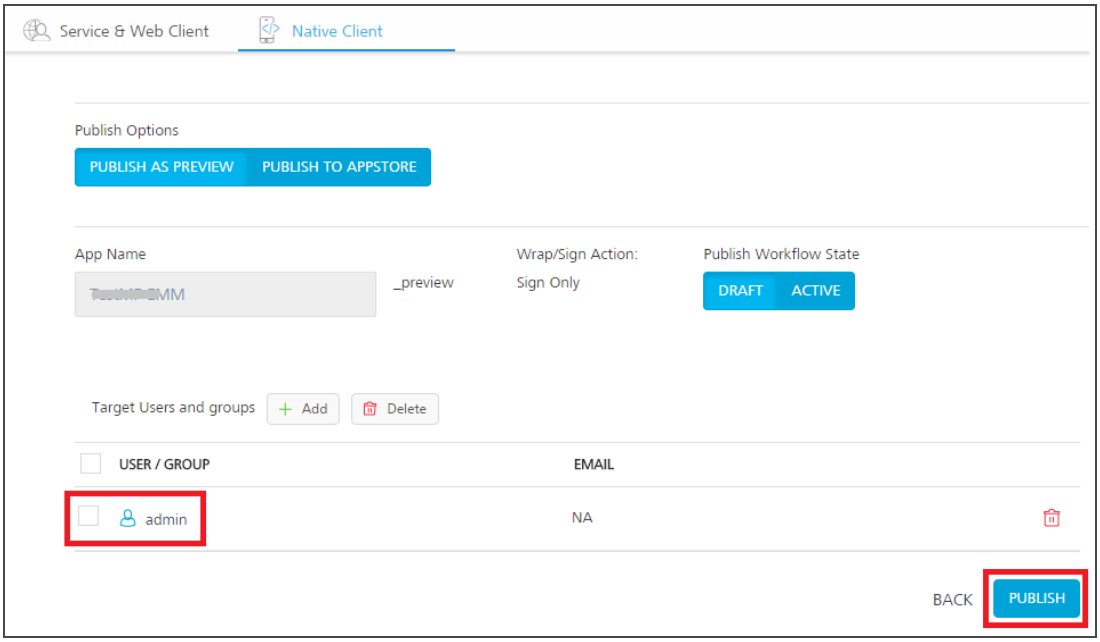


    b. Under **Select Users / Groups**, follow these steps:
        i. Add Groups and Users:
            - **Groups**: Click **Groups** and select the check boxes for the groups, as required. You can click the filter buttons (All or A, B, C) or type the name of the group for filter groups from Kony Management Environment.

- **Users**: Click **Users** and select the check boxes for the users, as required. You can click the filter buttons (All or A, B, C) or type the name of the group for filter users from Kony Management Environment.

> ***Note:*** Ensure that you do not close the publish progress dialog while the app publish is in progress.

ii. Click **ADD** to add the selected users and groups under the **Target Users and groups**. The system closes the **Select Users / Groups** window.



You can delete users or groups from the **Target Users and groups** list. To delete any added users or groups, select the required check boxes and click the **Delete** button.

11. Click the **PUBLISH** button to start the publishing.

    The process of publishing the app and binaries begins. To view the various stages involved in publishing an app, click **+ Details**. To cancel the publishing, click **CANCEL**.

12. After the status changes to **Published** in the Publishing dialog, click **OK** to confirm the publishing.

For more details, refer to Uploading Client Binaries to Kony Fabric Console.

31.2.2 Publishing Services and Web Client Binaries to the Server

After you have configured all the required services for an app, you need to publish the app. Publishing allows your app to start using the Kony service in real-time. After an app is published, Kony Fabric generates the code that you can integrate with platform SDKs.

> ***Note:*** You can upload maximum up to 10 Web Client Binaries for an app.

> ***Important:*** When you publish an app to an environment, all the identity services associated with the app are published only to the selected run-time environment. The latest published Identity Services will affect any other apps in the same environment if they use these identity services.
>
> The latest published identity services on the current environment will not affect any apps in different environments.

Based on environments created, Kony Fabric Console allows you to publish apps to the environments.

Publishing services and Web client binaries of apps can be done in two ways:

- Asynchronous Publish. By default, the asynchronous publish is enabled.
- Synchronous Publish

Note: For On-premises only:

To skip a .war file if you have uploaded it for **Web** platform under the **Manage Client App Assets** tab, select the **Allow Manual Publish Only** check box in the Add a New Environment window.

Important: Make sure that your Kony Fabric Console version used for creating apps is same as Kony Fabric runtime components' (Integration, Sync, and Messaging) versions. All the components of Kony Fabric must be upgraded to same versions.

For example, if the Kony Fabric Console installed version is V8 for creating your apps, you must use the same Kony Fabric version for runtime components to publish your apps. If there is a version mismatch, Kony Fabric's Publish functionality and other runtime server components may not work as expected.

31.3 Publishing Native Client Binaries from Kony Management to Devices

After you publish native client binaries from Kony Fabric Console to a Kony Management environment, the app with these native client binaries needs to be updated in Kony Management Console.

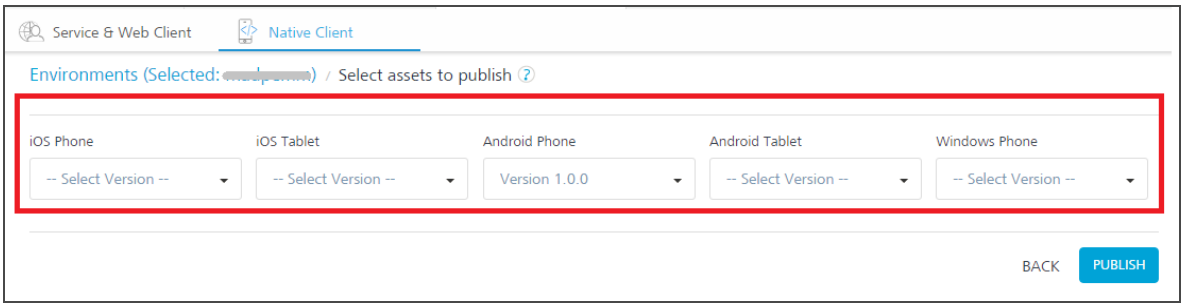
Go to Kony Management Console, enable the new update of application and target the app to Kony Management Users. In Kony Management Console, an admin targets the uploaded version of the app binaries to the enrolled users and then publishes the app binaries from Kony Management Console. For more details, refer to Kony Management Console User Guide.

For related topics, refer to Manage Client App Assets.

31.4 Upgrading Client Binaries

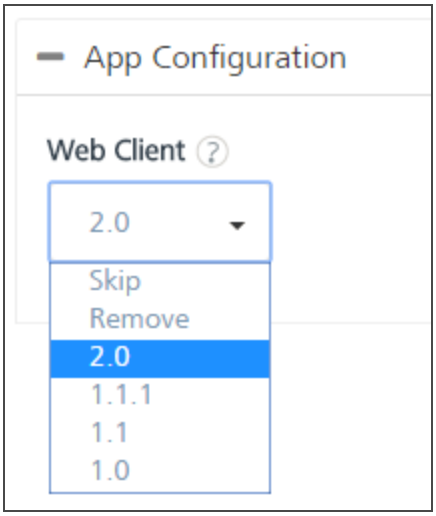
You can upgrade a binary file by importing another binary that has the same version, make changes to it, and then upload the new version to Kony Fabric Console. Kony Fabric allows you to upload multiple versions of a binaries.

- **For native client binaries**, if you select a binary version from the **Platforms** drop-down list and publish as preview or publish it to an EMM environment, the system publishes the selected version to EMM. EMM supports storing all the uploaded versions of native client binaries in the database.



  In EMM Console, an Admin can select a particular version and need to re-publishing the app for the targeted users.

- **For Web client binaries**, if you select a `.war` version from the **Web Client** platform drop-down list and publish it to the Server, the system overrides the existing version of the .War file with new version in the Server. Kony Fabric publishes only one version of a `.war` file to the server.



For more details, refer to Managing Client App Assets.

32. Kony Enterprise App Store (EAS) Service for Digital App Distribution

32.1 Overview

Kony Enterprise App Store (EAS) is a simplified app distribution service that enables an enterprise to securely and easily manage and distribute their apps to their users. All features of Kony EAS are available on Cloud and On-Premises.

With Kony AppPlatform Release V8 Service Pack 4, Kony App Server has been enhanced to provide the backend support for the Kony Enterprise App Store.

> ***Note:*** From Kony AppPlatform V8 SP4, Kony EAS Service is available to all Kony AppPlatform users by default.

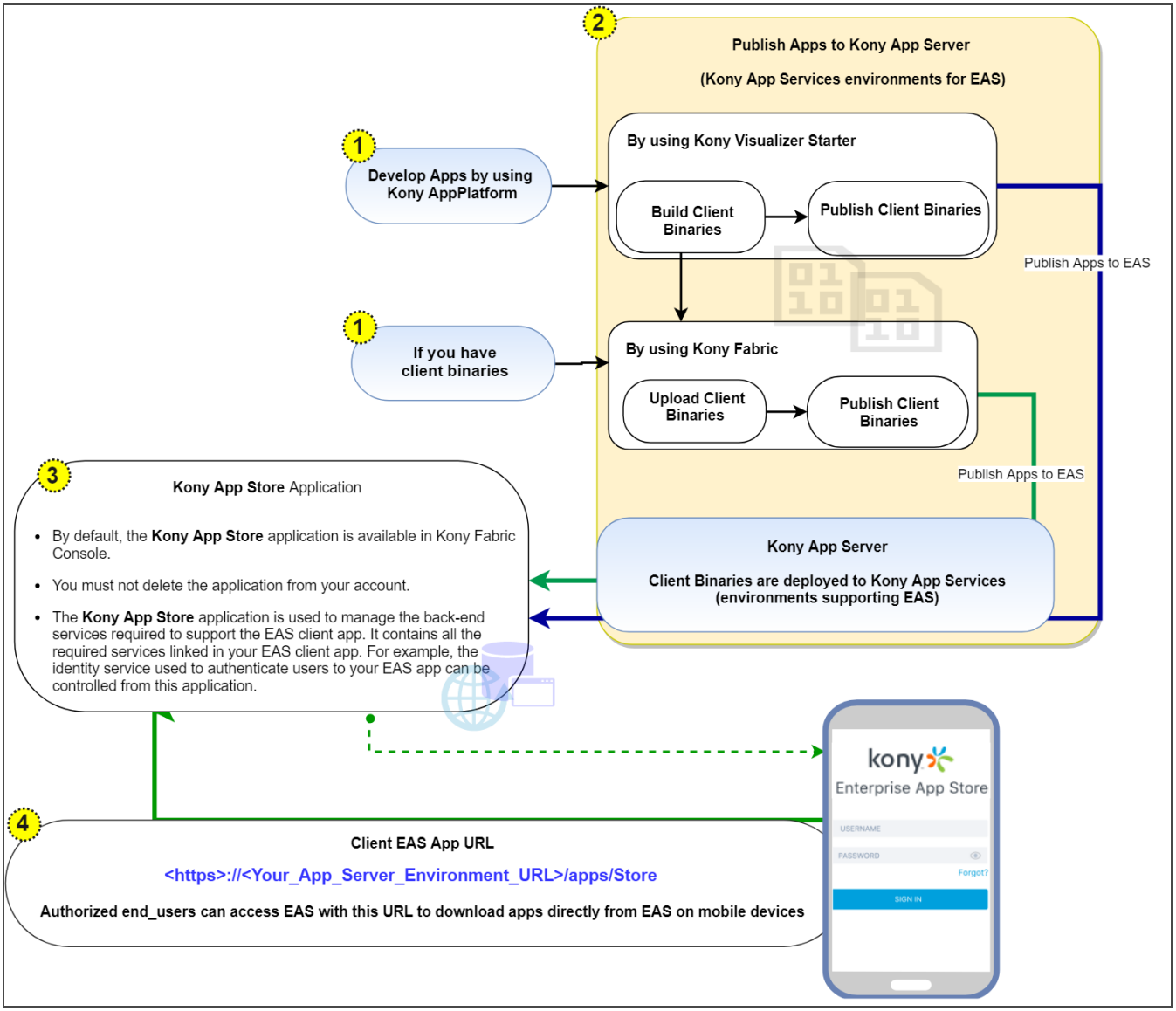
After an app is built by using Kony AppPlatform, Kony EAS service allows Kony AppPlatform users (**Admin/Member/Owner**) to choose to publish an app to the Kony App Server available in their Kony Fabric account. AppPlatform users can then allow authenticated access to certain end-users. The authorized end-users can access the published apps directly from Kony EAS, just like how you can access apps from the Google Play Store and the Apple App Store.

The following table details Kony Fabric roles to manage EAS Apps

EAS Apps Management			
PERMISSIONS	User Roles in Kony Fabric		
	OWNER	ADMIN	MEMBER
Developing Apps	✓	✓	✓
Publishing Apps to EAS (Kony App Server)	✓	✓	✓

EAS Apps Management			
PERMISSIONS	User Roles in Kony Fabric		
	OWNER	ADMIN	MEMBER
Unpublishing Apps to EAS	✓	✓	✓
Setting Public access mode for an EAS App: By default, the authentication is disabled for EAS. So, when an app is published to the EAS, users can view and download your published apps from the EAS client app without logging in to EAS.	✓	✓	✓
Setting Protected access mode for the EAS App: Users with Kony Fabric account can enable authentication to EAS so that, only authorized users can log in to the EAS and access the published apps	✓	✓	✓

32.2 Workflow of Apps for Kony App Server



32.3 Prerequisites for Cloud

- Access to a Kony Cloud account. If you do not have a cloud account, you can register for it at Kony Cloud Registration.

- Access to a Kony Cloud Build Environment version, and access to Fabric App Server for publishing apps to EAS.

EAS and Platform Versions Compatibility Chart

The following table details the supported versions of EAS source and Platform.

EAS Source App version		Platform Supported Version		
Client App (Store.zip)	**Server App (Kony App Store.zip)**	Fabric	Visualizer	Middleware
1.0.0 Features • Initial Relese	1.0.0	NA	V8 SP4 FP44 or lower	8.4.3.x.
1.1.0 Features • Support for Web apps	1.0.0	NA	V8 SP4 FP44 or lower	8.4.3.x.
2.0.0 Features • Support for Android 10 • Support for iOS 13 • Push Notifications	2.0.0	NA	V8 SP4 FP48 or higher	8.4.3.x

EAS Source App version		Platform Supported Version		
Client App (Store.zip)	Server App (Kony App Store.zip)	Fabric	Visualizer	Middleware
2.1.0 Features • Bug fixes	2.0.1	NA	V8 SP4 FP66 or higher.	8.4.3.x
3.0.0 Features • Support for Desktop View • Help section in iOS Native Apps	2.0.1	NA	V8 SP4 FP66 or higher.	8.4.3.x
• **SP**: Service Pack • **FP**: Fix Pack				

32.4 Prerequisites for On-premises

- For publishing apps to the Enterprise App Store, you have to have Kony Fabric App Server.
  Click here for more details on **EAS and Platform Versions Compatibility Chart**

  The following table details the supported versions of EAS source and Platform.

EAS Source App version		Platform Supported Version		
Client App (Store.zip)	Server App (Kony App Store.zip)	Fabric	Visualizer	Middleware
1.0.0 Features • Initial Relese	1.0.0	NA	V8 SP4 FP44 or lower	8.4.3.x.
1.1.0 Features • Support for Web apps	1.0.0	NA	V8 SP4 FP44 or lower	8.4.3.x.
2.0.0 Features • Support for Android 10 • Support for iOS 13 • Push Notifications	2.0.0	NA	V8 SP4 FP48 or higher	8.4.3.x
2.1.0 Features • Bug fixes	2.0.1	NA	V8 SP4 FP66 or higher.	8.4.3.x

EAS Source App version		Platform Supported Version		
Client App (Store.zip)	Server App (Kony App Store.zip)	Fabric	Visualizer	Middleware
3.0.0 Features • Support for Desktop View • Help section in iOS Native Apps • **SP**: Service Pack • **FP**: Fix Pack	2.0.1	NA	V8 SP4 FP66 or higher.	8.4.3.x

- Set the variable `max_allowed_packet` as per the size of your application:

> *Important:* **For on-premises**: If the size of your application is more than the variable size of server database, an error occurs. Ensure the variable `max_allowed_packet` is set to a higher value than the application size.
>
> To avoid this error, increase the global variable in the server database.
>
> - Modify the SQL Statement to set global `max_allowed_packet=10*1024*1024`.
> In this statement, the server database size is configured to 10-Megabytes(MB).

> For example, if your application size is 1024-kilobytes (1 MB) and you try to publish an application from Kony Studio of 2048-Kilobytes(2 MB) an error appears while publishing. Increase the size of server database to a value more than 2048-Kilobytes (2 MB), for publishing the application. Refer to Increase innodb_log_file_size in my.ini file- MySQL.

- Before you publish apps to EAS you must set the `File Storage Root Directory` path in the **Settings > Runtime Configuration > File Storage Configuration** in App Services Console.
    - Kony Fabric uses client app binaries uploaded to the location set in `File Storage Root Directory` and publishes them to EAS. So, the `File Storage Root Directory` path should be in the same system as the Kony Fabric installation.

      > ***Important:*** If your Kony Fabric is on Windows and the path is in the **D** drive, then provide the value as **D**:
      >
      > For example,
      > Kony Fabric is installed on `D:\Installers\KonyFabric`
      > The file path can be `D:\Installers\Content\KonyEAS`

      Configuring File Storage Root Directory (Target location) path:

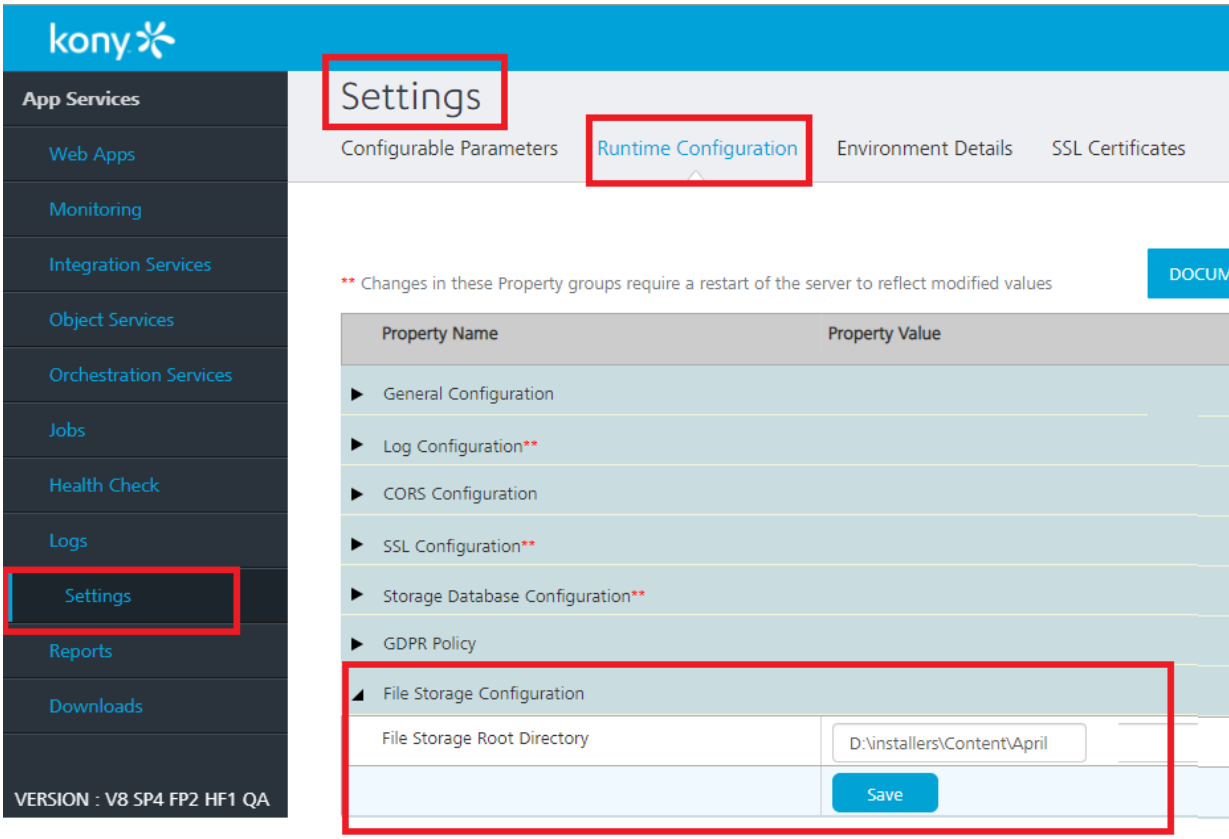
      a. Sign in to your on-premises instance of Kony Fabric.

      b. From the left navigation pane, select **Environments**.

      c. Click the server icon to open the **Admin Console**.

      d. From the left navigation pane of the **Admin Console**, select **Settings**.

      e. Click on the **Runtime Configuration** tab and expand the **File Service Configuration** section.



f. Provide the path of your file system target location under **File Service Root Directory**.

32.5 Supported Channels and Platforms for Kony App Server

- The following table details the supported channels and apps in EAS:

Android	iOS	Desktop
• Mobile (Native and Web Apps) • Tablet (Native and Web Apps)	• Mobile (Native and Web Apps) • Tablet (Native and Web Apps)	• Web Apps

- EAS support is available from Kony AppPlatform V8 SP4 onwards.

32.6 Publishing Apps to EAS (Kony App Server)

32.6.1 Publishing Client Binaries to Kony App Server from Kony Visualizer (for Cloud only)

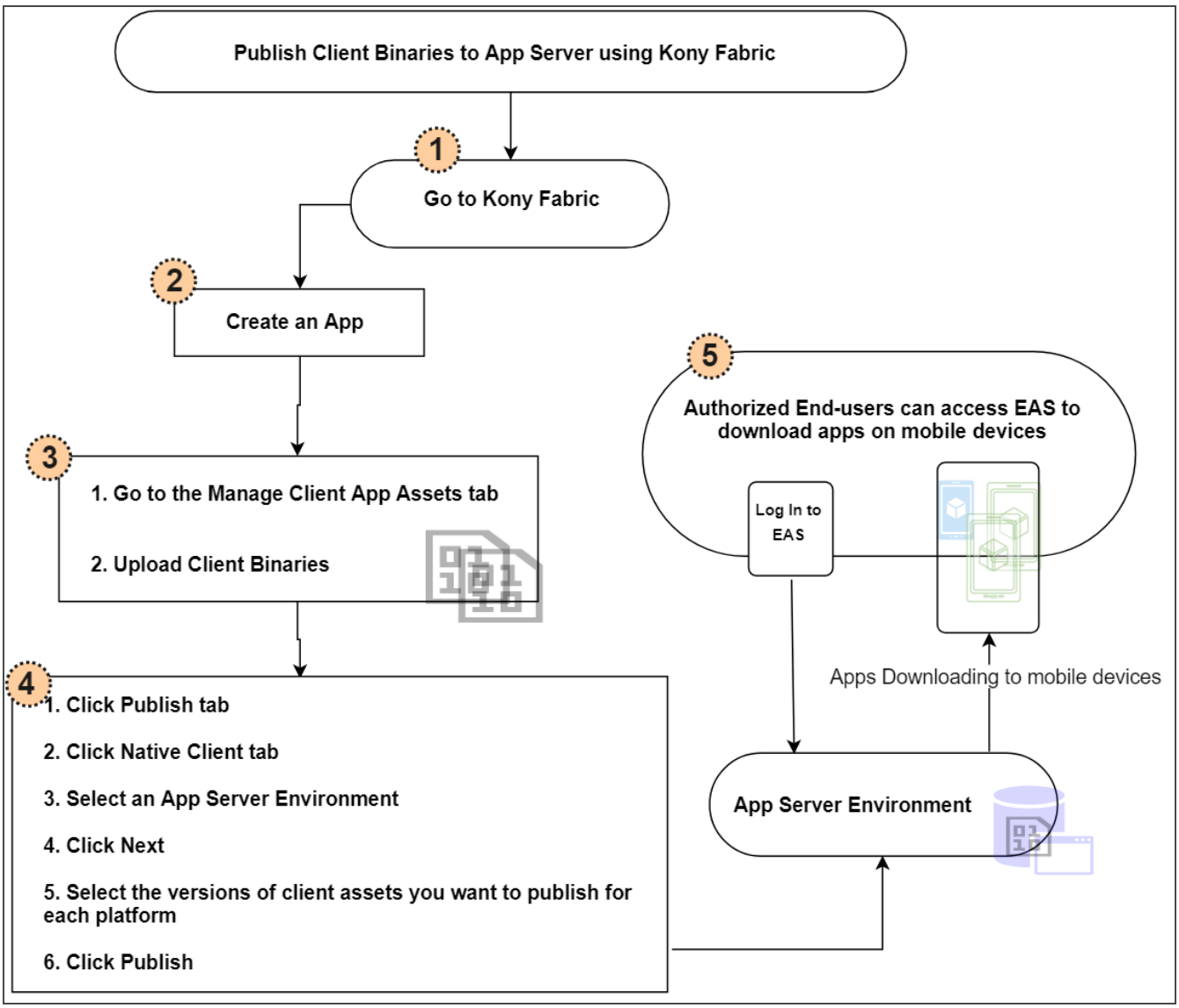
- With Visualizer Starter, you can build client binaries and publish them directly to Kony App Server from Visualizer. For more information, refer to Publishing Native Apps to Enterprise App Store from Visualizer Starter.

> ***Note:*** If you are a Kony Visualizer Enterprise user, you must publish apps to EAS from Kony Fabric Console. The behavior of EAS is the same for Kony Visualizer Enterprise and Visualizer Starter Editions.

32.6.2 Publishing Client Binaries to Kony App Server from Kony Fabric

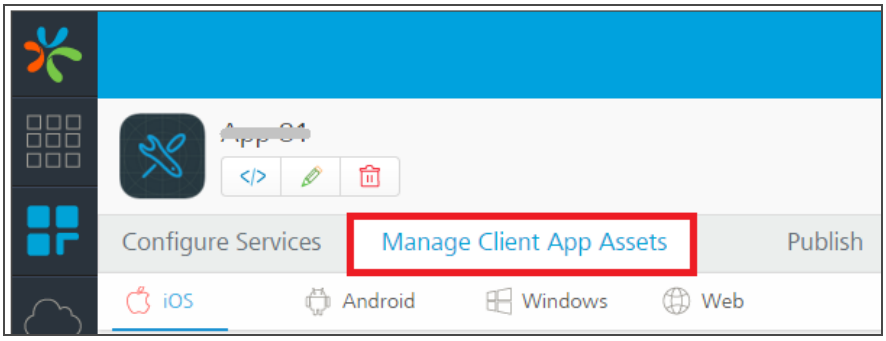
By using Kony Fabric, you can directly upload client binaries and publish them to Kony App Server. After an app is published to Kony App Server, an authorized end-user can access EAS Client app to view available applications and download them using a mobile device.

The following flow diagram explains the process of publishing client binaries to Kony App Server by using Kony Fabric

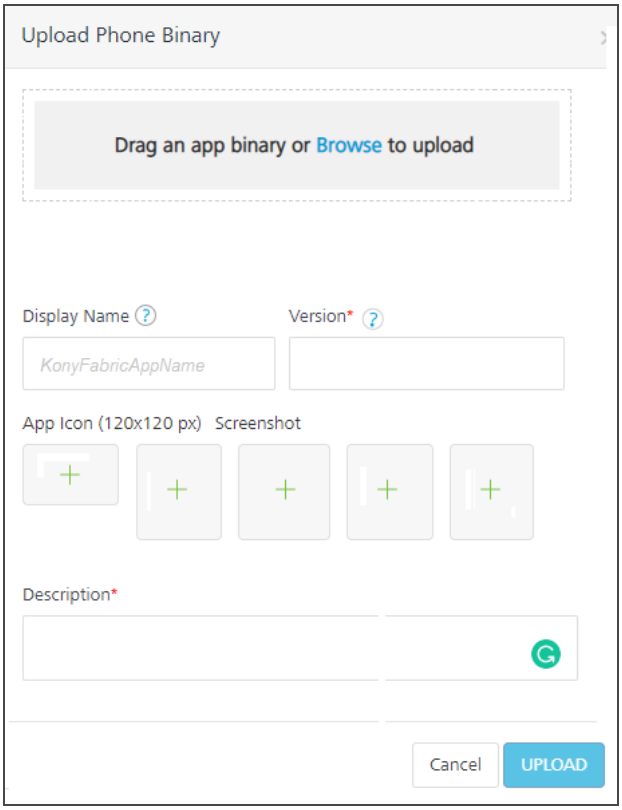


To publish client binaries to Kony App Server by using Kony Fabric, follow these steps:

1. Create an app in Kony Fabric.
2. In the App details page, click the Manage Client App Assets tab for the app. By default, the iOS tab is selected.



3. **Upload the client binaries for supported platforms such as iOS, Android, and Web**, as follows:
    a. In the **Mange Client App Assets**, click the tab for iOS or Android or Web.
        i. Click the **UPLOAD** button for the specific channel. The **Upload** dialog appears for the selected channel.



(This sample screen shot is for native phone binary upload)

> *Important:* **For on-premises**: If the size of your application is more than the variable size of server database, an error occurs. Ensure the variable max_allowed_packet_size is set to a higher value than the application size.
>
> To avoid this error, increase the global variable in the server database.
>
> - Modify the SQL Statement to set global max_allowed_packet=10*1024*1024.
> In this statement, the server database size is configured to 10-Megabytes (MB).
>
> For example, if your application size is 1024-kilobytes (1 MB) and you try to publish an application from Kony Studio of 2048-Kilobytes(2 MB) an error appears while publishing. Increase the size of the server database to a value more than 2048-Kilobytes (2 MB) for publishing the application. Refer to Increase innodb_log_file_size in my.ini file- MySQL.

ii. Drag and drop the app binary or click **Browse** and navigate to the required app binary file.

b. In the **Display Name** field, specify the name of the application. The app is displayed with the specified name in EAS when you view the App Store in a mobile device. This field is optional.

   If the display name is not specified for the app binary, the published app is displayed with the default Kony Fabric app name in EAS.

c. In the **Version** field, add the version for the app.

d. To add the app icon, click the **Plus** symbol in the **App Icon** section and navigate to the icon file. This is applicable to native binary.

e. To change the screens, click the **Plus** symbol in the **Screenshot** section and navigate to the required image file. This is applicable to native binary.

f. In the **Description** field, add the description of the app.

g. Click **UPLOAD**. The add details are updated to the selected channel.

> ***Important:*** In the case of Web binary, if you have uploaded a Web binary file, you must publish the Web binary to the server as follows:
>
> 1. Go to the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected.
> 2. Under the **Select environment to publish**, click an environment.
> 3. Click the **PUBLISH** button. The process of publishing the app begins.
>
> After the binary is published, the publish status changes to **Published** in the **Select environment to publishing** section.

4. **After you upload a app binary to the server, you must publish the binary to the EAS as follows:**

   a. Go to the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected.

   b. Click **Native Client**. The **Native Client** page lists Kony App Server environments suitable for EAS. The list also displays one of the following app statuses for that cloud or environment.

      - **Published**: An app is published to a cloud or an environment. You can unpublish the app, if required.
      - **Not Published**: An app is not published to a cloud or environment. You can publish the app, if required.
      - **Error**: An app is canceled while publishing or unpublishing. You can publish or unpublish the app, if required.

   c. Select the required **Kony App Services** environment.

d. Click **NEXT**.

e. In the **Select assets to Publish** page, select the version of each client binary you want to publish to the selected environment.

f. Click **PUBLISH**. The process of publishing the app begins.

> ***Important:*** If you are publishing the Kony App Store version 2.0.1 or higher to an environment where the lower version of the Kony App Store is already published, then in order to get the latest changes, you must restart the application server.
>
> For more information on How to Stop and Start a Deployment Application Server, refer to How to Stop and Start Kony Fabric.

When the app is published, the **Application published** window appears and displays link to the **App Store** and **Manage App Store Users**, along with the **QR Code** to access the EAS.

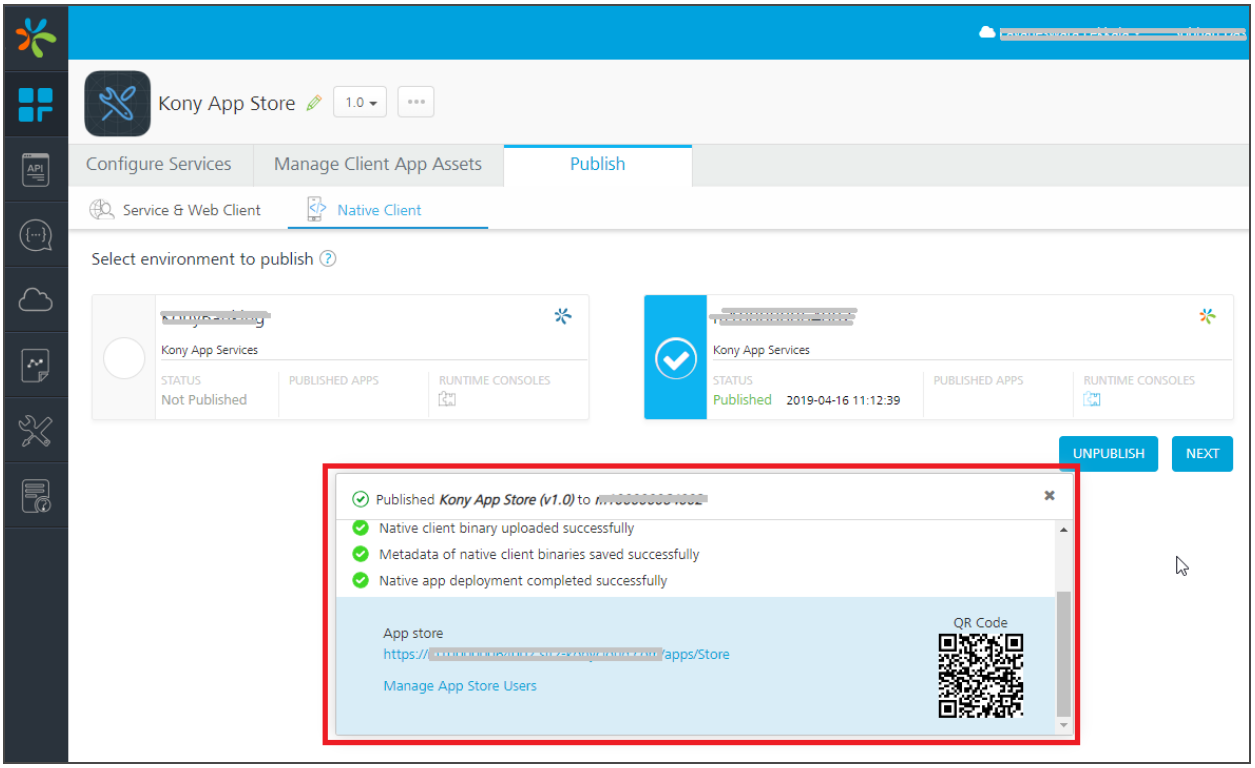
App Store URL
• If you open the **App Store URL** by using a **mobile device**, the **Responsive Web view** of the EAS Store displays all the published native and Web apps. • If you open the **App Store URL** by using a **PC system** (for example, **Desktop or Macbook**), the **Desktop view** of the EAS Store displays all the published Web apps suitable for Desktop only. For more details. refer to Downloading Apps from EAS
App Store QR Code
• If you open the **App Store QR Code** by using a **mobile device**, the **Responsive Web view** of the EAS Store displays all the published native and Web apps.

> ***Note:*** By default, the authentication is disabled for EAS. Users can view your published apps from EAS client app without having to log in. To enable authentication to your app, refer to Securing Apps in EAS.

32.7 Accessing EAS App Link

32.7.1 To access EAS link after the app is published (Kony Fabric)

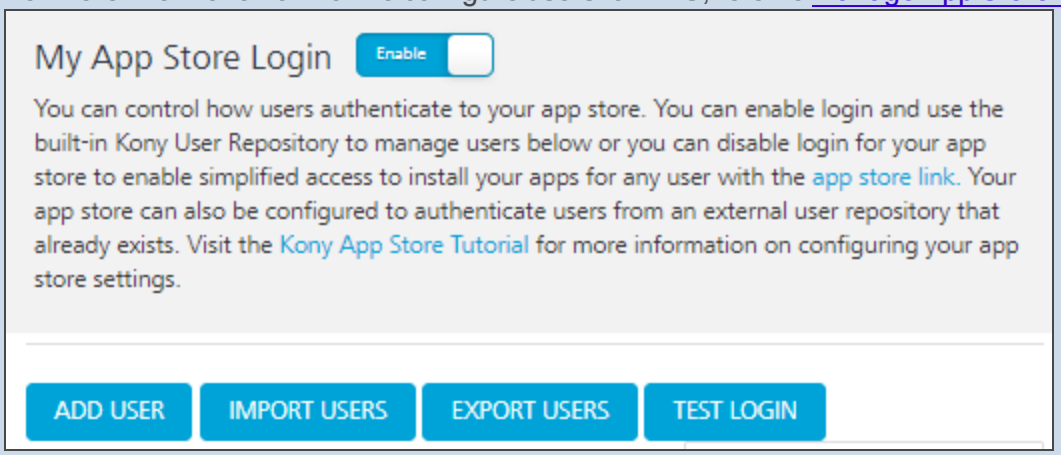
- When the app is published, the **Application published** window appears and displays link to the **App Store** and **Manage App Store Users**, along with the **QR Code** to access the EAS



> ***Important:*** Use the **App Store** link on a browser or scan the displayed **QR** code to launch the Enterprise App Store on your mobile.
>
> You can manage access to EAS by using the **Manage App Store Users** link, which allows

you to manage user access to published apps. When you click on the **Manage App Store Users** link, the **My App Store Login** window appears.

For more information on how to configure users for EAS, refer to Manage App Store Users.



32.7.2 To access EAS link from the published Environment (Kony Fabric)

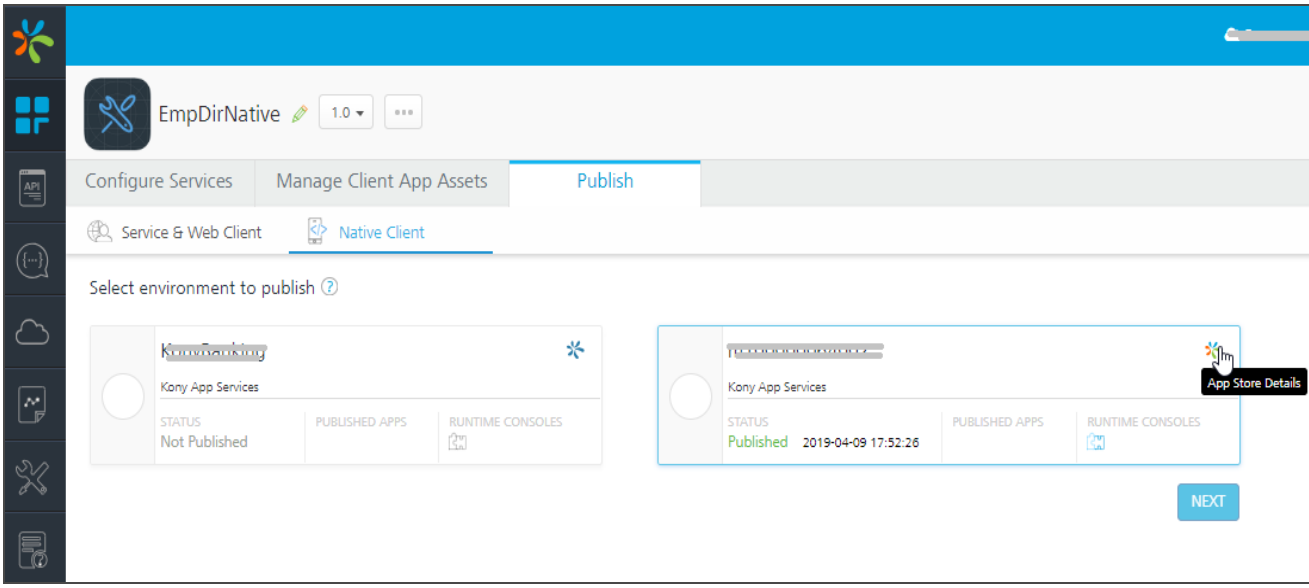
a. Publish an app to an EAS environment.

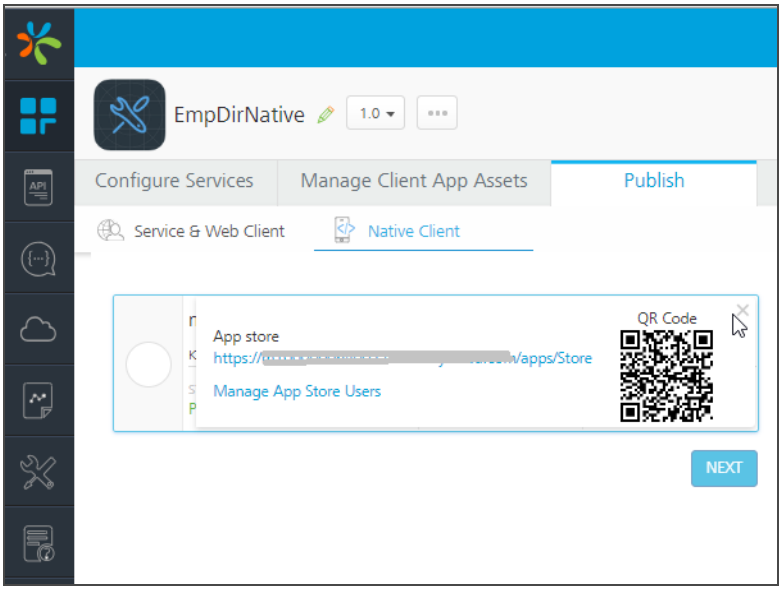
b. Click **Publish**.

c. Click **Native Client**.

   If the app status shows as **Published** in the environment, the **Kony logo** become active.

d. Click the **Kony Logo** to view links to the **App Store** and **Manage App Store Users**, along with the **QR Code** to access the EAS.





Important: Use the **App Store** link on a browser or scan the displayed **QR** code to launch the Enterprise App Store on your mobile.

You can manage access to EAS by using the **Manage App Store Users** link, which allows you to manage user access to published apps. For more information on how to secure, refer Securing Apps by using Kony Fabric.

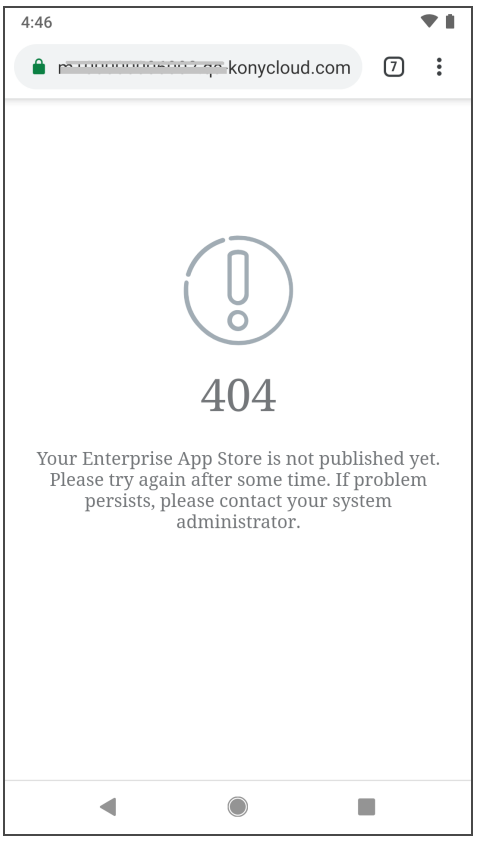
Note: By default, the authentication is disabled for EAS. Users can view your published apps from EAS client app without having to log in. To enable authentication to your app, refer to Securing Apps in EAS.

32.7.3 To access EAS link from the App Service Document and App Key (Kony Fabric)

- After you publish an app to EAS, navigate to Kony App Store. Click **Publish**. If the app status shows as **Published** in an environment, click the **App Key** to view link to the **App Store**.
- The App Service Document is available in Publish -> Environment -> Download -> App Service Document. The EAS client app is available at `<https://<your_App_Server_Environment_URL>/apps/store`. You can retrieve this URL from your App Service Document, as the URL under **Webapp**.

32.7.3.1 Handling EAS Unavailability

If the Kony App Store app is in an unpublished state, and the user tries to access the Enterprise App Store URL, the app displays the warning message of the problem.



32.8 Downloading Apps from EAS

The EAS client app is available at `<https://<your_App_Server_Environment_URL>/apps/store`.

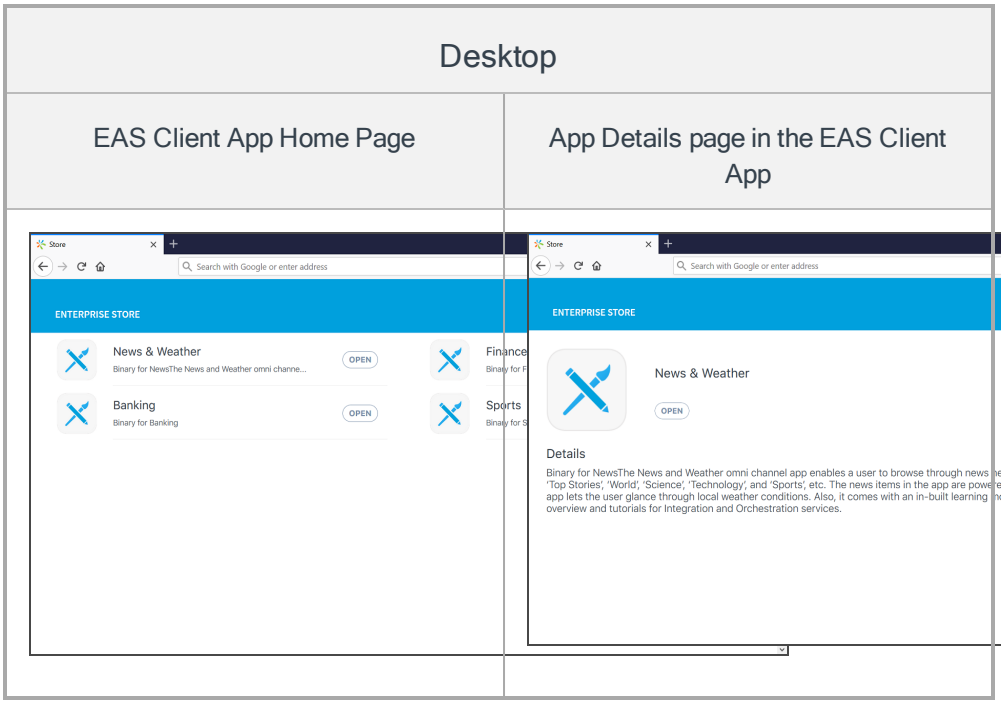
> ***Note:*** You can enable the authentication for your apps if required. For details, refer to Securing Apps in EAS.

1. To launch EAS on your mobile device, go to the URL `<https://<your_App_Server_Environment_URL>/apps/store` from your mobile device.

   The EAS client app is launched in the mobile device displaying all the published apps.

   - If you have published the app using Visualizer, your published app will be listed with the Visualizer app name.

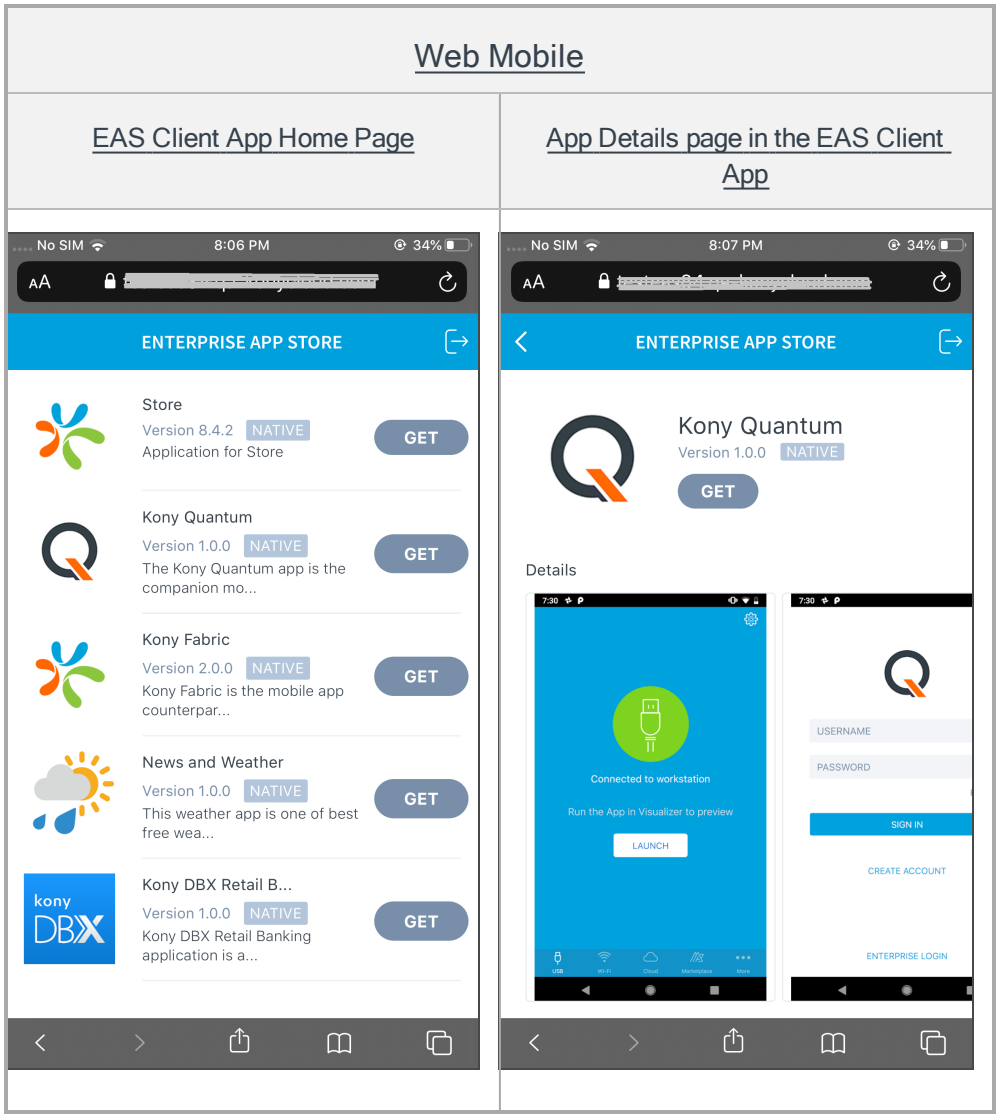
- If you have published the app using Kony Fabric, your published app will be listed with the same name as your linked Kony Fabric app. You can use the **Display Name** field to change the name of your application. As a result, the modified name of the app will be displayed in EAS.
    - Sample Screen-shots for EAS in Desktop View



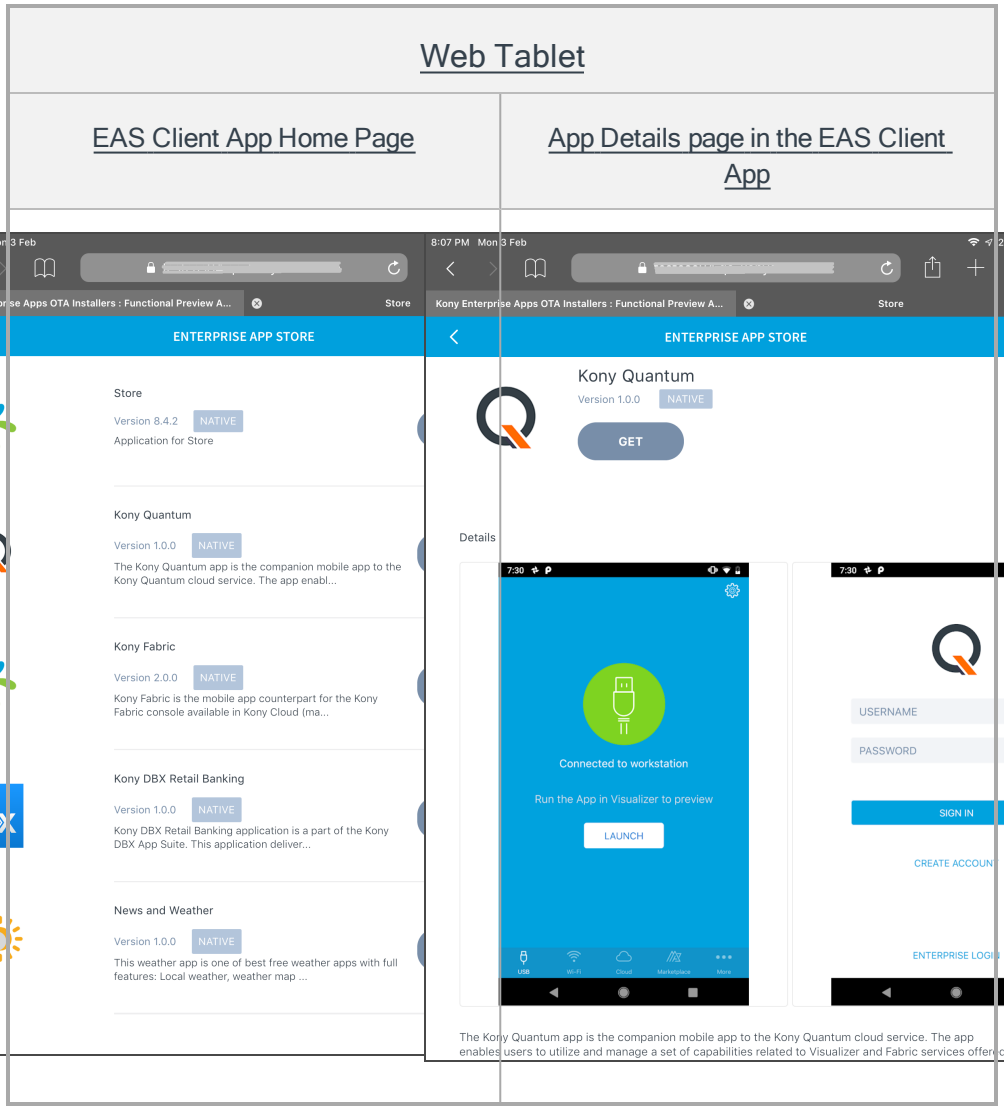
- Sample Screen-shots for EAS in Native Mobile View

- Sample Screen-shots for EAS in Native Tablet View

- Sample Screen-shots for EAS in Web Mobile View



- Sample Screen-shots for EAS in Web Tablet View



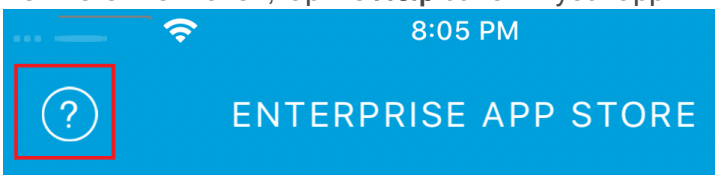
2. To download an app, click the **GET** button of the app. The application will start downloading to your mobile device.

32.9 Trust iOS Certificates

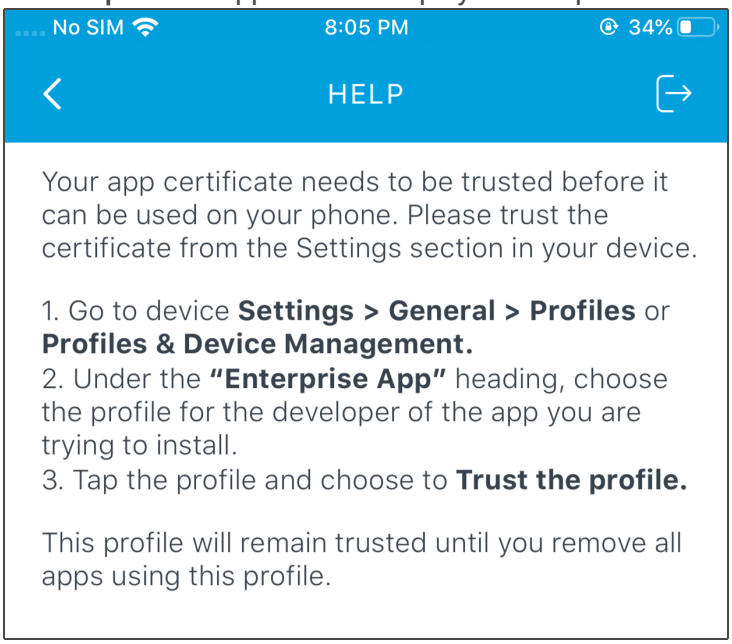
While viewing the iOS apps installed from EAS, if your iOS device displays the **Untrusted Enterprise Developer** pop-up message, you cannot view these apps. You must trust iOS certificates on your devices before viewing the apps.

1. Go to your iOS device **Settings** > **General** > **Profiles** or **Profiles & Device Management**.
2. Under the **Enterprise App**, choose the profile.

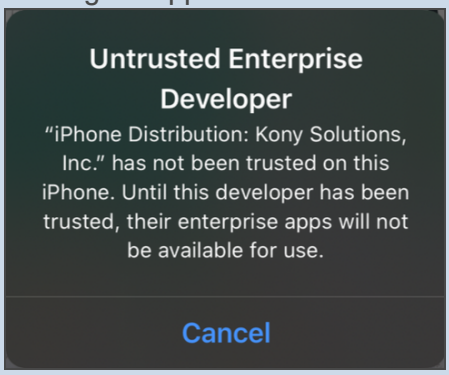
   For more information, tap the **Help** button in your app.



   The **Help** screen appears and displays the help details.



Note: The following pop-up message appears if your iOS certificates are not trusted while viewing the app installed from EAS.



32.10 Securing Apps in Kony App Server for EAS

The Kony Enterprise App Store (EAS) is a platform for sharing mobile apps. The default authentication type for EAS is Kony User Repository. The authentication type is disabled, by default. After an app is published to Kony App Server, users can view your published apps from EAS client app without logging in. You can browse and download apps from the store. An organization that uses Kony EAS can customize the user access based on their requirements.

Note: By default, from Kony AppPlatform V8 SP4, Kony EAS Service is available to all users with the **Store URL** that is displayed in the **Your App is ready** dialog window after your app is published.

To secure your apps, you can enable authentication. When you enable authentication to apps in EAS, by default the **Kony User Repository** is associated to the **Kony App Store** for authentication. All users registered in the Kony User Repository are authorized to access this app by specifying log-in credentials.

Note: When you log in to EAS by providing login credentials, the login details are stored in the browser settings of the mobile device. So, when you try to launch EAS on the same device next time, your authentication happens automatically, and you can start downloading apps.

Important: If you log out from EAS in a mobile device, and try to launch the EAS on the same device next time, the **Log In** screen appears, which helps you to enter the credentials to launch the EAS.

Note: By default, the authentication is disabled for EAS. Users can view your published apps from EAS client app without having to log in. To enable authentication to your app, refer to Securing Apps in EAS.

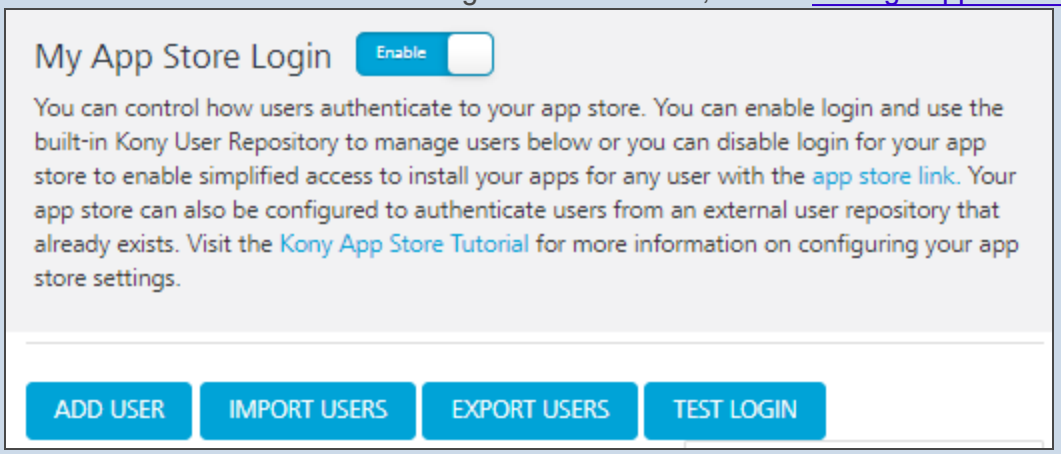
You can **enable** authentication to apps in EAS by using **Kony Fabric**, **Kony Visualizer Starter**, and **Kony App Services Console**.

32.10.0.1 Securing Apps by using Kony Fabric

When the app is published, the **Application published** window appears and displays link to the **App Store** and **Manage App Store Users**, along with the **QR Code** to access the EAS.

Important: Use the **App Store** link on a browser or scan the displayed **QR** code to launch the Enterprise App Store on your mobile.

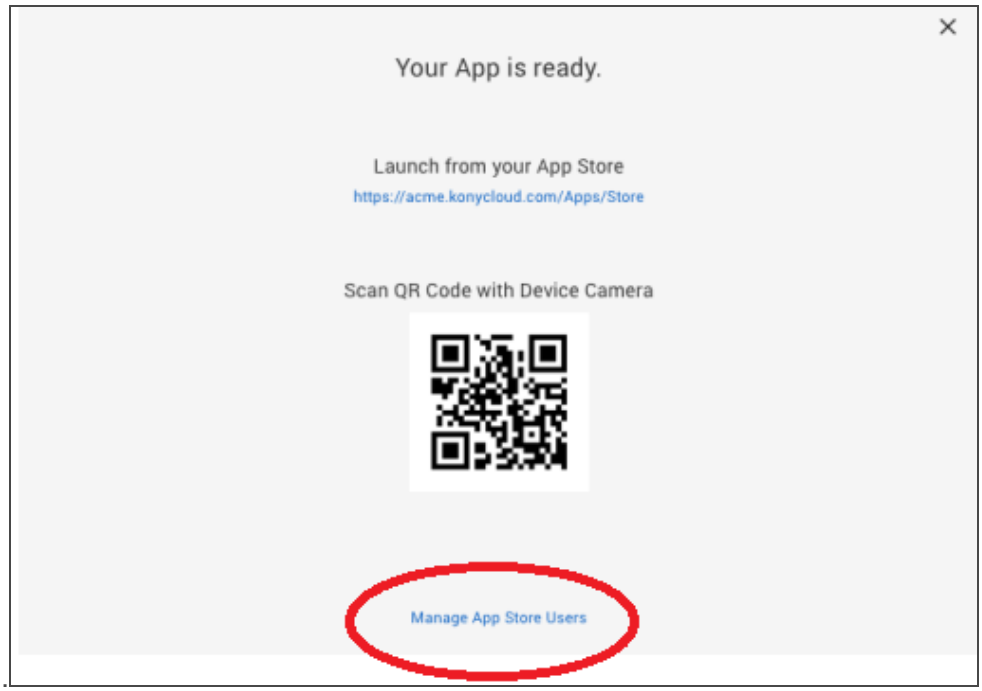
You can manage access to EAS by using the **Manage App Store Users** link, which allows you to manage user access to published apps. When you click on the **Manage App Store Users** link, the **My App Store Login** window appears.
For more information on how to configure users for EAS, refer to Manage App Store Users.



32.10.0.2 Securing Apps by using Kony Visualizer Starter (for Cloud only)

In the case of Kony Visualizer Starter, after you publish an app to EAS, you can view the **Manage App Store Users** link to enable authentication for the apps in the **Your App is ready** dialog.

For more details on how to manage users, refer to the **Manage App Store Users** section in the Accessing Enterprise App Store on Visualizer Starter.



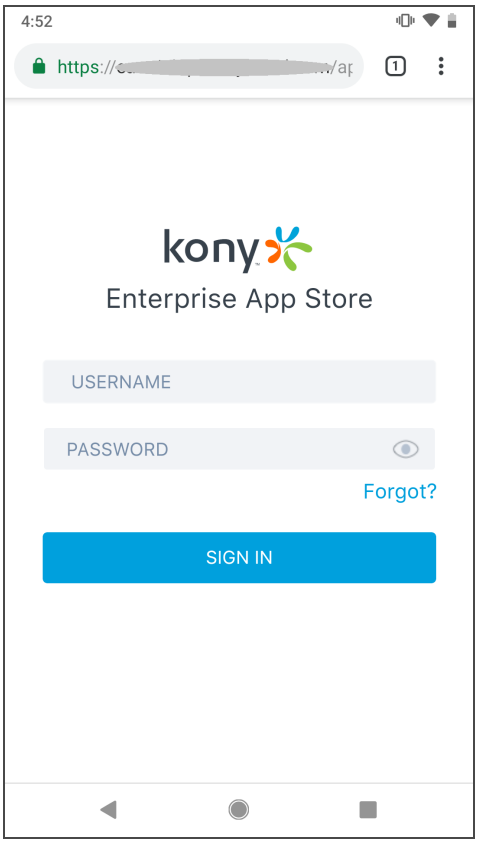
32.10.0.3 Securing Apps by using Kony App Services Console

The following procedure explains how to enable security to an app in EAS.

1. Log in to App Services Console, where you have published the app for EAS.
2. Navigate to **Settings**.
3. Click the **Configuration Parameters** tab.

4. Click the **Client App Properties** tab and specify the following details:
    - Field Name: `KONY_APPSTORE_LOGIN`.
    - Field Value: `true`. The Field Value is a case-sensitive in small letters.
5. Click **Save** to save the settings. Now your app has enabled for authentication.
6. Navigate to the **Kony App Store** app in your Kony Fabric account.
7. Click the **Identity** tab. The associated Kony User Repository is listed in the **Identity** page. Add users and groups if required. Your app is enabled with authentication for these users.
8. Publish the **Kony App Store** app to the Kony App Services environment. Now, when a user launches EAS, the log in screen appears on a mobile device.
    - In the case of the app that you want to access is protected by an authorization service, the Login page for EAS appears. For example, the following screen is EAS login screen

based on OAuth.



9. Enter the user name and password.
10. Click **SIGN IN**.

32.11 Forgot Password for Client Apps for EAS (for Cloud only)

The default authentication type for EAS is Kony User Repository. After an app is published to Kony App Server, any user from Kony User Repository can access EAS without authentication by using the URL of the published app.

When you enable authentication to apps in EAS, the authorized users must specify login credentials to access app published to EAS.

In this case if you forget your login password for EAS, you can reset the password by clicking the **Forgot Password** button on the Login screen in your mobile device. An email will be sent to your registered mail account for the resting password.

> ***Note:*** The Forgot Password functionality is only available for Kony User Repository auth provider.

32.12 Switching Between Identity Services in Kony App Store App for EAS

After you enable authentication to apps for EAS, you can switch between identity services configured in your Kony Fabric account.

1. Go to Kony Fabric Account.
2. Navigate to the **Kony App Store**.
3. Click the **Identity** tab. The authentication service type is listed in the Identity page. The default identity service type is `User Repository`, and the service name is `AppStoreUserRepository`.
4. Configure a new identity service or use an existing identity service.

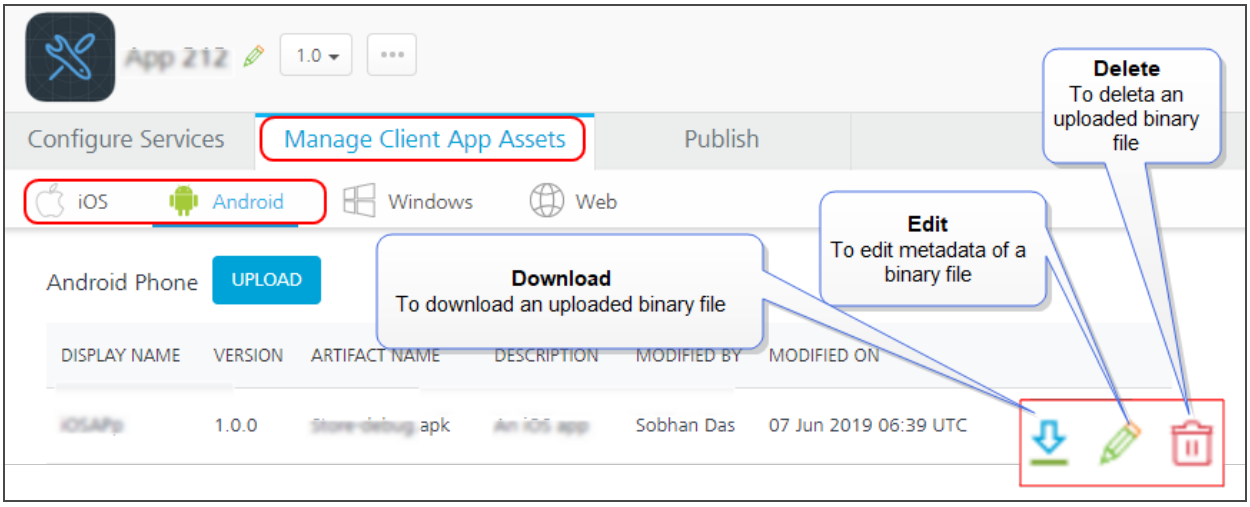
   The new service is listed in the **Identity** page. You need to have only one service in the list. In the case of more than one service listed, the first service from the list is used for authentication.
5. Publish the app to App Server.

   Now, the new service is associated to the app, and the existing identity service is unlinked from the app. When a user launches EAS, the user must specify the login credentials configured based on the new identity service that you have associated with the app.

32.13 Configuring Properties for Client Binaries

After you upload client binaries in Kony Fabric, you can reconfigure a few of the basic properties such as the description of a binary, the icon of a binary, and the related screen shots of a binary file. You can reconfigure these properties of the client binaries only by using Kony Fabric.

1. Open the app where assets are available.
2. Click the Manage Client App Assets tab.
3. Click the required platform tab, for example iOS or Android.

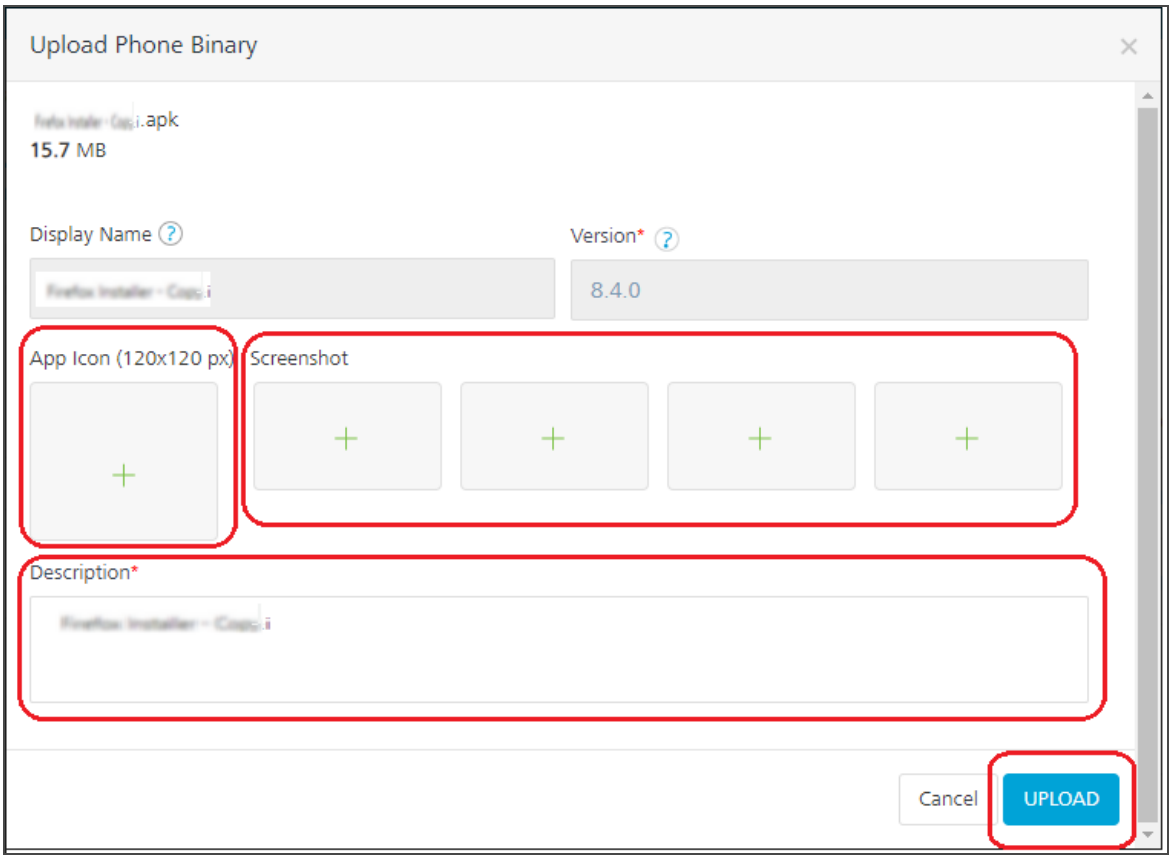


4. In the **Description** field, modify the text if required.
5. Click the **Edit**. The **Upload** dialog appears for the selected channel.

   You can perform various actions for a client binary such as the download, edit, and delete.

   - To change the app display name, click the **Display Name** text field and modify the name.
   - To change the app icon, click the **Plus** symbol in the **App Icon** section and navigate for the icon file.
   - To change the screens, click the **Plus** symbol in the **Screenshot** section and navigate for the required image file.

- To change the description, modify the text in the **Description** field.



6. Click **UPLOAD**. The modified details are updated to the client binaries.
7. Publish the client binaries to EAS.

32.14 Download Kony App Store App to Kony Fabric Account (for Cloud only)

The **Kony App Store** app is available in an account in Kony Fabric by default, and the app should exist in your account always. The **Kony App Store** app is configured with required services to be used in an app for EAS, for example, Identity Services.

In case you had deleted the **Kony App Store** app from your Kony Fabric Account, you can re-import it by raising a request with the Kony Support Team. To register and raise a customer ticket, refer to https://basecamp.kony.com.

> ***Note:*** For more information on how to customize EAS App for branding, refer to Walk-through of Kony Enterprise App Store (EAS) Source Application.

32.15 Unpublish an App from the EAS using Kony Fabric

Once your app is published to the Enterprise App Store (EAS) , if you want to unpublish it for any reason, you can do so in Kony Fabric.

To unpublish an app from the EAS using Kony Fabric, follow these steps:

1. Go to your app in Kony Fabric Console.
2. Go to the **Publish** tab. The **Publish** tab has the **Service & Web Client** and **Native Client** tabs. By default, the **Service & Web Client Publish** tab is selected.
3. Click the **Native Client** Tab.
4. Select the environment.
5. click **UNPUBLISH.**

> ***Note:*** You can also unpublish the published app through the Kony Quantum Visualizer. For more information, refer Unpublish an App from the EAS using Kony Quantum Visualizer.

32.16 Limitations for EAS

32.16.0.1 Android Limitations for EAS

When you try to download multiple apps simultaneously, the alert message dialog appears `This site is attempting to download multiple files. Do you want to allow this?`. Perform one of the following actions:

- If you click **Allow**, the download multiple apps task is resumed and start downloading the apps.
- If you click **Block**, all apps currently in the process of downloading, will stop.
  To continue downloading apps from EAS, you must un-block pop-ups from site settings in your browser in your Android mobile browser.

32.16.0.2 iOS Limitations for EAS

When you launch EAS from the Safari browser, and click the **GET** button of the app, an alert message dialog appears asking for `Open this page in iTunes`.

- If you want to continue to download the app from EAS, click **OK**. The app files will start downloading to your mobile device.
- If you click **Cancel** and try to download the same app or another app, an error message appears `Safari cannot open the page because the address is invalid`. All the apps download in-progress are stopped.
  - If you want to continue to download apps from EAS, you must refresh the browser page in the iOS mobile device.

32.16.0.3 Supported Databases (on-premises only)

- MySQL
- MSSQL
- Oracle

- MariaDB

> ***Note:*** IBM DB2 is not supported.

33. Sending Push Notifications to Enterprise App Store

33.1 Overview

Enterprise App Store (EAS), the app distribution service that enables an enterprise to manage and distribute customized apps to their users supports sending Push Notifications. EAS has been enhanced to leverage the Kony Engagement Services for the client app. Using which, you can send manual push messages to all registered devices, or to a subset of devices, based on the device ID. This functionality is available for all devices as well as the web channel. The Engagement Services provide a subscription-based notification mechanism that enables users to subscribe their devices to receive notifications.

When an EAS app is launched on a device, the device is registered to the Engagement Server that is linked to your environment. In case the EAS login is disabled, the device is registered only by the device ID. If the EAS login is enabled, the username of the person is also registered. After the device registration is completed, you can send push notifications to your entire user base or to select users.

33.2 Supported Channels and Platforms

Sending Push Notification to EAS is supported for the following channels and platforms:

- **iOS**- iPhone and iPad
- **Android**- Phones and Tablets
- **Web** - Android Web

> ***Note:*** Support for push notifications is available in Kony Enterprise App Store version 2.0.0, or later. For more information on this refer to Kony App Store on Marketplace.

33.3 Configuring a Push Notification

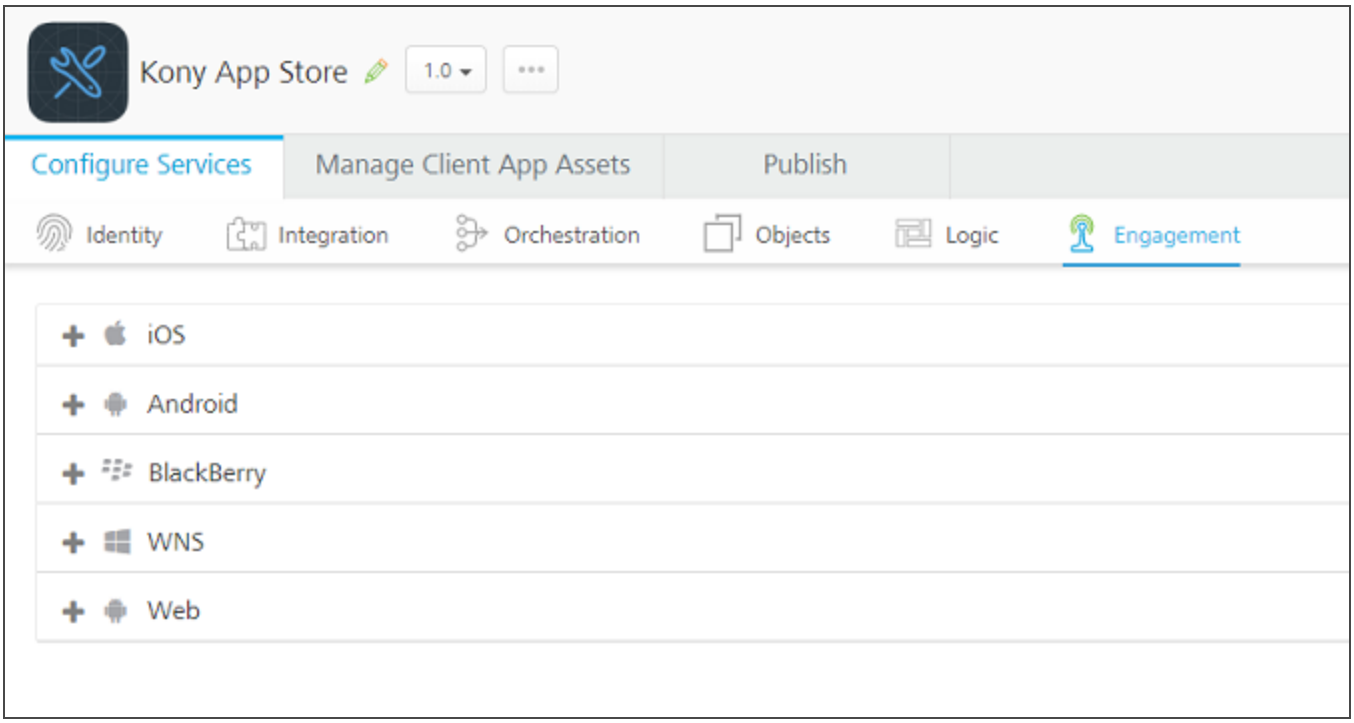
1. Go to the Kony Fabric Console. The **Apps** page appears.
2. Click on the Kony App Store. The **Configure Services** page appears.

3. Click on **Engagement**.

   Based on your requirement, choose Apple, Android, or Web platforms and enter platform specific details; for example, the FCM Authorization Key for Android. You can skip other platforms by clicking Next. You need to configure at least one platform to save and publish the application successfully.

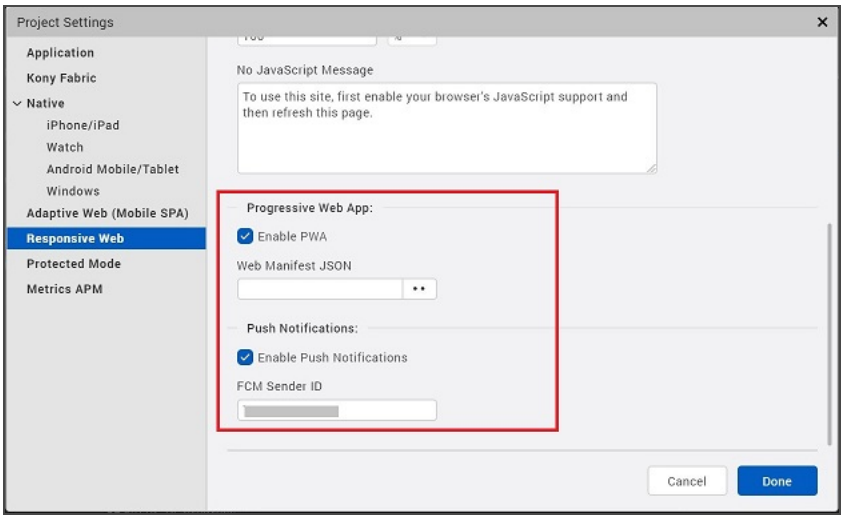
   > ***Note:*** For more information on generating a Server Key and Sender ID for Web devices, refer to Generating FCM Server Key and Sender ID for Web Devices.

   > ***Note:*** To create an app for the Web channel in the Engagement Server, refer to Add New App for Web.



33.3.1 Configuration on Client Side

- For enabling push notifications on the Web channel, do the following:
  1. Go to **Visualizer** > **Project Settings** > **Responsive Web**.
  2. Select the **Enable PWA** and **Enable Push Notifications** check boxes.
  3. Enter the **FCM Sender ID**.
  4. Click Done.



> ***Important:*** If you have enabled Push Notifications for EAS Web apps, then registration callbacks will be triggered for the first launch of the EAS Web app. If you change any settings at the Engagement Server side, then for the same changes to reflect in the EAS Web app, the Settings of the Web app must be cleared.

- For enabling push notifications on the Android channel, do the following:

  Go to your **Visualizer Project Workspace** > **projectProperties.json** add following key value pair:

  - `"enablefcmpushnotifications": "true"`

33.3.2 Configuration on Admin Console

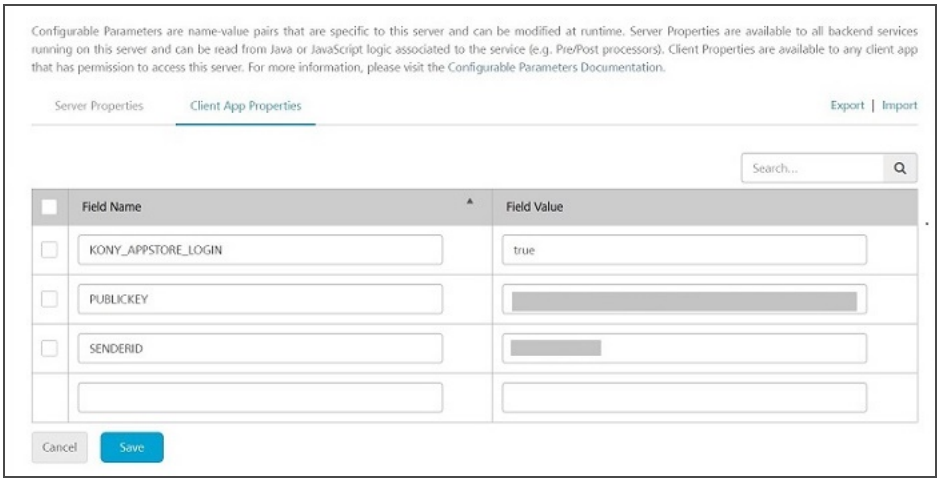
Using Admin Console you can specify the name-value pairs of your public key and sender ID and they can be modified at runtime. Client App Properties are available to any client app that has permission to access your Admin Console Server.

- Go to **Admin Console** > **Settings** and do the following:

  In **Client App Properties** of **App Services** add the following key-value pairs:

  - `PUBLICKEY : xxxxxxxxxxxxxxxxxxxxxxxxxx`
  - `SENDERID : xxxxxxxxxxx`

> ***Note:*** This is a provision made for you to change the public key and sender ID details everytime you change your FCM project.

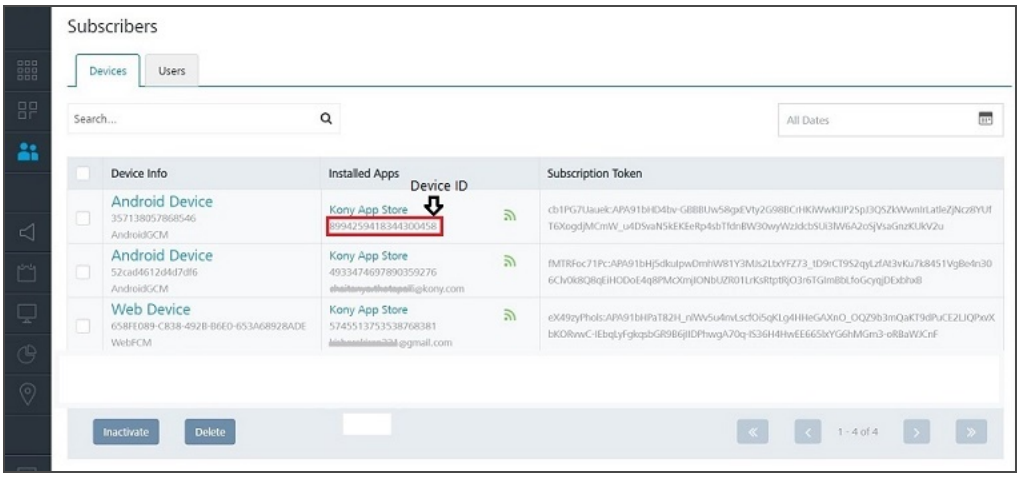


33.4 Device Registration on the Engagement Server

When a user opens an EAS app, the following scenarios can occur:

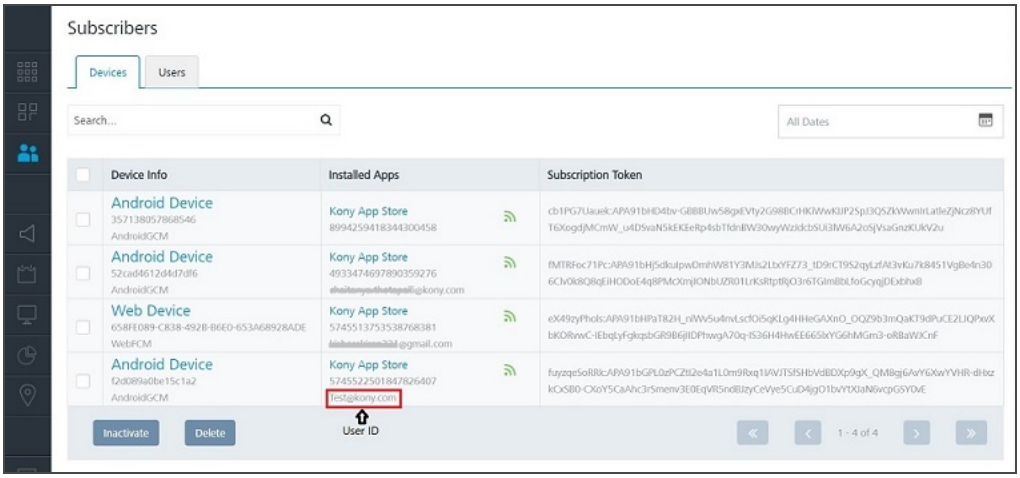
- **EAS without authentication**: The app opens and the device ID of the device on which the app is opened is registered on the Engagement Server.

- In the Kony Engagement Console, under Overview, click Subscribers. The Devices tab appears, with a list of all the devices registered to the server.
- You can send a **generic message** to these users, for example, "The Enterprise App Store will be unavailable on Sunday, 1st Dec 2019, to perform maintenance operations on the system.".



- **EAS with authentication**: The sign in screen appears, you must sign in to the app with the required credentials. The device ID of the device on which the app is opened is registered on the Engagement Server along with your User ID.
  - In the Kony Engagement Console, under Overview, click Subscribers. The Devices tab appears, with a list of all the users and their User IDs registered to the server.
  - You can send **specific messages** to these users, for example, "Your account on the HR App will be made inactive in a week due to inactivity. To continue to use the HR App,

please login to your account.".



33.5 Sending Push Messages

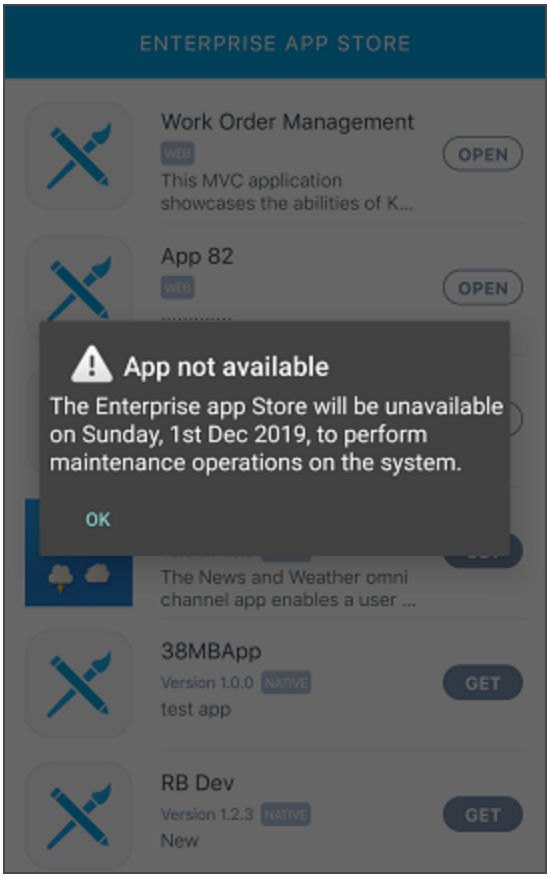
To configure a push message, follow these steps on the Kony Engagement Console:

1. From the Engagement section, click **Adhoc** from the left panel. The Adhoc screen appears, displaying the following three tabs: **Adhoc information**, **Select Users**, **Define Message**.
2. In the **Adhoc Information** tab, provide the type of message and application name to which the message should be sent. You can also schedule the date and time at which the message should be sent. For more information on selecting date and time, refer to Send Message.
3. In the **Select Users** tab, you can choose the entire subscriber list of the selected application or a Segment of users. For more information on how to create a Segment in Kony Engagement Server, refer to Adding a Segment.
4. In the **Define Message** tab, you can define the message that should be sent to your subscribers.

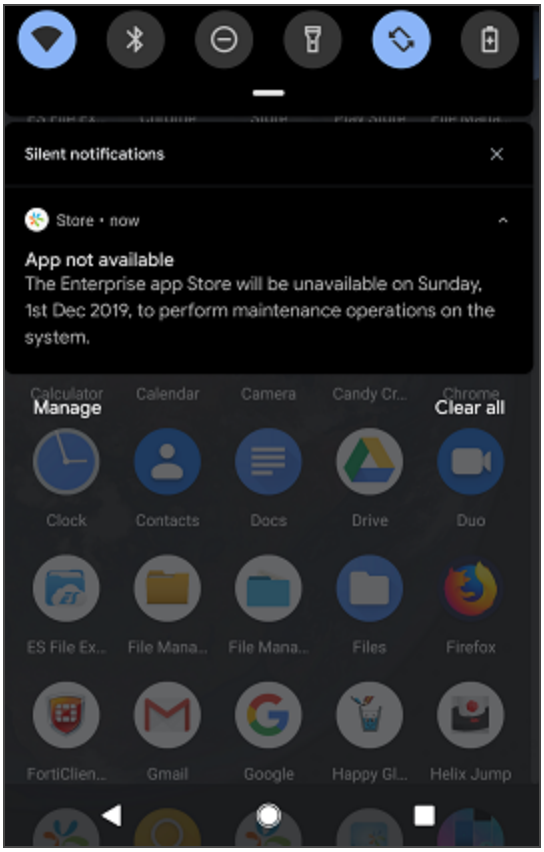
After this process is done, the message gets queued with the respective platform specific server.

For more information on how to trigger a notification through Engagement Server, refer to Adhoc Messages.

- When a push is triggered, if the EAS app is open on a device, a notification appears on the device.



- When a push is triggered, if the EAS app is closed on a device, a status notification appears on the device.



33.6 Known Issues

- In iOS 13 devices, the network error alert gets dismissed without user action.

34. Checking App Versions in Enterprise App Store

34.1 Overview

Enterprise App Store (EAS), the app distribution service that enables an enterprise to manage and distribute customized apps to their users supports sending Push Notifications. EAS has been enhanced to check versions of apps that are installed on devices and notify users about the latest versions of those apps along with the app upgrade sources.

You can now use the custom code in apps to enable for checking the latest versions of apps published to EAS. When an app is installed through EAS and launched, the app checks if the latest version of the app is published to EAS. This functionality is available for all devices as well as the Web channel.

For example: You built an app with the custom code and published the app to EAS. An app user downloads the app from EAS, and launches the app. After every launch of the app on a device, the custom code checks if the a newer version of the application is available on EAS, or at another source. If a newer version of the app is available, the custom code snippet displays the relevant information, and ask for the user's action - to Upgrade, or ignore the message.

34.2 Supported Channels and Platforms

Sending Push Notification to EAS is supported for the following channels and platforms:

- **iOS**- iPhone and iPad
- **Android**- Phones and Tablets
- **Web** - Android Web

34.3 Enabling Custom Code to Check Latest App versions in EAS

You can enable version check custom code in your apps for validating the installed apps on devices with the latest versions of apps published on Enterprise Stores. To do so, follow these steps:

In Kony Quantum Visualizer, sign in to your Kony Cloud account. To do so, from the top right corner of the Visualizer window, click Login. The Kony Account sign-in window opens. Enter your Kony Cloud email and password credentials, and then click Sign in.

1. . For example, in the Visualizer Project for your app.

2. Publish the app to EAS.

3. Users install the client app on devices.

4. Users launch the app.

The custom code checks if a newer version of the application is available on EAS and displays the relevant information to users.

Hi Team,

We have an issue while executing service task in workflow. If Service task fails with below mentioned error, following below steps need to be followed.

Please find service input params for that service task and exception pop up in admin console:

Service Input Params in workflow:

- **Issue**: While execution of a Service Task in Workflow Service, if the service task fails with the following error:

```
com.kony.component.service.common.ServiceComponentException:
Service call unsuccessful. Possible input parameter mismatch in
service configuration : Property access failed in MVEL script
```

```
causing exception [Error: could not access property (firstName)
in: java.lang.String].
```

Workaround:

1. Navigate to the installed application server folder.
2. Add the `mvel2.compiler.allow_override_all_prophandling=true` as `-D` property.
3. Restart the application server.

35. Walk-through of Kony Enterprise App Store (EAS) Source Application

35.1 EAS App Implementation

Kony Enterprise App Store (EAS) is a simplified app distribution service that enables an enterprise to securely and easily distribute and manage apps to the users. All features of Kony EAS are available on Cloud and On-Premises. For more information on EAS, refer to Kony Enterprise App Store (EAS) Service for Digital App Distribution.

From Kony AppPlatform V8 SP4 JuneFP onwards, Kony allows you to access the complete EAS App assets including the front-end project details and the back-end app. You can download the EAS App assets from the Kony Marketplace. These assets contain the source code of the Enterprise App Store, which is available for you to edit as per your requirement. You may choose to re-brand the app, or add specific forms to best suit your requirements.

For example, you can brand some of the following use cases:

- Name, Slogan, Style
- Graphics: Logo, app icon, splash screen, and visual image of the app
- Look of the app
- Brand keywords

35.2 App Architecture under Kony Reference Architecture

- **Kony Reference Architecture**: Kony EAS App is built on the Kony Reference Architecture. Kony Reference Architecture encapsulates your app's code into Kony's implementation of the Model View Controller (MVC) software architecture. Under Kony Reference Architecture, the code for your app has a clear division between the domain objects that model the app's functional domain and the presentation objects that represent what the user sees on the screen. Therefore, domain objects work independently of the presentation objects and can support multiple presentations, even simultaneously. Both domain objects and presentation objects are

encapsulated into specific modules that Kony Reference Architecture generates automatically for you from the user interface that you create in Kony Visualizer. For more information on Kony Reference Architecture, refer to A Deeper Look at Kony Reference Architecture.

- **Extensibility and Customization**: While Kony EAS provide optimal functionality out of the box, Kony understands that each enterprise has unique requirements. The solutions can be easily extended to support business-specific applications and requirements by leveraging the power of the Kony platform. This allows customers to add their own functionality via third-party solutions or their own built-in modules on the Kony AppPlatform. For more information on Extensibility and Customization, refer to Extensibility.

  Kony EAS is built by using the industry-leading Kony Visualizer and Kony Fabric tools, which allow enterprises to customize and extend the following:

  - Look and feel of the application
  - Business rules that define the application functionality
  - Rules around the data flow between the device and the back end

  Further, the solution includes an extension framework that allows implementation teams to do the following:

  - Introduce a new form in an existing business workflow
  - Replace an existing business logic method with new implementation
  - Programmatically add new widgets to existing forms
  - Update various properties of widgets
  - Hide a feature that the bank does not need

35.3 Prerequisites to Configure EAS Source App in Visualizer

Before you start using the source of the EAS App, ensure you have access to the following:

- Access to a Kony Cloud account. If you do not have a cloud account, you can register for it at Kony Cloud Registration.
- Access to a Kony Cloud Build Environment, Kony Visualizer Starter, and access to Fabric App Server for publishing apps to EAS.

  EAS and Platform Versions Compatibility Chart

  The following table details the supported versions of EAS source and Platform.

EAS Source App version		Platform Supported Version		
Client App (Store.zip)	**Server App (Kony App Store.zip)**	Fabric	Visualizer	Middleware
1.0.0 Features • Initial Relese	1.0.0	NA	V8 SP4 FP44 or lower	8.4.3.x.
1.1.0 Features • Support for Web apps	1.0.0	NA	V8 SP4 FP44 or lower	8.4.3.x.

EAS Source App version		Platform Supported Version		
Client App (Store.zip)	Server App (Kony App Store.zip)	Fabric	Visualizer	Middleware
2.0.0 Features • Support for Android 10 • Support for iOS 13 • Push Notifications	2.0.0	NA	V8 SP4 FP48 or higher	8.4.3.x
2.1.0 Features • Bug fixes	2.0.1	NA	V8 SP4 FP66 or higher.	8.4.3.x
3.0.0 Features • Support for Desktop View • Help section in iOS Native Apps	2.0.1	NA	V8 SP4 FP66 or higher.	8.4.3.x

- **SP**: Service Pack
- **FP**: Fix Pack

- Configure the various Project Settings.
    - Go to **Project** > **Settings** and configure the build settings for each Native platform. For more information on **Project Settings**, click here.
- Platform specific prerequisites:
    - If you choose to build an application for the **iOS** platform, you must provide the Mobile Provision, .P12, P12 password, and the Development method. To do so, go to **Project Settings > Native > iPhone/iPad**. For more details on the iOS configurations, click here.
    - If you choose to build an application for the **Android** platform in **Release mode**, then the Android signing details are mandatory. To do so, go to **Project Settings** > **Native** > **Android Mobile/Tablet**. For more details on Android sign in details, click here.
- If you choose to build an application in **Protected mode**, then setting the public and private keys is mandatory. To do so, go to **Project Settings > Protected Mode**. For more details on how to generate public and private keys, click here.

35.4 Downloading EAS App Assets

The EAS App assets are divided as follows:

- `store.zip` is the front-end source app of the EAS.
- `kony_app_store.zip`is the back-end source app of the EAS.

Store.zip	Kony App Store.zip
`store.zip` is the Visualizer project created for EAS based on Kony Reference Architecture. • The Store project contains the client app/front-end app details of the EAS App. It contains the details of the project layout, modules and forms, and the client-side code of the front-end app. The **Store** project is associated with the **Kony App Store** App for the linked services.	`kony_app_store.zip` is the Server app containing the services linked to the EAS App.

35.4.1 Downloading Front-end Project for EAS App for Kony Visualizer

Store is the front-end project of the EAS created based on the Kony Reference Architecture. You can download the EAS App assets from Kony Marketplace. The store.zip app contains details such as forms for supported channels, splash screen, client-side code, and modules and forms of EAS.

For Cloud, the Store app is bundled with the front-end and Server apps.

1. Go to Kony Marketplace.
2. Click **Download**. The `store.zip` file gets downloaded to your local system. For more details on EAS versions, refer to EAS and Platform Versions Compatibility Chart.

35.4.2 Downloading Server App (Kony App Store) for Kony Fabric

The **Kony App Store** is the Server app (Kony Fabric), which contains services linked to EAS. You can download the source app (Kony App Store) from Marketplace.

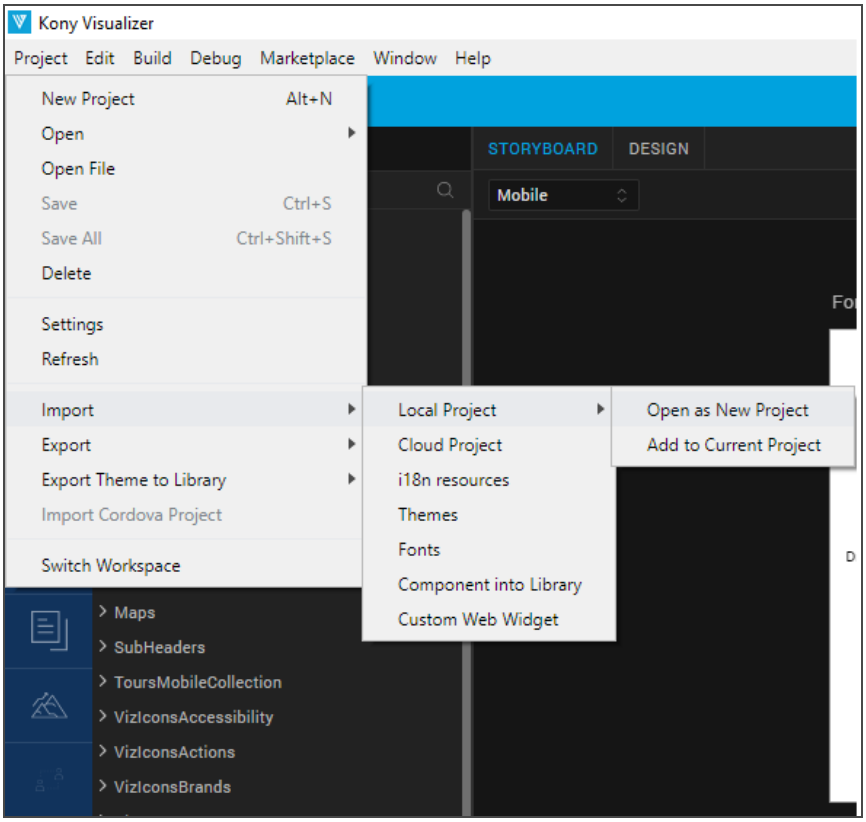
1. Go to Kony Marketplace.
2. Click **Download**. The `kony_app_store.zip` file gets downloaded to your local system. For more details on EAS versions, refer to EAS and Platform Versions Compatibility Chart.

35.5 Configuring EAS App in Visualizer

35.5.1 Importing EAS Front-End Project in Visualizer

You can view the layout of the front-end app and the source code by importing the Store project to Kony Visualizer.

1. Make sure you are logged into Kony Visualizer.
2. From the **Project** menu, click **Import > Local Project** > **Open as New Project**.



3. In the **Import** dialog box, select **Select archive files**, and then click **Browse** to select the **store.zip** file that you have downloaded.

   The Store project gets imported to your Visualizer. You can view the layout of the EAS project.

35.5.2 Importing Server App to Kony Fabric

The Kony App Store app is available in your account in Kony Fabric by default. The app should be existing always in your account for EAS to work. The Kony App Store app is configured with the required services to be used in an app for EAS, for example, Identity Services.

In case you want to import a different version of the app, you can re-import it from Kony Marketplace.

1. Log in to Kony Fabric.
2. In the **Apps** > **Fabric App**, click **IMPORT**, and drag and drop or browse for the `kony_app_store_v1.0.zip` file to upload.
3. In the **Open** dialog, select the **Kony App Store** app that you downloaded.

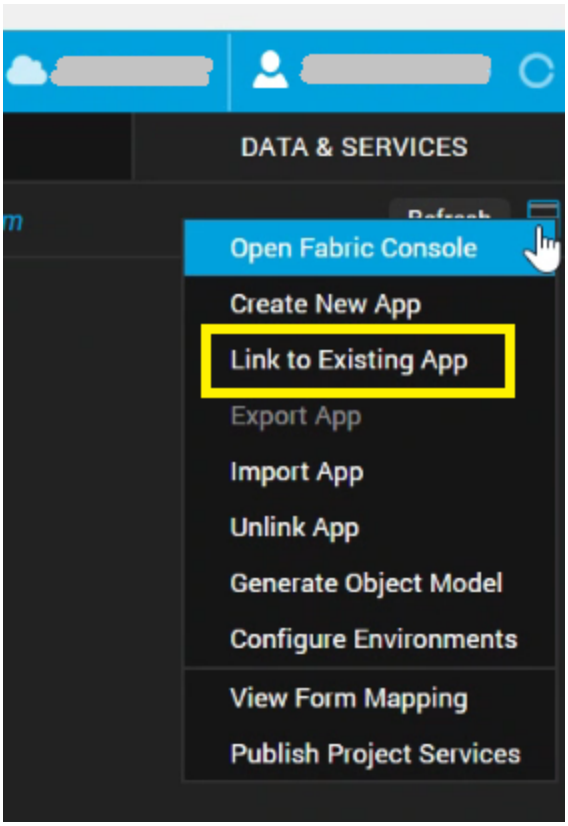
   > ***Important:*** Ensure the Kony App Store App must be published to Server before you build and publish the EAS App in Visualizer.

4. Associate the Store project with the Server app. For more information, refer to How to Link EAS Front-End Project with Back-End App.

35.5.3 Associating Front-end Project with Server App

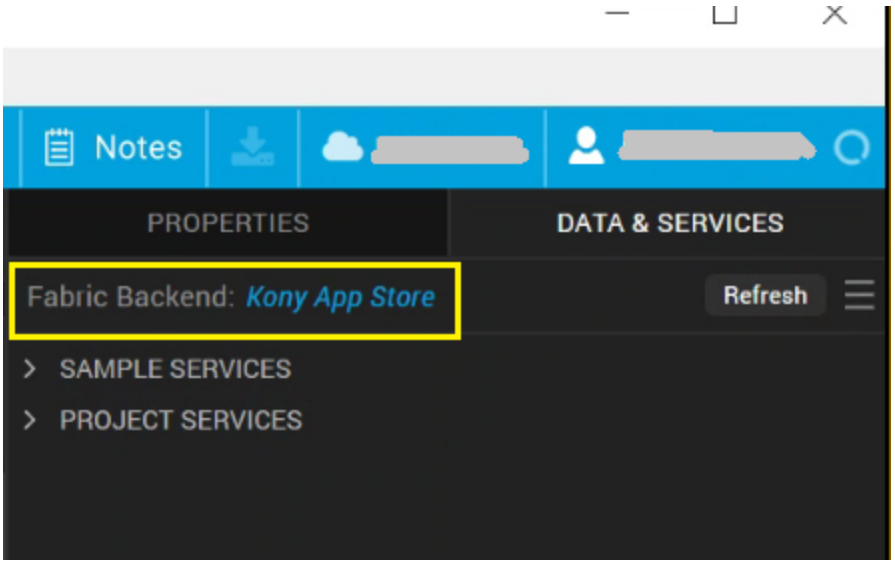
To publish your app to Kony Fabric, your Kony Visualizer client app must be associated with a Kony Fabric app. After you import the Store project in Visualizer, you must associate it with the Kony App Store App.

1. In Kony Quantum Visualizer, sign in to your Kony Cloud account. To do so, from the top right corner of the Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your Kony Cloud email and password credentials, and then click **Sign in**.
2. In Visualizer, open the **Store** project.
3. From the **Data & Services** panel, click **Open Fabric Console > Use Existing** or **Link to Existing App**. The list of pre-configured services appears.



4. In the Kony Fabric Applications, search for **Kony App Store**, and then click **ASSOCIATE**.

   The Kony App Store app is associated with the Store project.

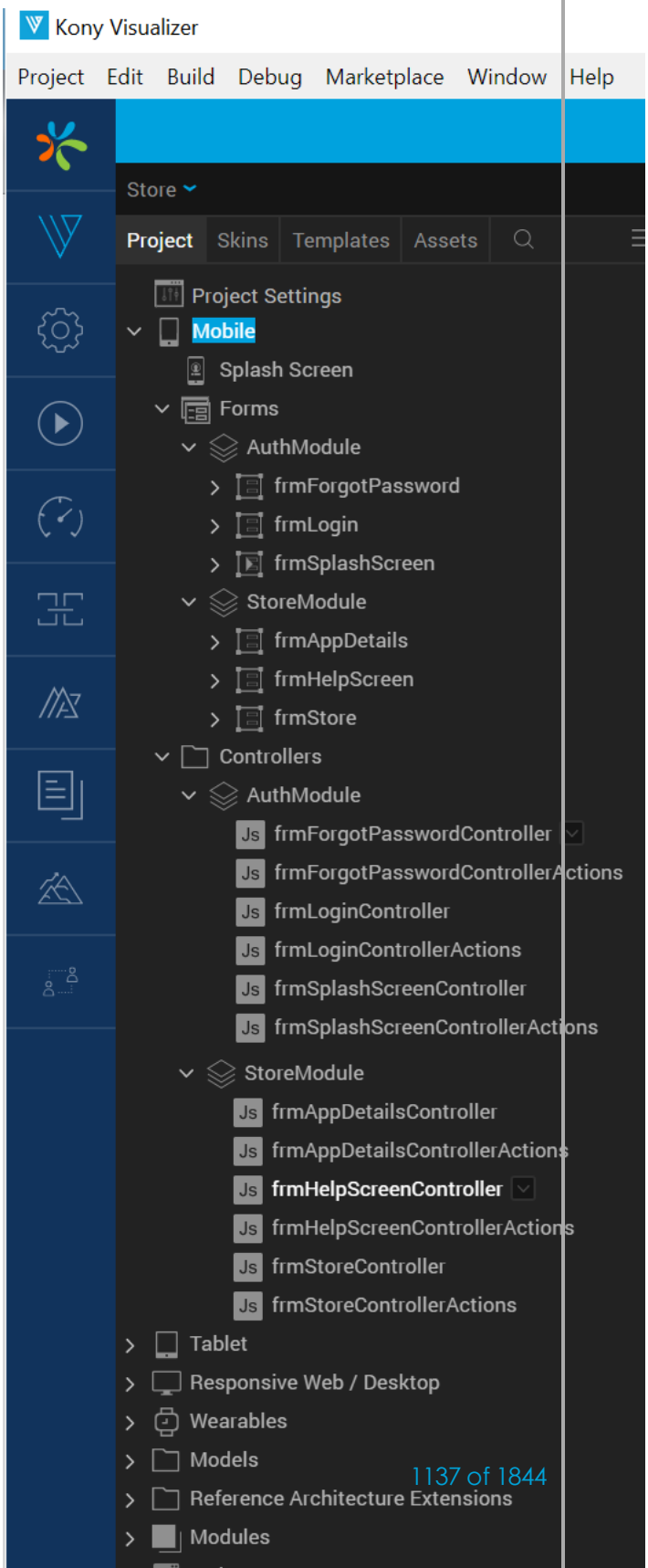


35.6 Source of EAS App

35.6.1 Structure of the EAS Front-end Project - Store Project

35.6.1.1 The Layout of the Front-End App

Kony EAS Source app is supported for Native channels: iOS and Android.

Elements in the Store Project	Project Structure UI in Visualizer
App Meta Data	

Elements in the Store Project	Project Structure UI in Visualizer
• Store app name • Default app icon These details are displayed based on the metadata of the app.	

Elements in the Store Project	Project Structure UI in Visualizer
Supported Channels in EAS	

Elements in the Store Project	Project Structure UI in Visualizer
• Mobile • Tablet • Responsive Web/Desktop	

Elements in the Store Project	Project Structure UI in Visualizer
Forms and Controllers	

<table>
<tr><th>Elements in the Store Project</th><th>Project Structure UI in Visualizer</th></tr>
<tr><td>

Forms (two modules)

- **AuthModule**
  - frmForgotPassword
  - frmLogin
  - frmSplashScreen
- **StoreModule**

<table>
<tr><th>StoreModules</th><th>Channels</th></tr>
<tr><td>frmAppDetails</td><td>• Mobile
• Responsive Web/Desktop</td></tr>
<tr><td>frmHelpScreen</td><td>• Mobile (for iOS only)</td></tr>
<tr><td colspan="2">

Note: The frmStore screen contains the Help button for iOS devices only. When you click the Help button, the help is displayed in the frmHelpScreen for iOS mobile devices. And for iOS tablets, the help is displayed in a pop-up window.

</td></tr>
<tr><td>frmStore</td><td>• Mobile</td></tr>
</table>

</td><td></td></tr>
</table>

35.6.1.2 Supported Functionality and Forms - EAS Front-End Project

- The `frmSplashScreen` form connects to the Server and fetches the login details of the Store app.
    - If the Kony User Store custom auth type is enabled, the `frmLogin` form appears.
    - If the OAuth 2.0 auth type is enabled, the login screen for that service provider type appears.

  The startup screen is displayed after the default splash screen.
    - The `frmForgotPassword` form allows you to retrieve your login password for EAS.
- The `frmAppDetails` form displays the metadata of an app.
- If auth is not enabled, the `frmStore` form is displayed with the list of published apps.

> ***Note:*** For iOS apps, if the device displays the **Untrusted Enterprise Developer** pop-up message, you must trust your iOS certificates. The `frmStore` form displays with the **Help** button 
>
> , which you can follow in your device to trust your iOS certificate. For more details, refer to Trust iOS Certificates.

35.6.1.3 Reference Architecture Extensions

- **AuthModule** supports end-user login.
    - BusinessControllers: contains custom business logic for authentication for the complete Store project.
    - PresentationController: contains auth success and failure cases. Based on each case type, the required actions are called in the form Controllers. This is a common controller for all the supported channels.

> ***Important:*** For channel specific auth success and failure cases, you must use PresentationController_<Channel> to configure auth handles.

- **StoreModule**
  - BusinessControllers: handles the fetching list of published apps, logic to download and launch apps; contains the list of apps and custom business logic for authentication for the complete Store project.
  - PresentationController: handles the presentation of apps such as in-progress status of app downloads, success, and failure cases. Based on each case type, the required actions are called in the form Controllers. You can add filters to customize the order of apps list. This is a common controller for all the supported channels.

    > ***Important:*** For channel specific presentation of apps, you must use PresentationController_<Channel> to configure store handles.

35.6.2 Source Details of EAS Back-End App - Kony App Store App

The **Kony App Store** is the Server app (Kony Fabric), which contains the following services linked to EAS.

- Identity Services: By default, the AppStoreUserRepository Identity Service of User Repository type is configured to authenticate EAS. You can configure another service type, if required.
- Objects Services: By default, the EASDownloadBinaryService Object Service of File Store type is configured to store binaries of all the published apps. You cannot modify this service.
  - By default, the EASMetaServices Object Service of Store type is configured to store the metadata of all the published apps. You cannot modify this service.
- Contains uploaded app binaries to publish them to EAS.
- Contains environments for Server and Kony App Server (EAS).

35.7 Branding Enterprise App Store to your Requirements

You can use the source app of the EAS to brand it to suit according to your business requirements.

Kony allows developers to access complete suite of the EAS App source. You can view details of the source code of the app, structure, layout, forms, and functionality, and linked services, and can customize the brand according to business requirements as well.

For example, you can brand an app for the following business requirements:

- Name, Slogan, Style
- Graphics: Logo, app icon, splash screen, and visual image of the app.
- Look of the app
- Brand keywords

35.7.1 Branding EAS App to your Company Logo and Splash Screen

35.7.1.1 Changing an App Icon

1. In Visualizer Project, click the **Splash Screen**.
2. Click the **Assets** tab, right-click **Media**, and then click **Resource Location**.
3. In the `..\\Store\resources` folder, open **common** for the channel type, for example, mobile or tablet.
4. Change the image file of the app icon. Ensure the file name is same as the existing one. For example, `icon.png`.

35.7.1.2 Replacing Image for the existing Splash Screen

The splash screen and the frmSplashScreen can be same or different. You can modify the splash screen in the following two ways:

To replace the existing splash screen, follow these steps:

1. In Visualizer Project, click the **Splash Screen**.
2. Click the **Assets** tab, right-click **Media**, and then click **Resource Location**.
3. In the `..\\Store\resources` folder, open **common** folder.
4. Replace the new splash screen with the one existing. Ensure the file name is same as the existing one. By default, the image resource name is `splash_eas.png` for mobile and tablet. By default, the image resource name is `splash_screen.png` for responsive web.

35.7.1.3 Adding New Image to Splash Screen

If you choose to add new image to splash screen, the new image name must be

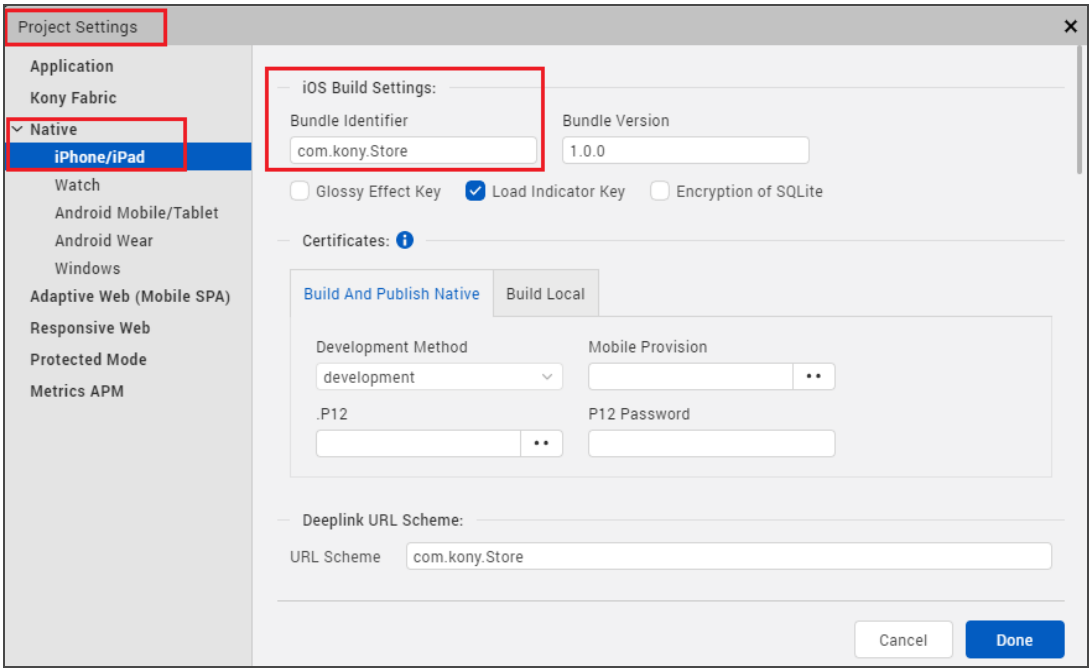
1. In Visualizer Project, click **Splash Screen**.
2. Click the **Assets** tab, right-click **Media**, and then click **Resource Location**.
3. In the `..\\Store\resources` folder, open **common** folder.
4. Add the new splash screen file to the common folder.
5. Now, you must link the new image to the start-up screen in your store project as follows:
    a. In **Visualizer > Store** project, expand the channel type. For example, **Mobile**.
    b. Click **Splash Screen**.
    c. In Properties > **Splash Screen > General**, click **Edit**.
    d. In the **Splash Image** dialog box, select the new file.
6. Go to **Forms** > **AuthModules** > **frmSplashScreen**, double-click the **Splash Screen** and select the new image file.

> *Important:* All images must be maintained with the default set scaling.

35.7.1.4 Changing the EAS Package Name

If you want to build the EAS app with a custom **Package Name** and **Bundle Identifier**, you must configure the native settings in the Store project that you imported to Visualizer.

- iOS
    1. Ensure that you imported the Store.zip app into Visualizer project.
    2. In Visualizer, open **Project Settings**.
    3. Open **Project Settings**, click **Native**.



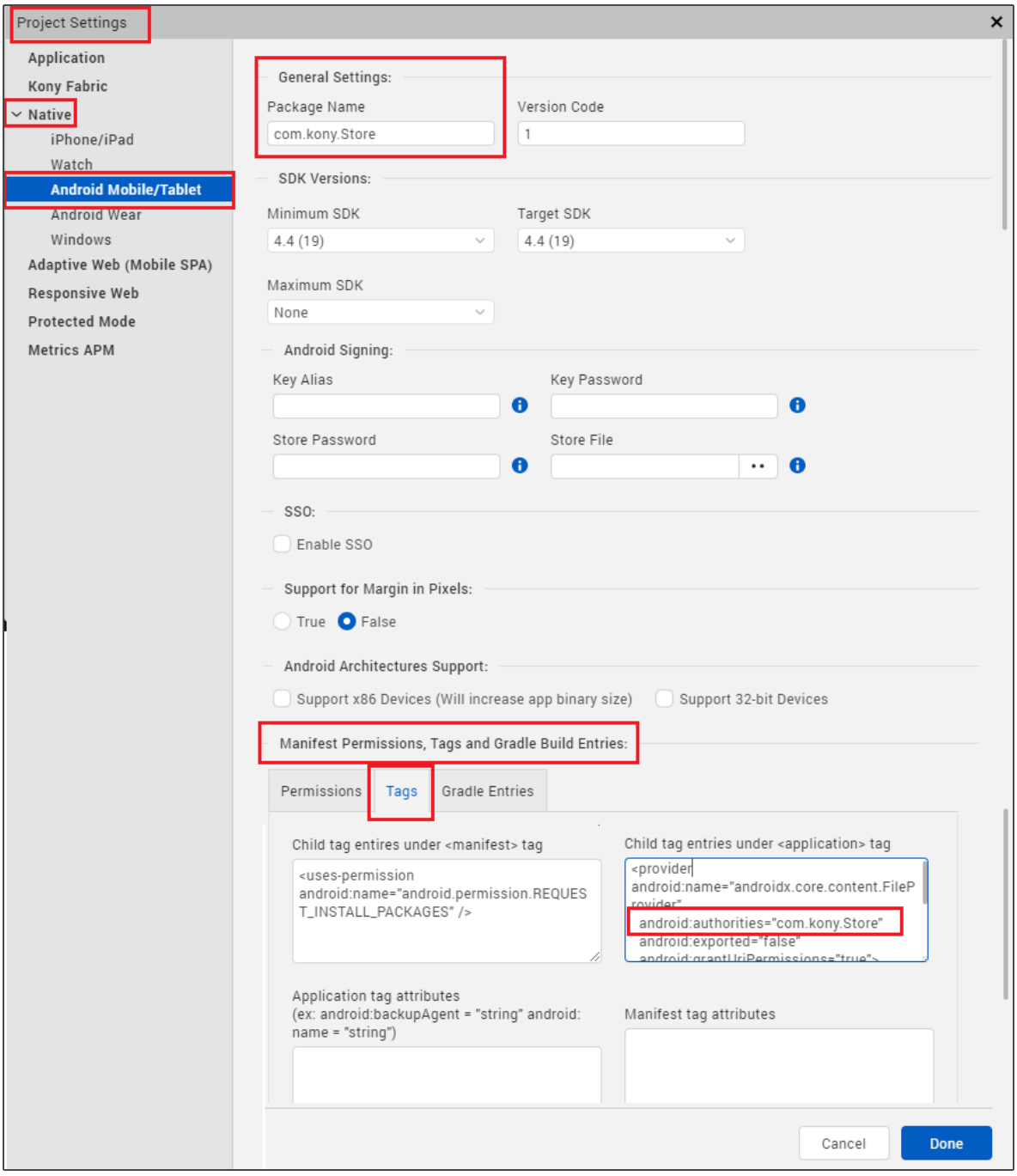
    4. Click **iPhone/iPad** > **iOS Build Settings** > **Bundle Identifier**, change the package name.

> ***Important:*** The **Bundle Identifier** must match with the iOS Certificates that you configured in the Store project. For more details, refer to Platform specific prerequisites.

    5. Click **Done**.

- **Android**
    1. Ensure that you imported the Store.zip app into Visualizer project.
    2. In Visualizer, open **Project Settings**.
    3. In the **Project Settings** dialog, click the **Native**.
    4. Click **Android Mobile/Tablet**, and do the following:
        a. Under **General Settings** > **Package Name**, specify the desired package name.
        b. Under **Manifest Permission, Tags and Gradle Build Entries**, click the **Tags** tab.
        c. Under the `Child tag entries under <application> tag` text box, specify the same package name value for `android:authorities="<package_name>"`.

5. Click **Done**.



35.7.2 Adding Contact Us Form and Support Details

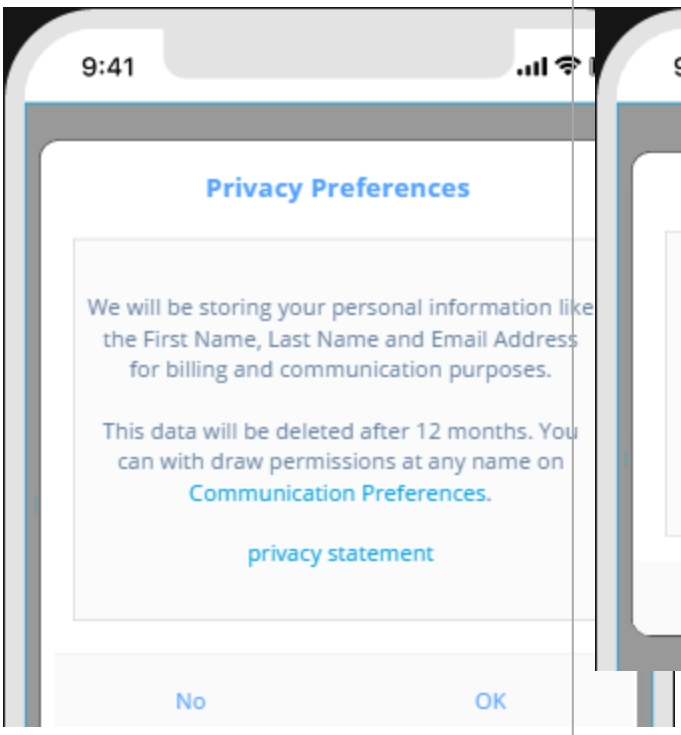
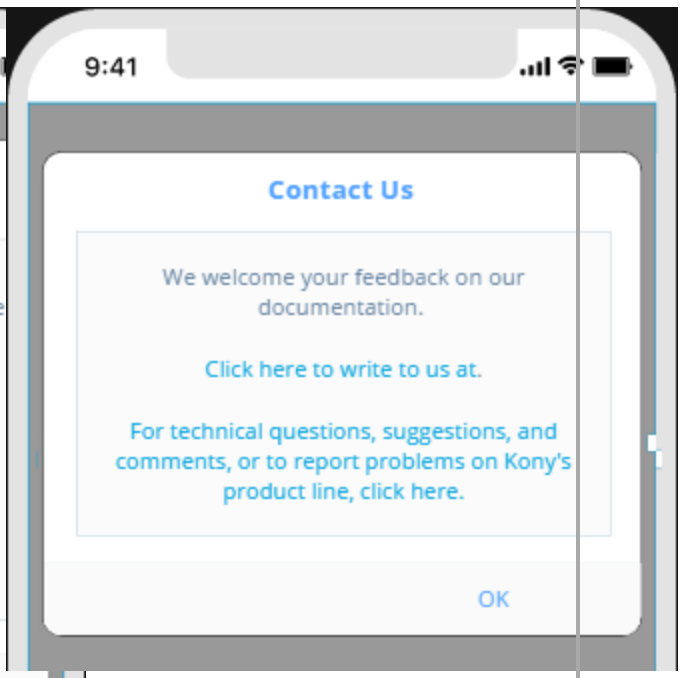
You can add new forms to your EAS App as per your business requirements.

For example, to add the **Contact Us** form to your EAS App and provide a link to the new form within the **StoreModule > fromStore** form. The following steps details how to add a **Contact Us** form, provide contact details, and links to the form for a Mobile channel. In this example, you will be using the sample component: **privacypreferences**, which contains ready-to-use text placeholder and action buttons to navigate between forms.

> ***Important:*** While modifying the source app, ensure that you follow Kony Reference Architecture and Extensibility Guidelines lines to support product upgrades.

1. Add new form to the project as follows:
    a. In **Visualizer**, open the **Store** project, and ensure you are in the **DESIGN** mode.
    b. Click **Store project > Mobile**, right-click **Forms**, and click **New Group**.
    c. Right-click the NewGroup and rename the new group to `ContactUS`.
    d. Right-click the ContactUS Group, and click **New Form**.
    e. Rename the new form to `frmContactUs`.
2. Download the **Privacy Preferences (GDPR)** component from Marketplace. To do so, click **Marketplace** menu > **Browse** and search for Privacy Preferences (GDPR). Click **Import to Collection Library**.
3. Click `frmContactUs` form.
4. From the downloaded components section, drag and drop the **privacypreferences** component to the `frmContactUs` form. The privacypreferences component is added to the form. You can change the caption, as follows:
    - In **Properties** panel, click **Component**.
    - Change the **ID** to `ContactUsDetails`.

    Now, you will be modifying the actions and text in the component details section according to the your business requirement for the **Contact Us** form as shown in the following table.

Default text and actions in the Component	Modified details in the Component section for the Contact Us form
	

5. Click `ContactUsDetails` to edit component properties.
    a. In **Properties** > **Component**, do the following:
        i. In **Header Text**, change the text to `Contact Us`.
        ii. In **Message**, specify the contact details.

For example, Kony Support URL:

```
<p>We welcome your feedback on our documentation.
<br><br><a href="https://techpubs@kony.com/">Click here
to write to us at</a>.<br><br>\n<a
href="https://support@kony.com/">For technical questions,
suggestions, and comments, or to report problems on
Kony's product line, click here.</a>\n</p>
```

b. To add the link to the **OK** button, do the following:

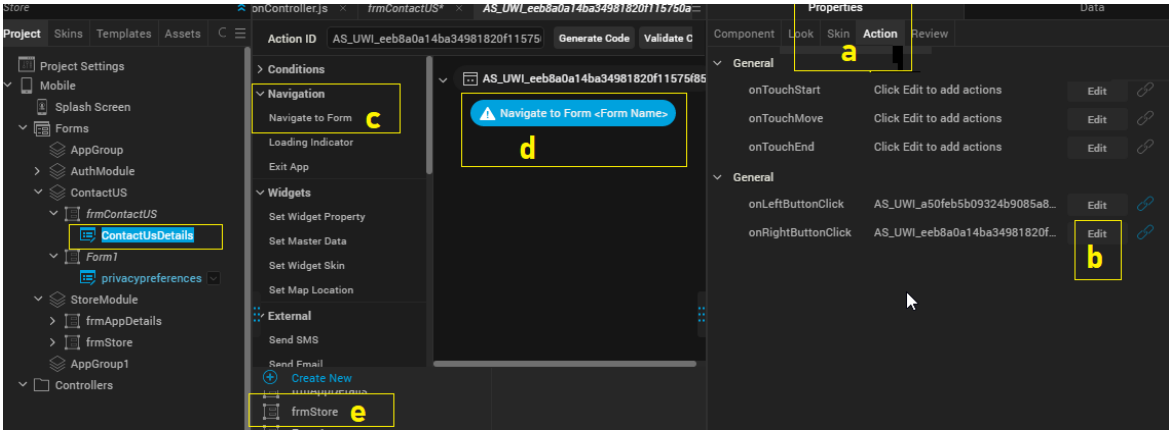
   i. On **Properties**, click **Action**.

   ii. From **onRightButtonClick**, click the **Edit** button.

   iii. From **Action ID > Navigation**, click **Navigate to Form**.

   iv. Click the `Navigate to Form <Form Name>` element.

   v. Click `frmStore`. By doing this, when you click the **OK** button on the **Contact Us** form, the `frmStore` form appears.



c. For the **Contact Us** form, you do not need any action for the **No** button. So, you can hide the button by removing the text, as follows:

i. From **Properties** panel, click **Component**.

ii. In **Left Button Text** field, delete the caption.

6. Now, add the link to the new button widget in the **frmStore** form to navigate to the **Contact Us** form.
   a. Select the `frmStore` form.
   b. From **Visualizer > Default Library**, drag and drop the **Button** widget to the form.

      You can change the button properties such as skin and text.
   c. On **Properties**, click **Action**.
   d. From **onClick**, click the **Edit** button.
   e. From **Action ID > Navigation**, click **Navigate to Form**.
   f. Click the `Navigate to Form <Form Name>` element.
   g. Click `frmContactUs`. By doing this, when you click the newly added widget on the `frmStore` form, the `frmContactUs` form appears.

      You can build or preview the EAS App to view the customized results now.

35.7.3 Modifying Server EAS App for Identity Services and Metadata of Binaries

35.7.3.1 Configuring Identity Services for EAS Authentication

You can configure new Identity Services and Object Services for EAS in the **Kony App Store** App. After you configure the new services, you must map the new services in the project and update the source code of the Store project.

35.7.3.2 Configuring Properties for Client Binaries

For information on how to configure basic properties such as the description of a client binary file, the icon of a binary, and the related screenshots of a binary file. refer to Configuring Properties for Client Binaries in Kony Fabric.

35.8 Publishing EAS App

EAS native apps need to be distributed in the same way as you distribute the other native apps. So, to distribute EAS native apps to end-users, you must publish your EAS native apps to the EAS Web app. End-users can access the EAS Web app to view and download EAS native apps on devices like any other native apps. By default, the EAS Web app is selected while publishing the apps to EAS.

35.8.1 Build and Publish App Binaries to EAS using Kony Visualizer

With Visualizer Starter, you can build client binaries and publish them directly to Kony App Server from Visualizer. For more information, refer to Publishing Native Apps to Enterprise App Store from Visualizer Starter.

- For more information on how to build app binaries and publish them to EAS, refer to **Publish to Enterprise App Store section** in Publishing Client Binaries to Kony App Server from Kony Visualizer (for Cloud only).

35.8.2 Publish App Binaries to EAS using Kony Fabric

By using Kony Fabric, you can upload the client binaries and publish them to Kony App Server. After an app is published to Kony App Server, an authorized end-user can access the EAS Client app to view the available applications and download them using a mobile device.

- For more information on how to upload app binaries to the Server and publish them to EAS by using Kony Fabric, refer to Publishing Client Binaries to Kony App Server from Kony Fabric.

36. Known Issues - Enterprise App Store

- In iOS 13 devices, the network error alert gets dismissed without user action.
- If you build your EAS native client app by using Visualizer version in between **8.4.28** and **8.4.50** and Fabric on-premises, the app will get stuck in the splash screen when you launch it.

  In this case please configure your project using the following steps:

  1. Ensure that you imported the Store project in Visualizer.
  2. Navigate to **Project > Reference Architecure Extentions.**
  3. Go to the **ConfigManager > BusinessControllers >** `BusinessController`.
  4. In the list of configurable parameters inside the `configManager()` file, add the snippet: `this.serviceUrl ="SERVICE_URL_OF_YOUR_ENVIRONMENT";`
  5. Scroll down in the same file and search for `"serviceUrl" : appConfig.serviceUrl` in the `getAppProperties` method.
  6. Replace the code with `"serviceUrl" :this.serviceUrl`.
  7. Rebuild the app.

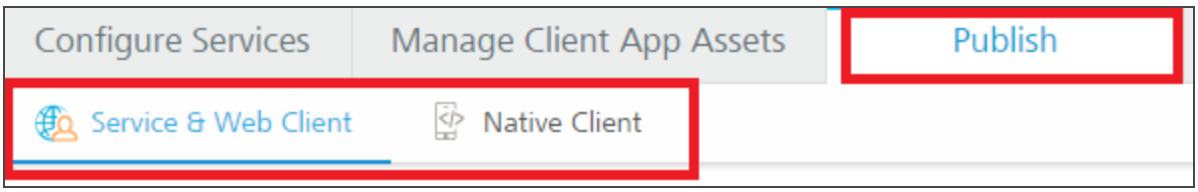
     > ***Important:*** You must configure the `"serviceUrl"` whenever you switch from one environment to the other, follow the steps mentioned in this section.

37. Publish

After you have configured all the required services for an app, you need to publish the app. Publishing allows your app to start using the Kony service in real-time. After an app is published, Kony Fabric generates the code that you can integrate with platform SDKs.

> *Important:* When you publish an app to an environment, all the identity services associated with the app are published only to the selected run-time environment. The latest published Identity Services will affect any other apps in the same environment if they use these identity services.
>
> The latest published identity services on the current environment will not affect any apps in different environments.
>
> If you are making an identity call directly (without SDK), you must provide the app key (or) secret header in the payload in below format:
> **For Basic Login - Headers**:
> `X-Kony-App-Key: <app-key-here>`
> `X-Kony-App-Secret: <app-secret-here>`
> `POST <identity-login-url>?provider=<identity-service-name>`
>
> **For OAuth and SAML Login**: Follow these steps to provide the app key (or) secret header
> **Step 1**: GET the Authorization URL to pass the app key as a query parameter.
> **Authorization call**: `GET <authorization>?appkey=<app-key-here>`
>
> **Step 2**: **Token Call - Headers**:
> `X-Kony-App-Key: <app-key-here>`
> `X-Kony-App-Secret: <app-secret-here>`
> `POST <token-endpoint>`

In Kony Fabric Console, the **Publish** tab of an app contains the following tabs:



- **Service & Web Client**: Helps you to publish the app services (Web client binaries) to the server. The app services you configure under the **Configure Services** tab such as identity, integration, orchestration, synchronization, and engagement.

  If you upload web binaries (.war) for **Web** under the **Manage Client App Assets** tab, you can publish these web binaries only to the server.

  - How to Publishing Services and Web Client Binaries to the Server

- **Native Client**: Helps you to publish native client binaries for platforms such as for **iOS**, **Android**, and **Windows** to a Kony Management environment. However native client binaries only available in Kony Enterprise Store if state is selected as **Active** and not Draft. You need to upload these binaries under the **Manage Client App Assets** tab.

  > ***Note:*** The .war file you upload for **Web** platform under the **Manage Client App Assets** tab is only published to the Server.

  - How to Publishing Client Binaries from Kony Fabric

  > ***Important:*** Make sure that your Kony Fabric Console version used for creating apps is same as Kony Fabric runtime components' (Integration, Sync, and Engagement) versions. All the components of Kony Fabric must be upgraded to same versions.
  >
  > For example, if the Kony Fabric Console installed version is V8 for creating your apps, you must use the same Kony Fabric version for runtime components to publish your apps. If there is a version mismatch, Kony Fabric's Publish functionality and other runtime server components may not work as expected.

37.1 Asynchronous Publish Apps in Kony Fabric Console

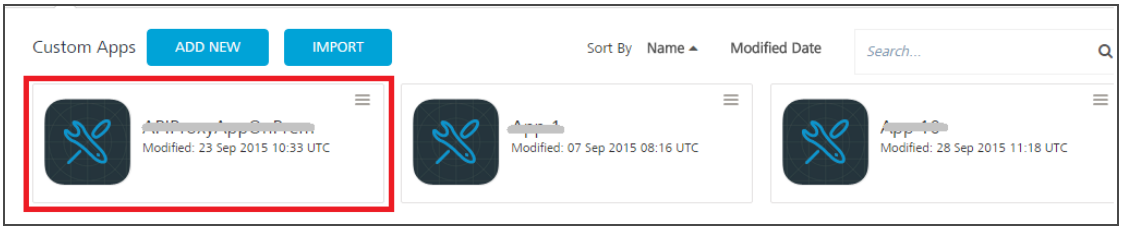
When you publish apps in the Console, the apps are published with the Asynchronous publish mode. By default, the asynchronous publish is enabled. When asynchronous publish is initiated, the **Publish** dialog can be closed and the app publish process continues in the background. The Asynchronous publish helps you to unblock the user interface (UI) when an app publish is in progress. While the app is in asynchronous publish progress, you can continue work in the console such as creating or modifying other apps and services. When you publish apps, your apps are published to clouds or environments.

> ***Note:*** For more details on publish apps synchronously, refer to Synchronous Publish.

> ***Note:*** **For on-premises only:**
>
> To skip a .war file if you have uploaded it for **Web** platform under the **Manage Client App Assets** tab, select the **Allow Manual Publish Only** check box in the **Add a New Environment** window.

To publish an app asynchronously, follow these steps:

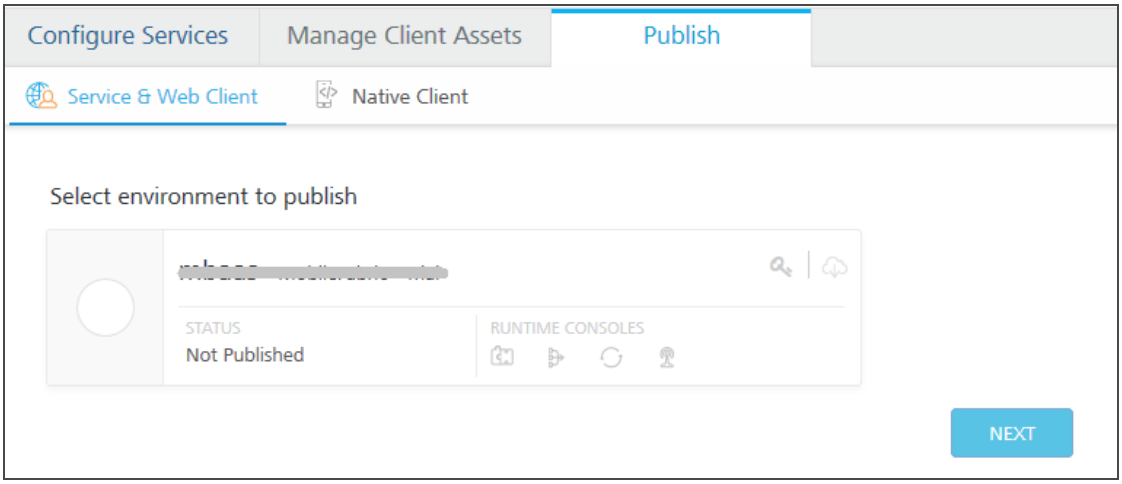
1. From the **Applications** page, select the desired app.



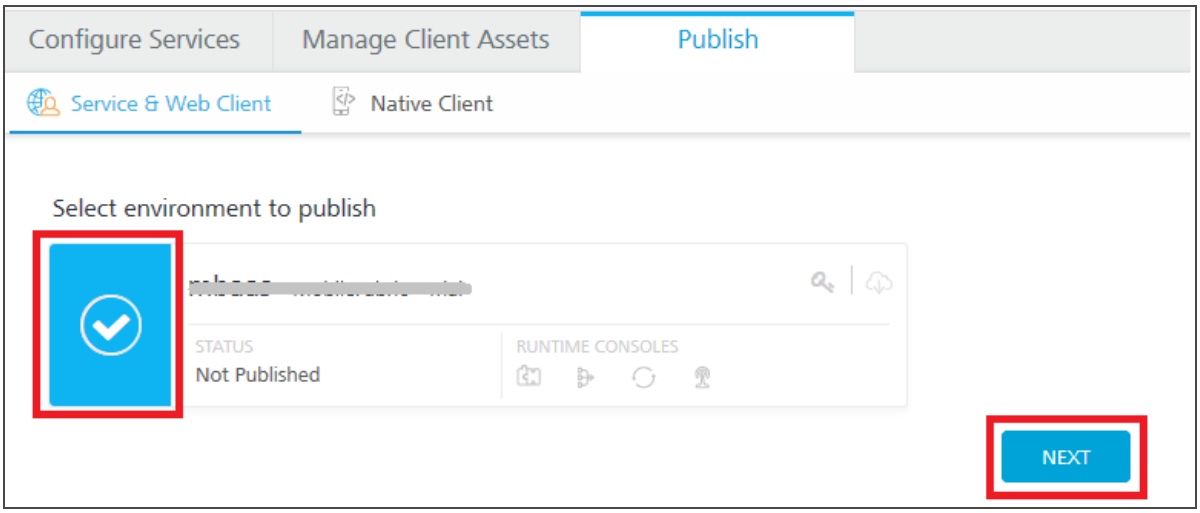
2. Click the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected and lists clouds or environments configured for the Kony Fabric account. The list also displays the following app status for that cloud or environment.

- **Published**: An app is published to a cloud or environment. You can unpublish the app, if required.
- **Not Published**: An app is not published to a cloud or environment. You can publish the app, if required.
- **Error**: An app is canceled while publishing or unpublishing. You can publish or unpublish the app, if required.

> ***Note:*** Ensure that you have added services (for example, identity service, integration service, sync service, and messaging service) for your application before publishing.
>
> To publish native client binaries to a Kony Management Environment, refer to How to Publish Native Client Binaries from Kony Fabric to Kony Management Environment

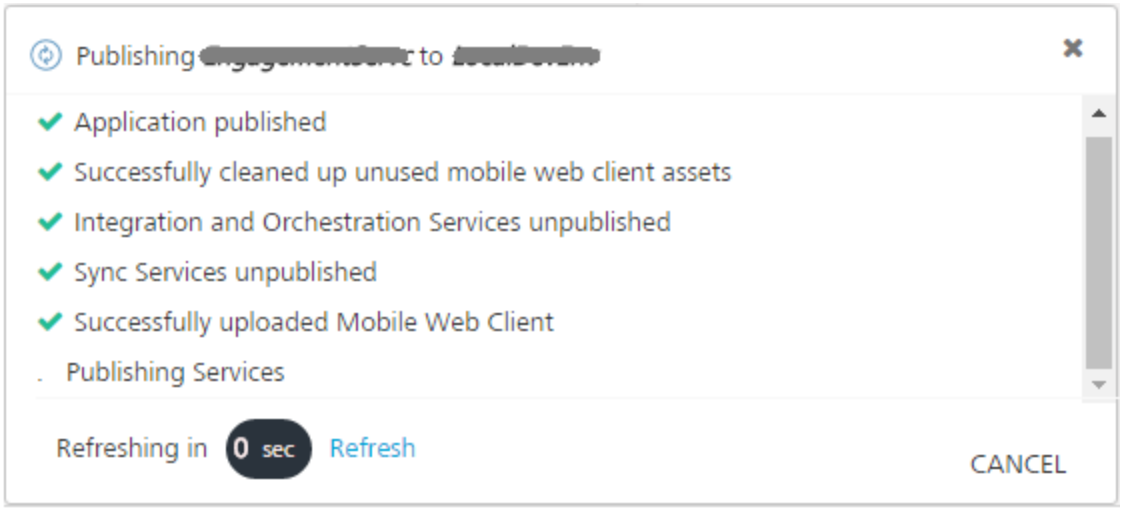


3. Under the **Select environment to publish**, click an environment.



> ***Note:*** The **CONFIGURE & PUBLISH** and **PUBLISH** buttons are dim when you have not selected any environment. When an environment is selected, only then is the **CONFIGURE & PUBLISH** and **PUBLISH** buttons are available.

4. Click the **PUBLISH** button. The process of publishing the app begins.



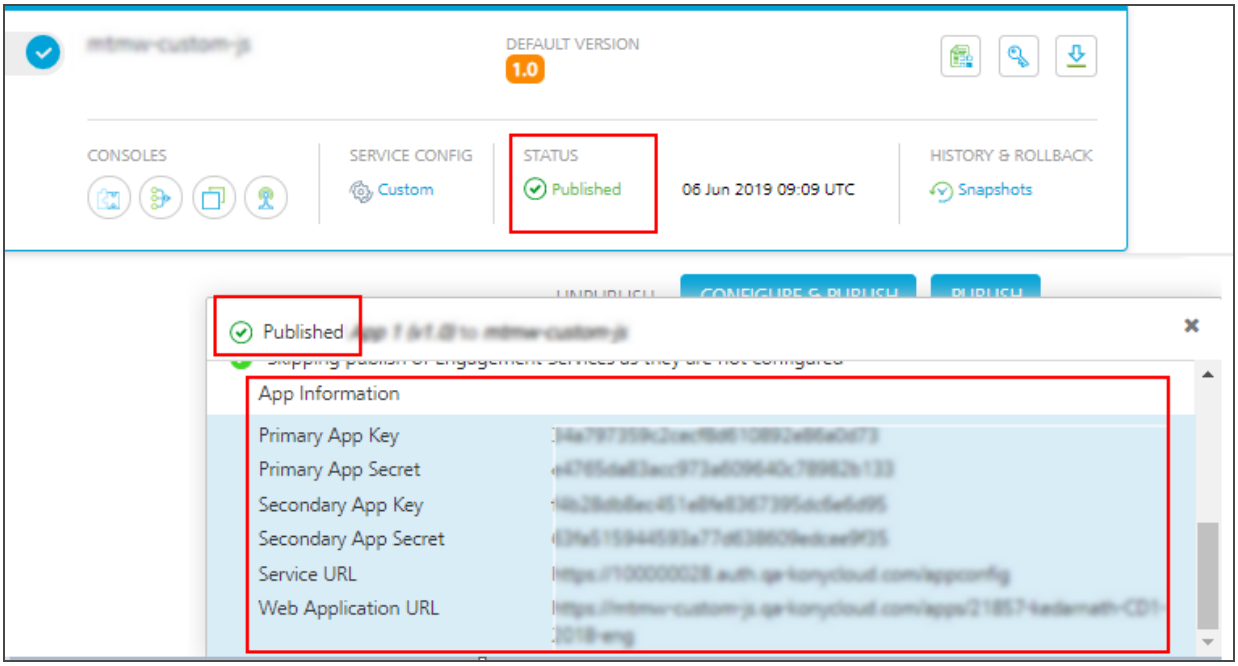
> ***Note:*** The asynchronous publish feature helps you unblock the UI, and then you can continue work in the console during an app publish.

5. Click the **Close** button in the **Publishing** dialog to continue with asynchronous app publishing. The app publish process continues in the background.

   The following actions can be performed in the **Publishing** dialog:

   - To force a refresh of the app publish progress, click **Refresh**. The app publish also gets auto-refreshed at every three seconds for on-premises. For Kony Cloud, the app publish gets auto-refreshed at every five seconds.
   - To cancel the publishing, click **CANCEL**. If the cancel is successful, the status changes from **in progress** to **Error**.
     For more details about app publish stages, refer to Publish Life-cycle.

   After the app is published, the app status changes to **Published** in the **Select environment to publishing** section. The **Publishing** dialog also displays details for published services and primary app key/app secret, secondary app key/app secret, service URL, and Web application URL. For more information, refer to Separate App Key/App Secret for Native and Web Channels.



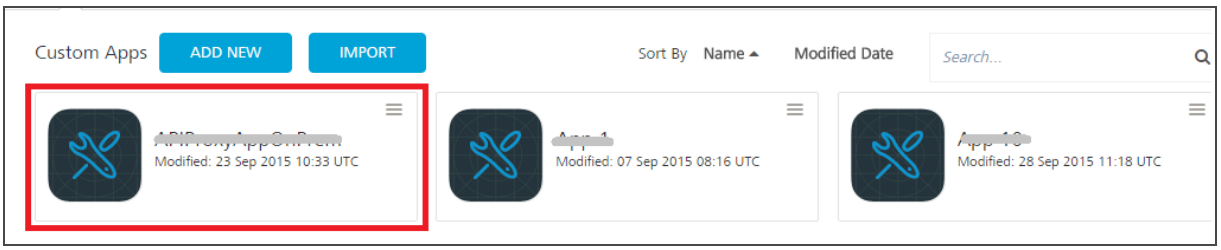
> ***Note:*** You can also publish an app via API. For more details, refer to Continuous Integration - Publish an app via API

37.1.0.1 How to Asynchronous Unpublish an app in Kony Fabric Console

Unpublishing an app allows you to modify, unlink, or delete a service. During the time an app is unpublished, the end users cannot access the app.

To unpublish an app asynchronously, follow these steps:

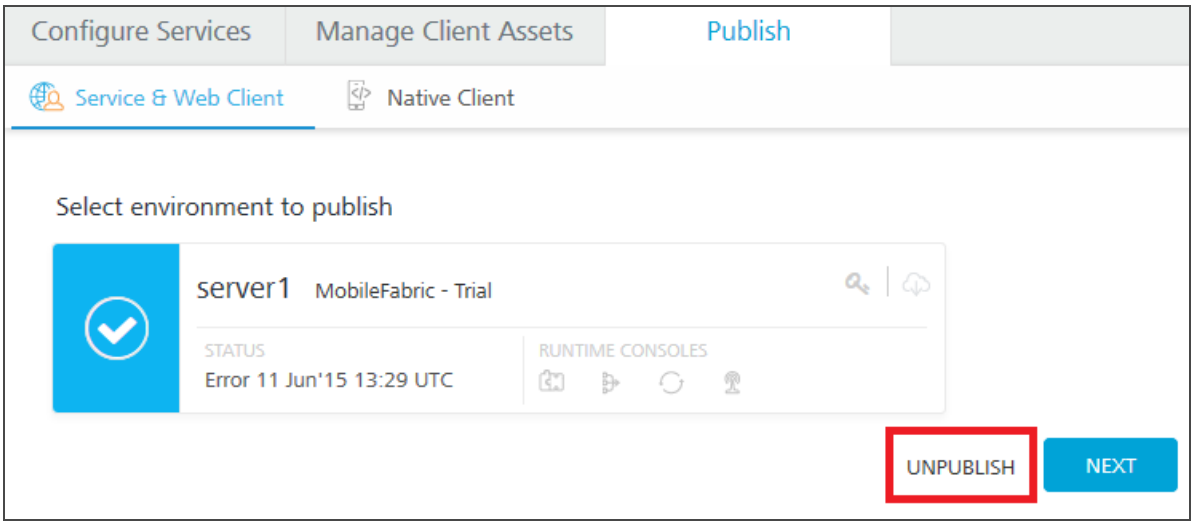
1. In the **Applications** page, click an app. The application details page displays.



2. Click the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected and lists clouds or environments configured for the Kony Fabric account.

3. Under the **Select environment to publish**, select the environment and then click **UNPUBLISH.**

> ***Note:*** When an environment is published, only then is the **UNPUBLISH** button available.



For more details about app publish stages, refer to Publish Life-cycle.

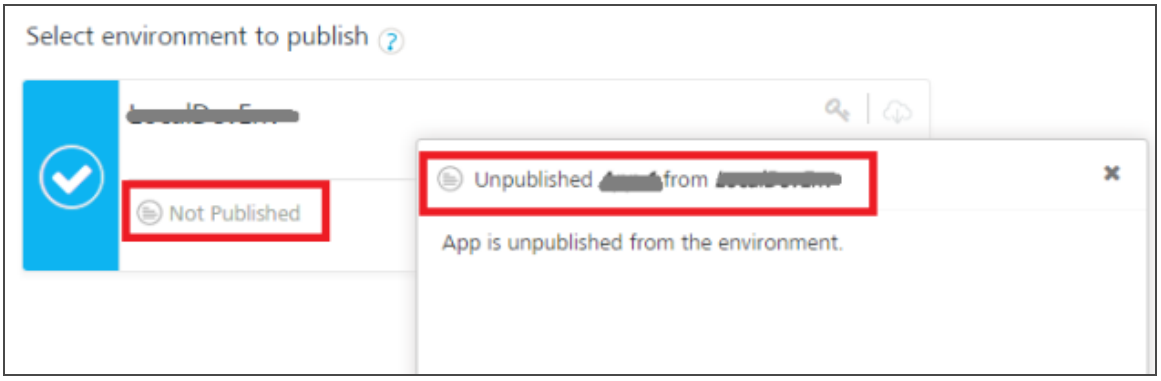
4. The console displays the alert message for confirmation of unpublishing the app. Click **OK** to confirm the unpublishing.
The unpublishing process begins.

> ***Note:*** The asynchronous publish feature enables you to unblock the UI, and then you can continue work in the console while an app unpublish is in progress.

The following actions can be performed in the **Unpublish** dialog:

- To force a refresh of the app unpublish progress, click **Refresh**. The app unpublish also gets auto-refreshed at every three seconds for on-premises. For Kony Cloud, the app unpublish gets auto-refreshed at every five seconds.

- To cancel the unpublishing, click **CANCEL**. If the cancel is successful, the status changes from **in progress** to **Error**.



After the app is unpublished, the app status changes to **Not Published** in the **Select environment to publishing** section.

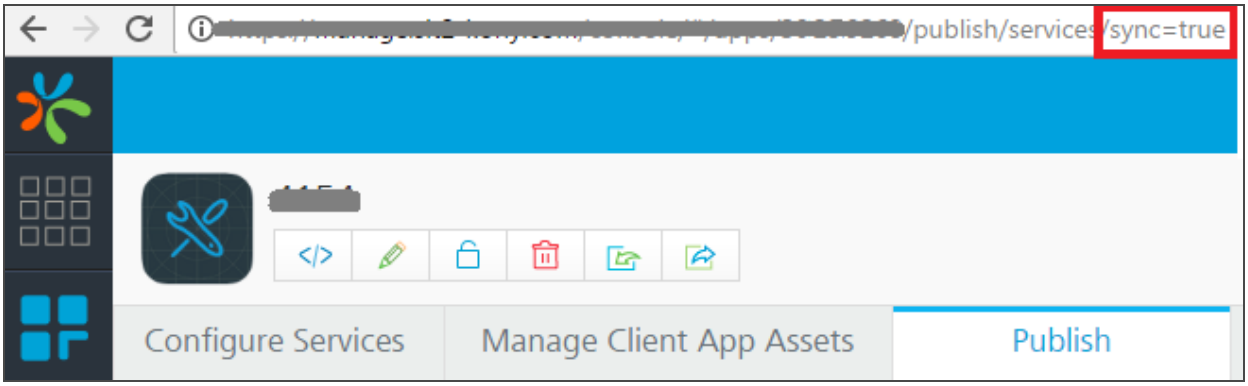
> ***Note:*** You can also unpublish an app via API. For more details, refer to Continuous Integration - Unpublish an app via API

37.2 Synchronous Publish Apps in Kony Fabric Console

When you publish apps, your apps are published to clouds or environments. By default, the asynchronous publish[1] is enabled. If you are required to publish apps synchronously, you can enable synchronous publish by adding the parameter `sync=true` at the end of the **Console** URL while you are in the **Publish** page of an app.

For example: `https://konyfabric_console/apps/<appid>/publish/services?sync=true`

[1]When asynchronous publish is initiated, the Publish dialog can be closed and the app publish process continues in the background. Also the user interface (UI) is unblocked, and you can perform other actions in the Console such as creating or modifying other apps and services.



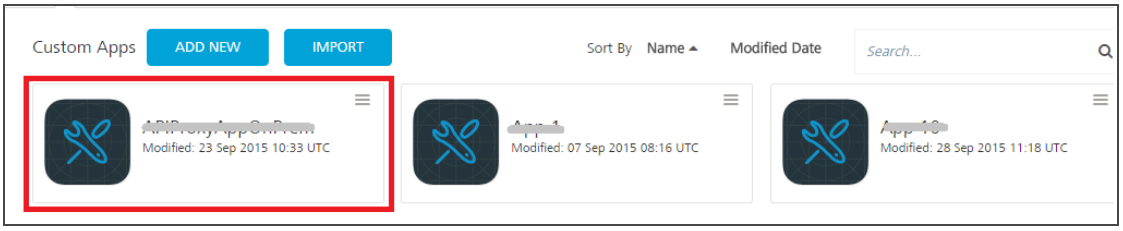
When you enable synchronous publish, the UI in the Console remains blocked until the initiated operation or app publish progress is finished. As the user interface (UI) is blocked, you cannot perform other actions in the Console such as creating or modifying other apps and services.

> ***Note:*** **For on-premises only:**
>
> To skip a .war file if you have uploaded it for **Web** platform under the **Manage Client App Assets** tab, select the **Allow Manual Publish Only** check box in the **Add a New Environment** window.

37.2.0.1 How to Synchronous Publish an app in Kony Fabric Console

To publish an app synchronously, follow these steps:

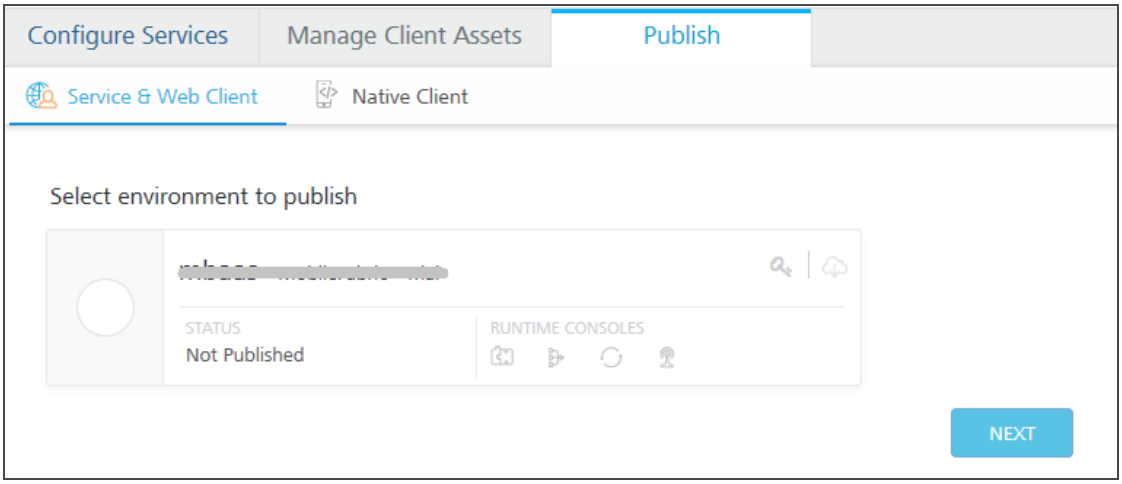
1. From the **Applications** page, select a desired app.



2. Click the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected and lists clouds or environments configured for the Kony Fabric account. The list also displays the following app status for that cloud or environment.

- **Published**: An app is published to a cloud or environment. You can unpublish the app, if required.
- **Not Published**: An app is not published to a cloud or environment. You can publish the app, if required.
- **Error**: An app is canceled while publishing or unpublishing. You can publish or unpublish the app, if required.

> ***Note:*** Ensure that you have added services (for example, identity service, integration service, sync service, and messaging service) for your application before publishing.
>
> To publish native client binaries to an EMM Environment, refer to How to Publish Native Client Binaries from Kony Fabric to EMM Environment



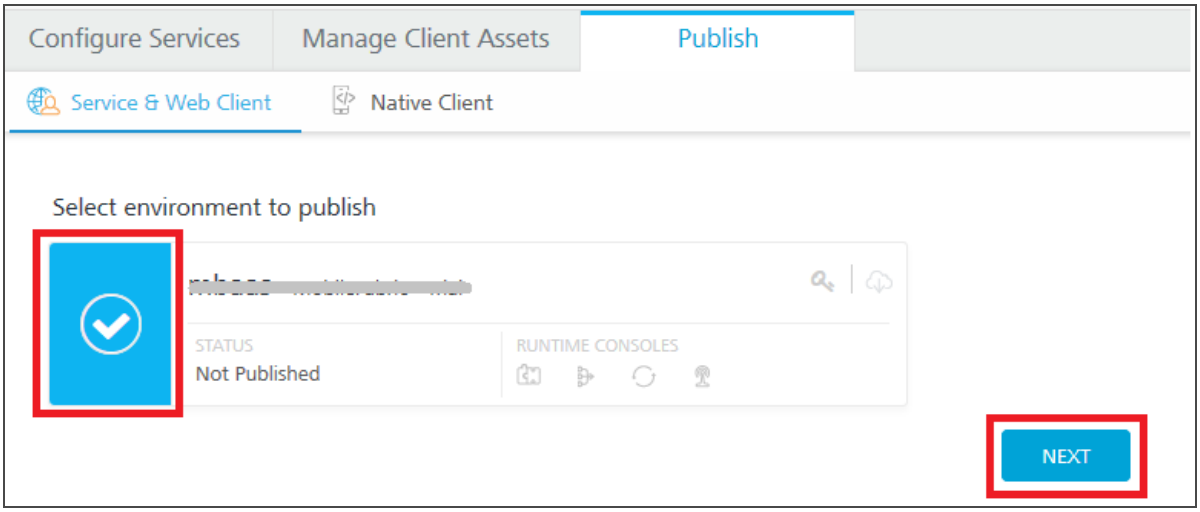
3. To enable synchronous publish, follow these steps:
    a. In the **Publish** page of the app, go to the end of the Console URL, and then add a parameter '`sync=true`'.

For example:`https://mobilefabric_console/apps/<appid>/publish/services?sync=true`

b. Press **Enter** key. The **Publish** page gets reloaded and enabled for synchronous publishing.

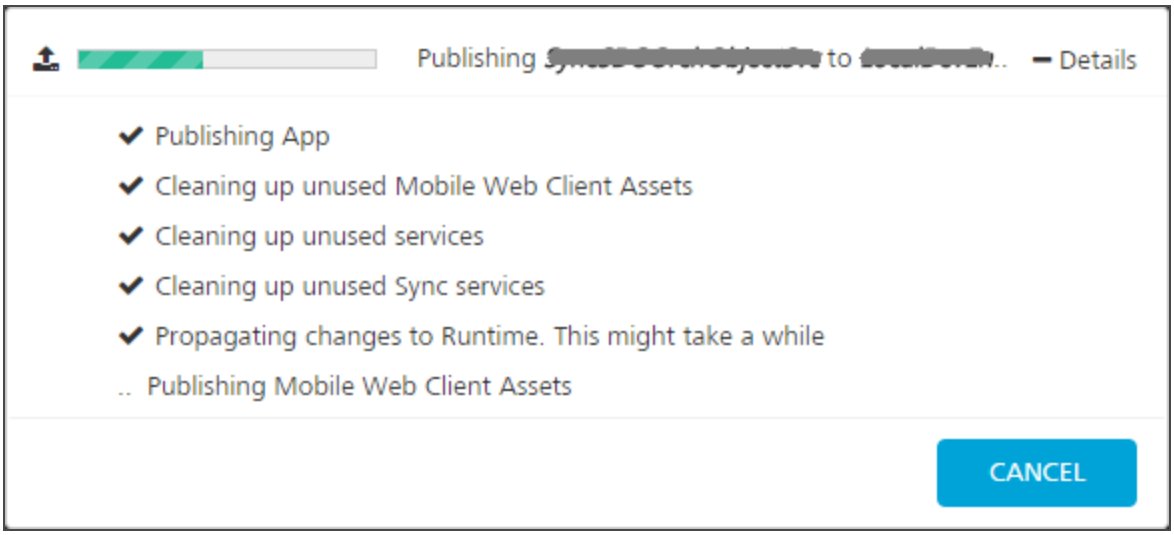
> ***Note:*** The asynchronous publish feature enables you to unblock the UI while an app publish is in progress.

4. Under the **Select environment to publish**, click an environment.

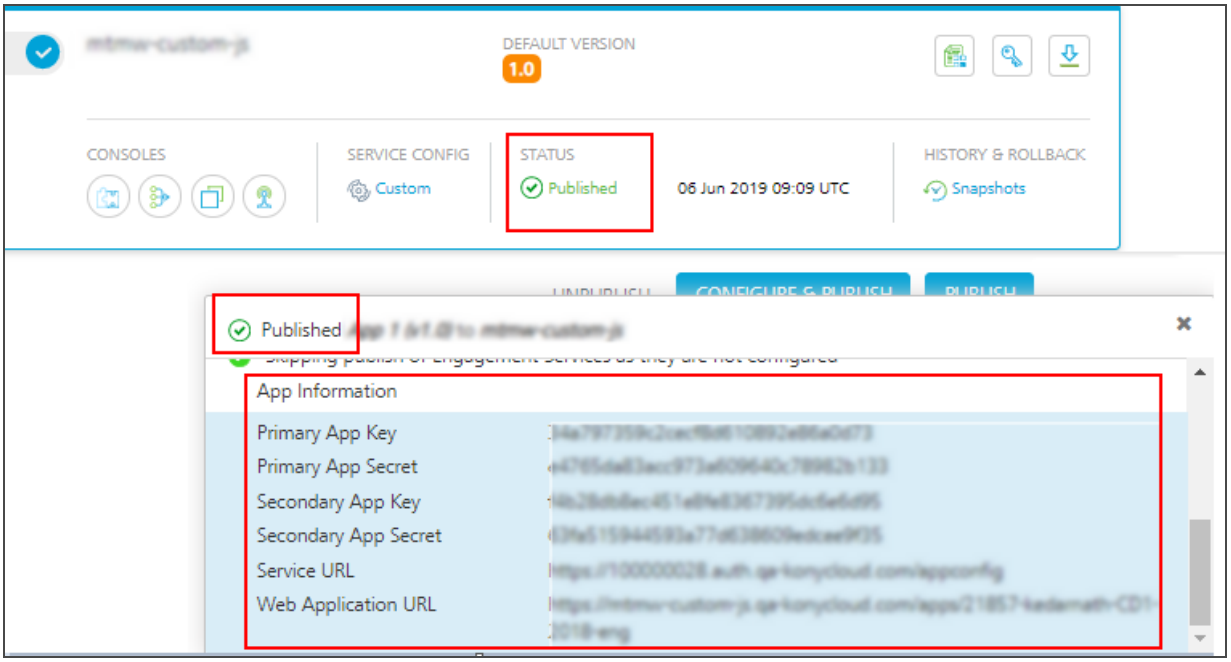


> ***Note:*** The **CONFIGURE & PUBLISH** and **PUBLISH** buttons are dim when you have not selected any environment. When an environment is selected, only then is the **CONFIGURE & PUBLISH** and **PUBLISH** buttons are available.

5. Click the **PUBLISH** button. The process of publishing the app begins.



After the app is published, the app status changes to **Published** in the **Select environment to publishing** section. The **Publishing** dialog also displays details for published services and primary app key/app secret, secondary app key/app secret, service URL, and Web application URL. For more information, refer to [Separate App Key/App Secret for Native and Web Channels](#).



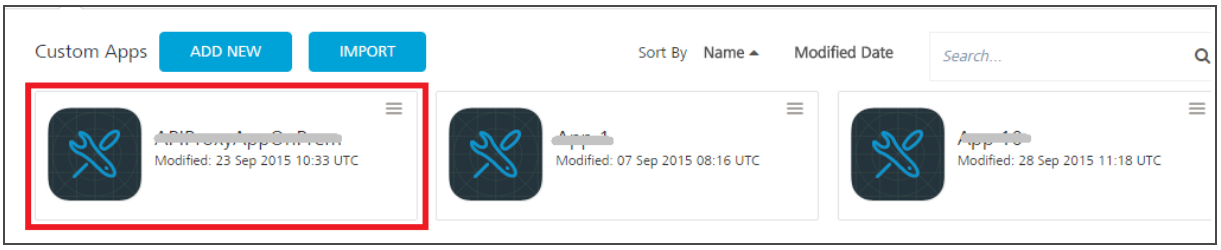
> ***Note:*** You can also publish an app via API. For more details, refer to Continuous Integration - Publish an app via API

37.2.0.2 How to Synchronous Unpublish an App in Kony Fabric Console

Unpublishing an app allows you to modify, unlink, or delete a service. During the time an app is unpublished, the end users cannot access the app.

To unpublish an app synchronously, follow these steps:

1. In the **Applications** page, click an app. The application details page displays.



2. Click the **Publish** tab. By default, the **Service & Web Client Publish** tab is selected and lists clouds or environments configured for the Kony Fabric account.
3. To enable synchronous unpublish in the **Publish** tab, follow these steps:
    a. In the **Publish** page of the app, go to end of the Console URL, and then add a parameter '`sync=true`'.

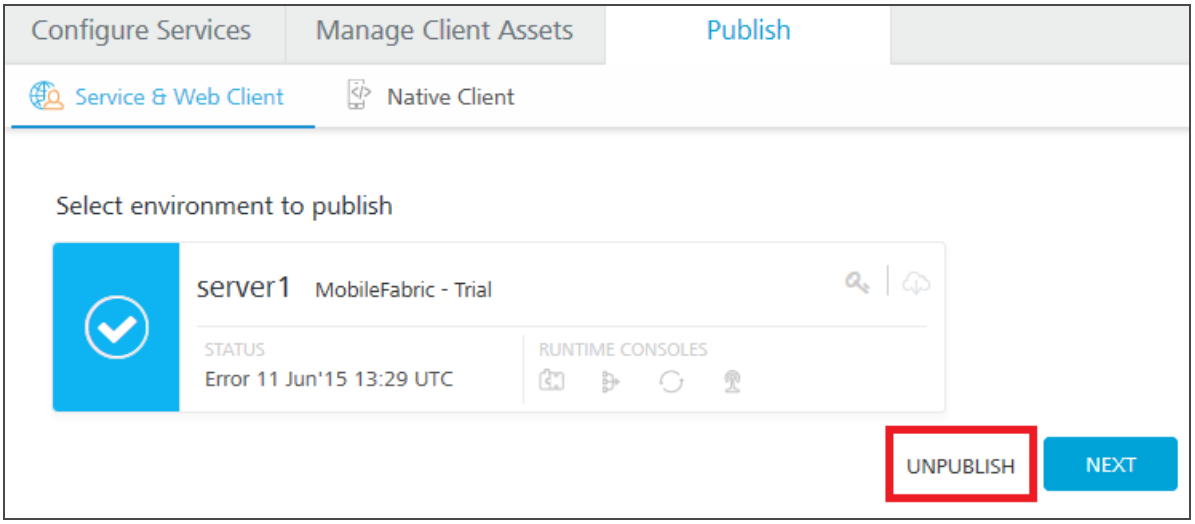
    For example:`https://mobilefabric_console/apps/<appid>/publish/services?sync=true`

    b. Press **Enter** key. The **Publish** page gets reloaded and enabled for synchronous publishing.

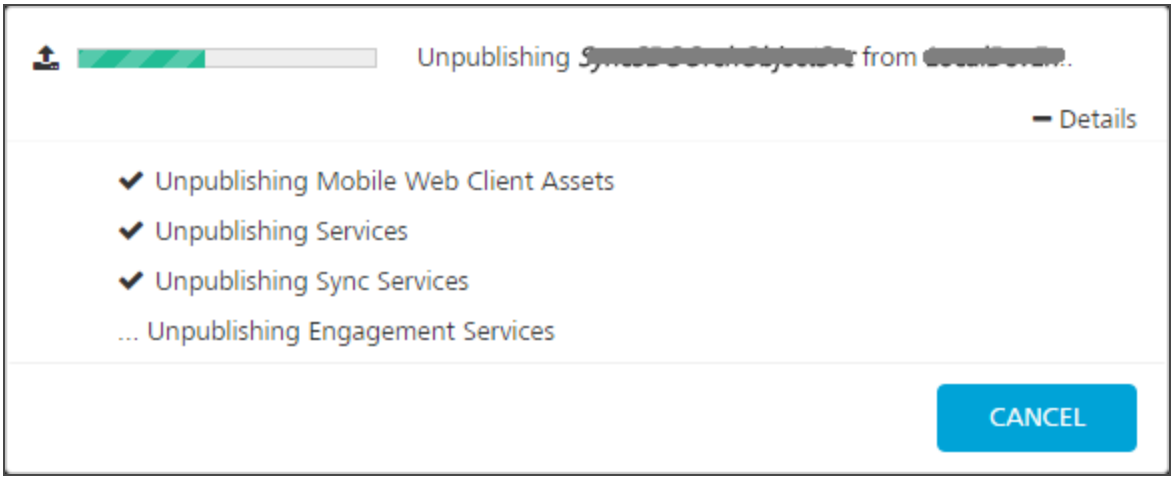
> ***Note:*** The asynchronous publish feature enables you to unblock the UI while an app publish is in progress.

4. Under the **Select environment to publish**, select the environment and then click **UNPUBLISH.**

> ***Note:*** When an environment is published, only then the **UNPUBLISH** button is available.



The process of unpublishing the app begins.



For more details about app publish stages, refer to Publish Life-cycle.

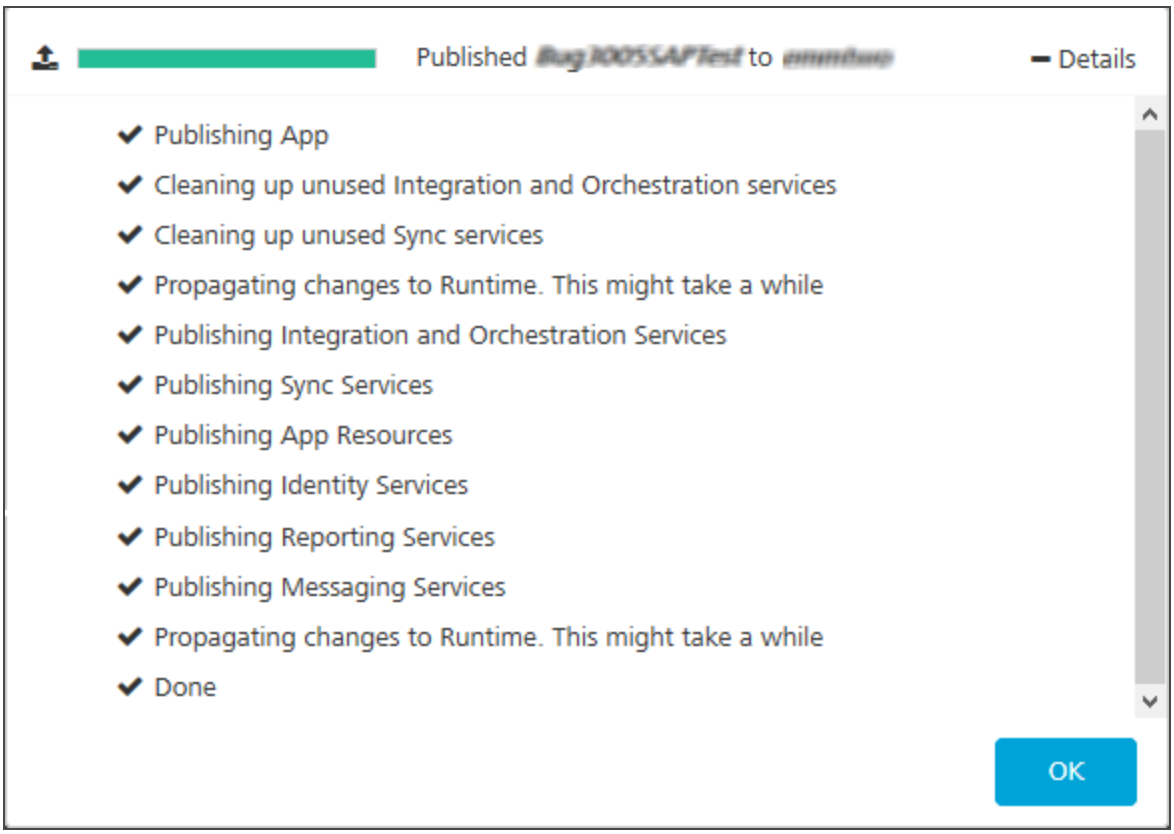
After the app is successfully unpublished, the app status changes to **Not Published** in the **Select environment to publishing** section.

> ***Note:*** You can also unpublish an app via API. For more details, refer to Continuous Integration - Unpublish an app via API

37.3 Publish Life-cycle

In the Publish tab, when you click the **Publish** button, the system starts to publish life-cycle of an app. The publish life-cycle has several app publish stages. During publish app, the system validates each of the stages. Each stage indicates with a tick mark after a successful publish of that stage.

> ***Note:*** If a service is not configured in an app during design time, the system skips that stage without errors while publishing the app.
> However, a tick mark is attached to the stage that indicates publish is successful. While accessing the app in runtime, the system skips non configured services automatically.

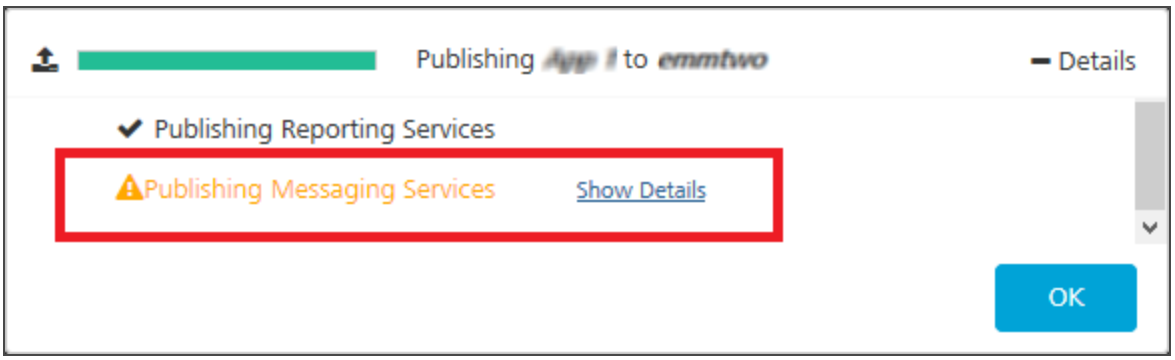


While publishing an app, the system processes a set of stages for the configured services in the app. The stages such as validating services, cleaning unused (Integration Orchestration, and Sync) services, propagating changes to runtime, publishing Sync services, publishing app resources, publishing identity services, publishing reporting services, and publishing messaging services.

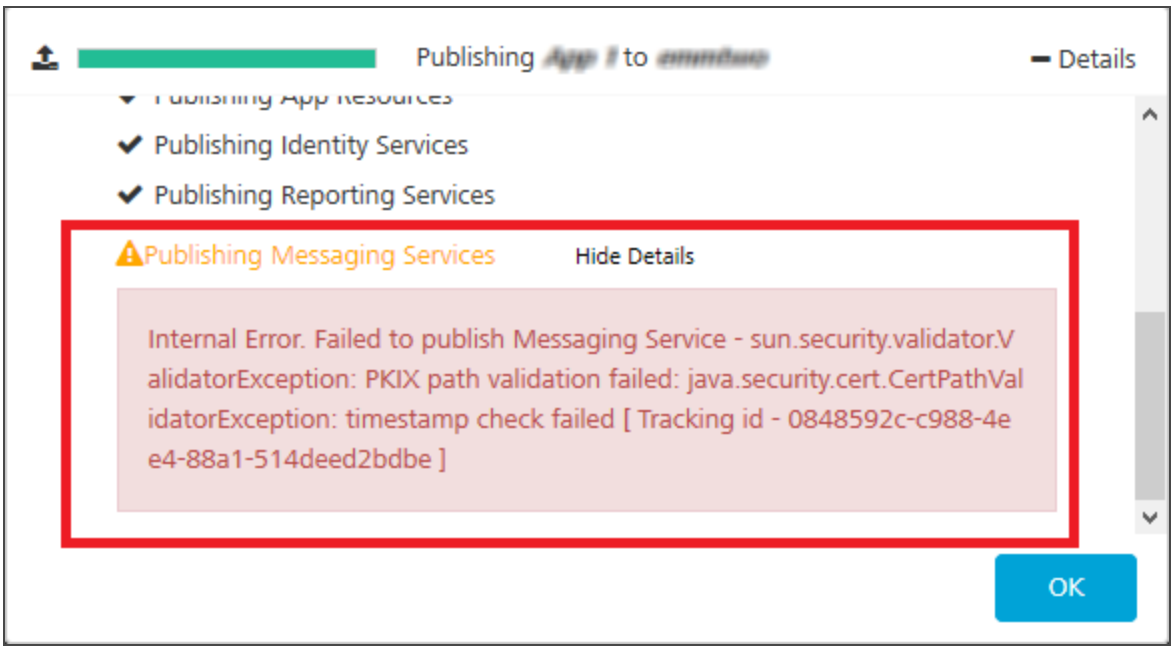
> *Important:* Ensure that you configure and validate services in the app are correct to avoid failures during publish life-cycle.

During publish life-cycle, the system fails to publish a stage due to wrong configurations in the app - such as, if a service is not configured, installed, or registered in the app, or an integration service has a broken reference of an identity service. An exclamation mark next to the failure message indicates that the stage was not successfully published.

For example, if message services have a wrong data configured in the app, the system does not publish this stage and throws the following error:



To view error details, in the Publishing dialog, click the **Show Details** link next that failed services. Click **OK** to close the Publishing dialog.



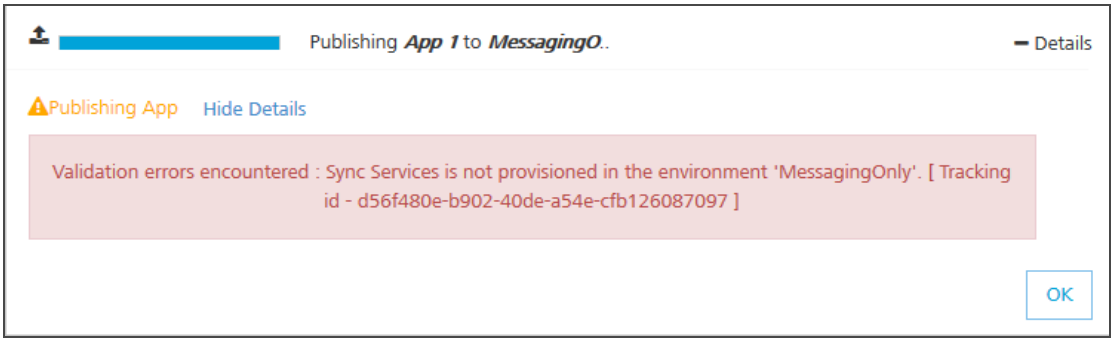
A publish error has the following details:

- **Internal Error message**: indicates service details that caused to the failure of publish.
- **Tracking ID**: indicates a unique error number for that app publish. A tracking ID helps Kony Support team to find the issue within the app and will give solutions to rectify the issue at the earliest.

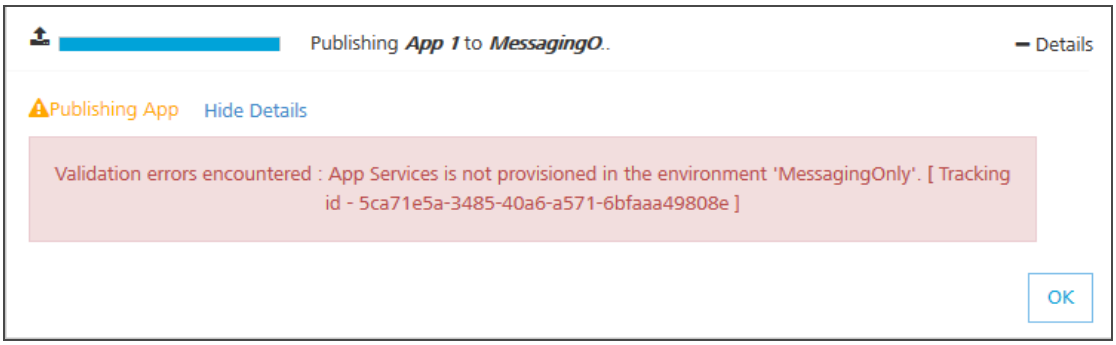
37.4 Publish Failure Error Messages

The following publish failure errors indicate that you have not installed a service or registered a service as an environment.

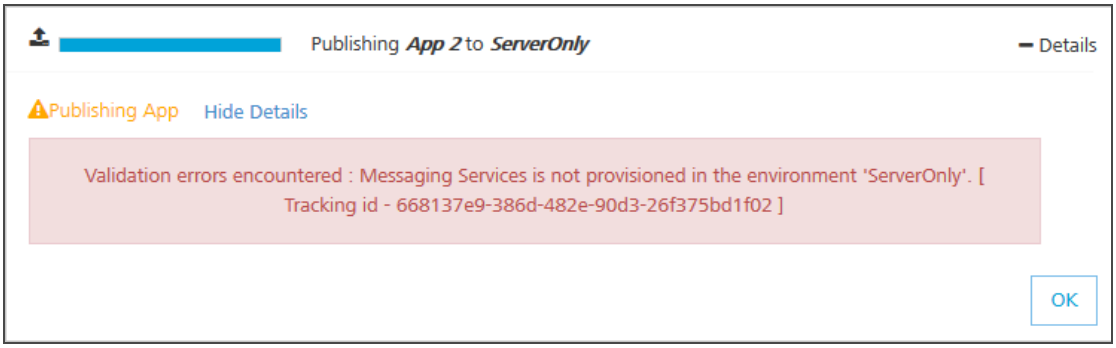
- If the error message says Sync Services is not provisioned in the environment, follow these steps:
  - For on-premises, install Kony Fabric Sync and register to Kony Fabric.
  - For cloud, consult Kony Support team to enable Kony Fabric Sync for that cloud.



- If the error message says App Services is not provisioned in the environment, follow these steps:
  - For on-premises, install Kony Fabric Integration and register to Kony Fabric.
  - For cloud, consult Kony Support team to enable Kony App Services for that cloud.



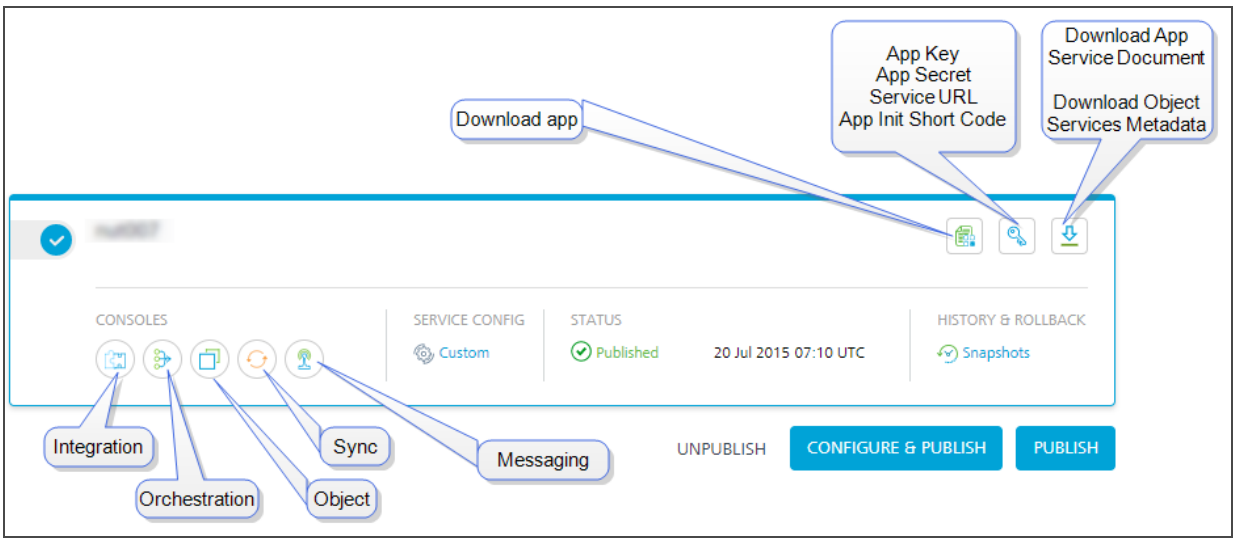
- If the error message says Engagement Services is not provisioned in the environment, follow these steps:
    - For on-premises, install Kony Fabric Engagement and register to Kony Fabric.
    - For cloud, consult Kony Support team to enable Kony Fabric Engagement for that cloud.



37.5 Code Results of a Published App

The following information is available after you publish an app:

- Sample Code
- App Documentation
- App Information (App Key, App Secret, Service URL, and App Init Short Code)
- Download > App Service Document, Object Services Metadata , and Sync Client Code
- Consoles
- Snapshots



37.5.1 Sample Code

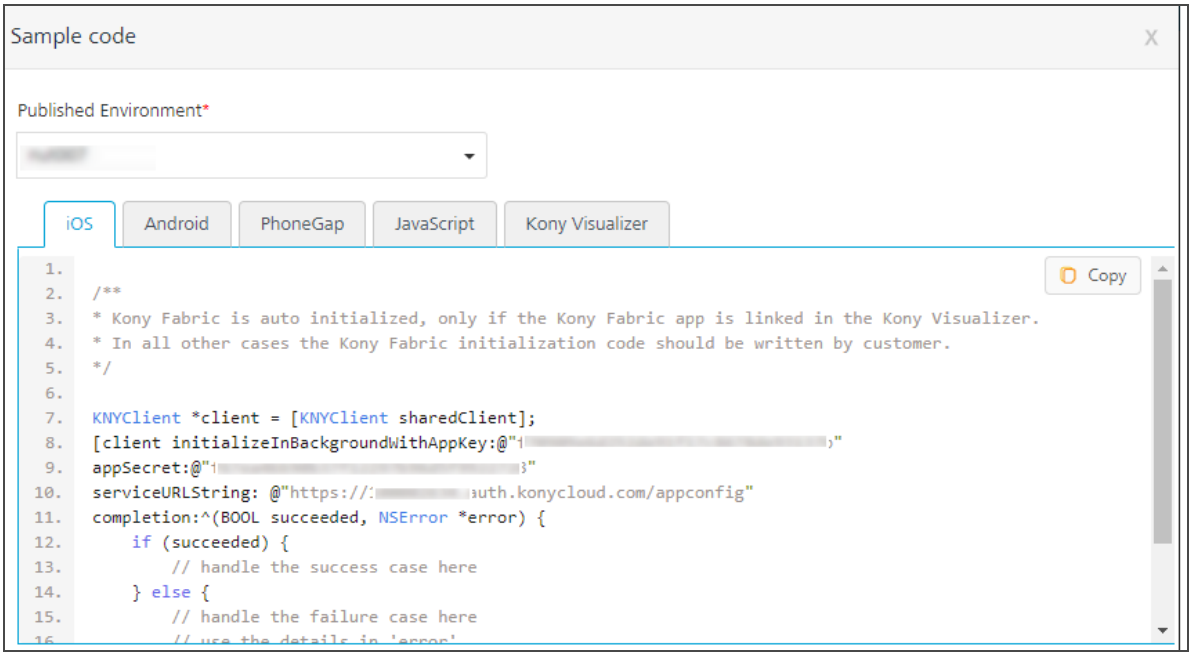
To view the sample code, follow these steps:

1. Click **</>** below the name of the app.

   The **Sample Code** dialog appears.

2. Select an environment from the **Published to environment** list.

3. Select a platform to view the corresponding sample code.



> ***Note:*** You need to add this code in the initialization function of the respective platform SDK.

37.5.2 App Documentation

After you publish an app to an environment, Kony Fabric generates app documentation for your app. You can download app documentation from the **Environments** list in **Publish** page.

a. To view the App documentation of a published app, click the **Download app documentation** button in the **Published Environment** box. The app documentation is downloaded in a zip folder to your local system.

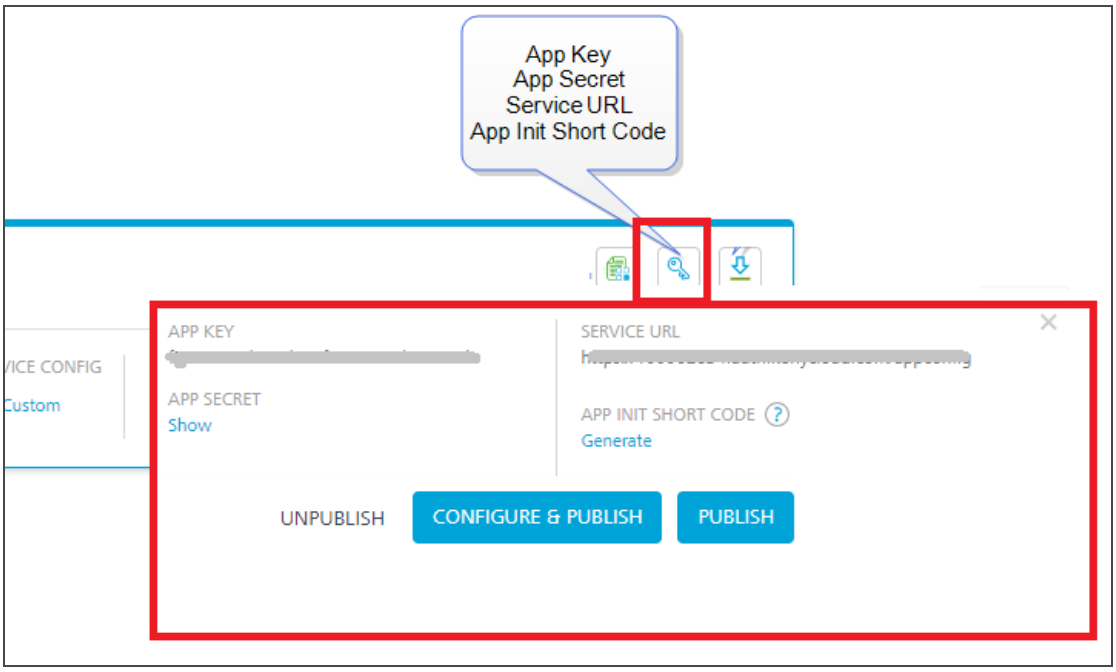
b. Unzip the app documentation folder.
The app documentation .zip comprises multiple artifacts such as `index.html` file, service definition in Open API Initiative (OAI) format, services references in HTML and JSON format.

c. Open the `index.html` in a browser to view the app details.

37.5.3 App Information

To view the App key and App Secret of a published app, click the **App Key** button in the **Published Environment** box.

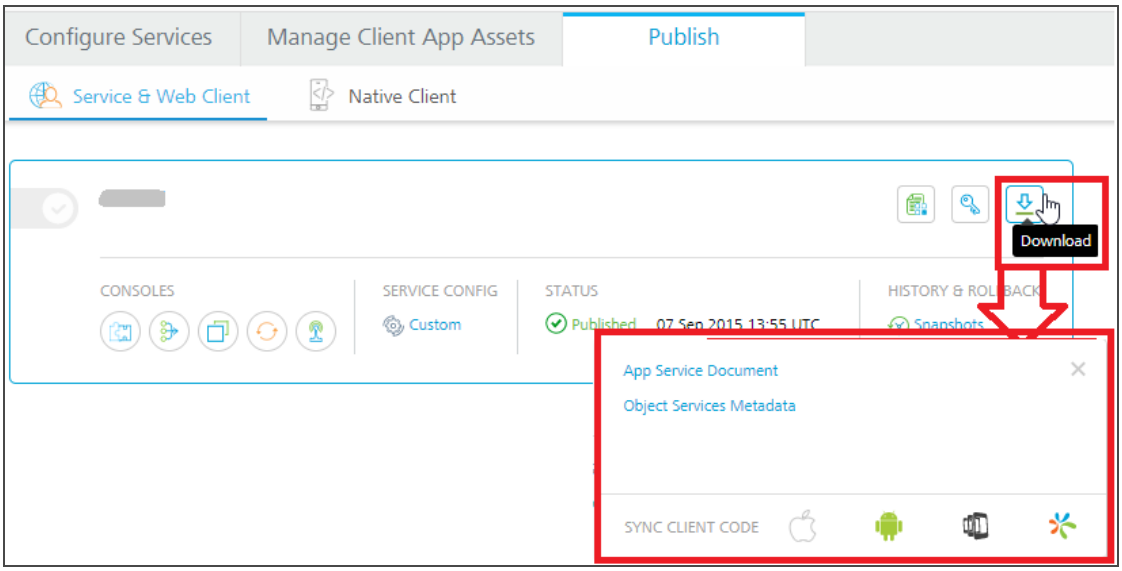
The Primary Key/App Secret, Secondary Key/App Secret, and Service URL are used to initialize a client app to use Kony Fabric services.
For more information, refer to Separate App Key/App Secret for Native and Web Channels.



37.5.4 App Service Document, Object Services Metadata, and Sync Client Code

To view the App Service Document, Object Services Metadata, and Sync Client Code, follow these steps:

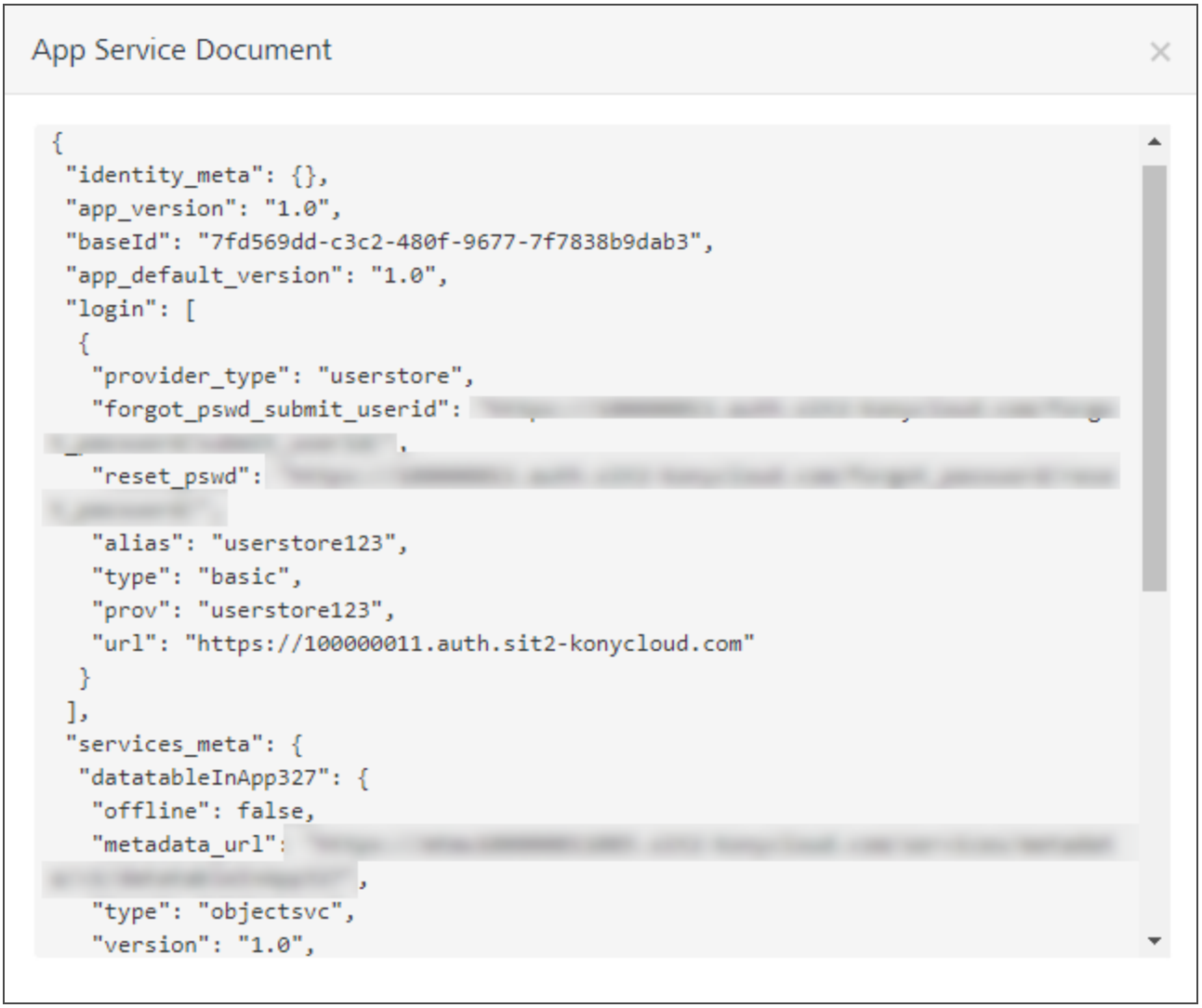
1. Click **Download** button in the **Published Environment** box.



Based on the configuration, the system displays links - for example:

- App Service Document
- Object Services Metadata
- Sync Client Code:
    - Sync client code (PhoneGap)
    - Sync client code (Kony Studio)
    - Sync client code (Android)
    - Sync client code (iOS)

2. Click **App Service Document** to view the code.



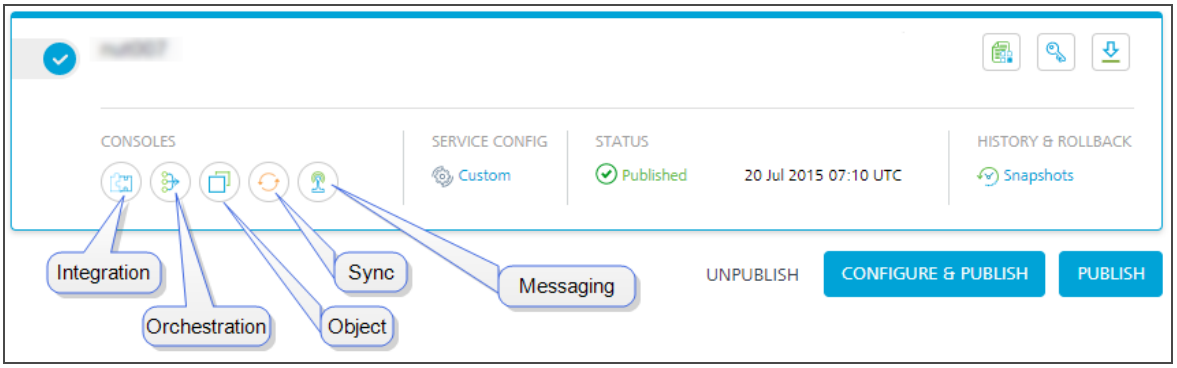
> ***Note:*** If an environment contains an alias hostname, the alias is used in the generated App Service Document. It is also used to make service calls from the client app.

3. Click **Sync client Code (PhoneGap)** to download the Sync client code (PhoneGap) file.

4. Click **Sync client Code (Kony Studio)** to download the Sync client code (Kony Studio) file.

37.5.5 Runtime Console

A published app provides link to the following services consoles:



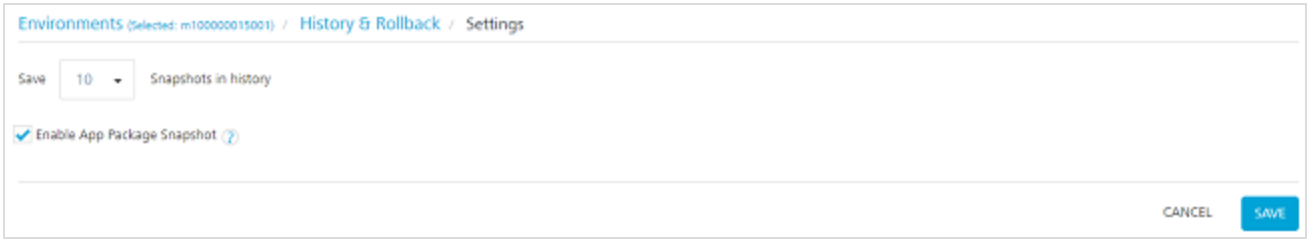
- **Integration Service**: For more information, refer Appendix - App Services.htm.
- **Orchestration Service**: For more information, refer Appendix - App Services.htm.
- **Synchronization Service**: For more information, refer http://docs.kony.com/konylibrary/sync/kony_sync_console_user_guide/Default.htm
- **Engagement Service**: For more information, refer http://docs.kony.com/konylibrary/messaging/kms_console_user_guide_cloud/Default.htm.

37.5.6 Snapshots

Whenever you publish a Kony Fabric app, a snapshot of the published app is created automatically in the Publish life-cycle. It captures the services and other configurations of the app at the time of publish. If the app is republished multiple times with some changes during the development, you can revert back to the previous versions using snapshots. This feature helps you to revert back to a stable state if the current publish results in an error so that the runtime is not effected.

To view the snapshots of the published app, click **Snapshots**. The **History and Rollback** screen is displayed.

By default, you can view the last two snapshots of an app in this screen and the last published app will be marked as **Active**. Click **Settings**to change the number of snapshots you can view.



You can do the following in the **Settings** screen:

- From th **Save** list, select the required number of snapshots you want to save for your Kony Fabric app. You can view maximum 10 snapshots for an app and environment combination.
- Select the **Enable App Package Snapshot** check box if you want to download the app package of a snapshot. An app package contains all the services of a Kony Fabric app. You can import this package to the same or a different Kony Fabric Console.
  If you have reconfigured an app before publish, then you can also download the details of the cumulative reconfigured parameters in a separate file by selecting this check box.
- Click **Save**. The settings will be saved accordingly.

You can perform the following activities from the contextual menu available for each snapshot in the **History and Rollback** screen:

- **Republish**: When you click **Republish**, the app will be restored as per the configurations in the selected snapshot.
- **Download**: It downloads the Kony Fabric App package of the selected snapshot. You can import this package to the same or a different Kony Fabric Console.

- **Download Reconfiguration**: You can download the JSON file that contains the cumulative reconfiguration data, if you have reconfigured and published an app using **Configure and Publish**. The data of all the reconfigured services are stored in the snapshot.
- **Delete**: It removes the selected snapshot from the list.

37.6 Separate App Key/App Secret for Native and Web Channels

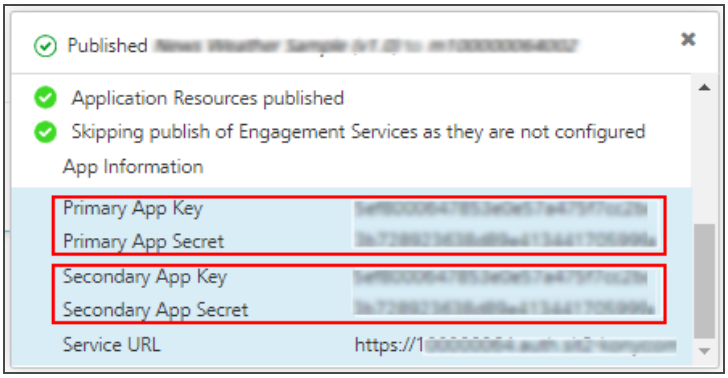
Kony Fabric from V8 SP4 FP4 HF4 supports generating a separate set of app key/ app secret for the Web and native channels. It also supports the deprecating app key/app secret to allow developers to rotate the keys and keep the app more secure.

Why Do You Need Separate App Key/App Secret: To ensure you follow the proper security guidelines and best practices for authentication your app, Kony recommends using a **separate** set of an app key/app secret for building apps for Web and Native channels.

What are Separate App Key/App Secret: Kony Fabric generates the separate set of app key/app secret for Web and Native channels for an app by default. After you publish an app, you can view the Primary Key/Secret and Secondary Key/Secret in the **Published** status dialog > **App Information**. You can modify the app key/app secret.

Kony Visualizer uses the following app key/app secret to build binaries:

- **Primary App Key and Primary App Secret** that is used by the Web channel.
- **Secondary App Key and Secondary App Secret** that is used by the Native channel.



Important: If your Kony Fabric version is V8 SP4 FP4 HF4 or later and Visualizer version is V8 SP4 FP45 or earlier, the primary key is used for Web and Native channels.

37.6.1 Prerequisites

To use separate app key or app secret, ensure you have access to the following:

- Kony Fabric version should be V8 SP4 FP4 HF4 or later.
- Visualizer version should be V8 SP4 FP45 or later.

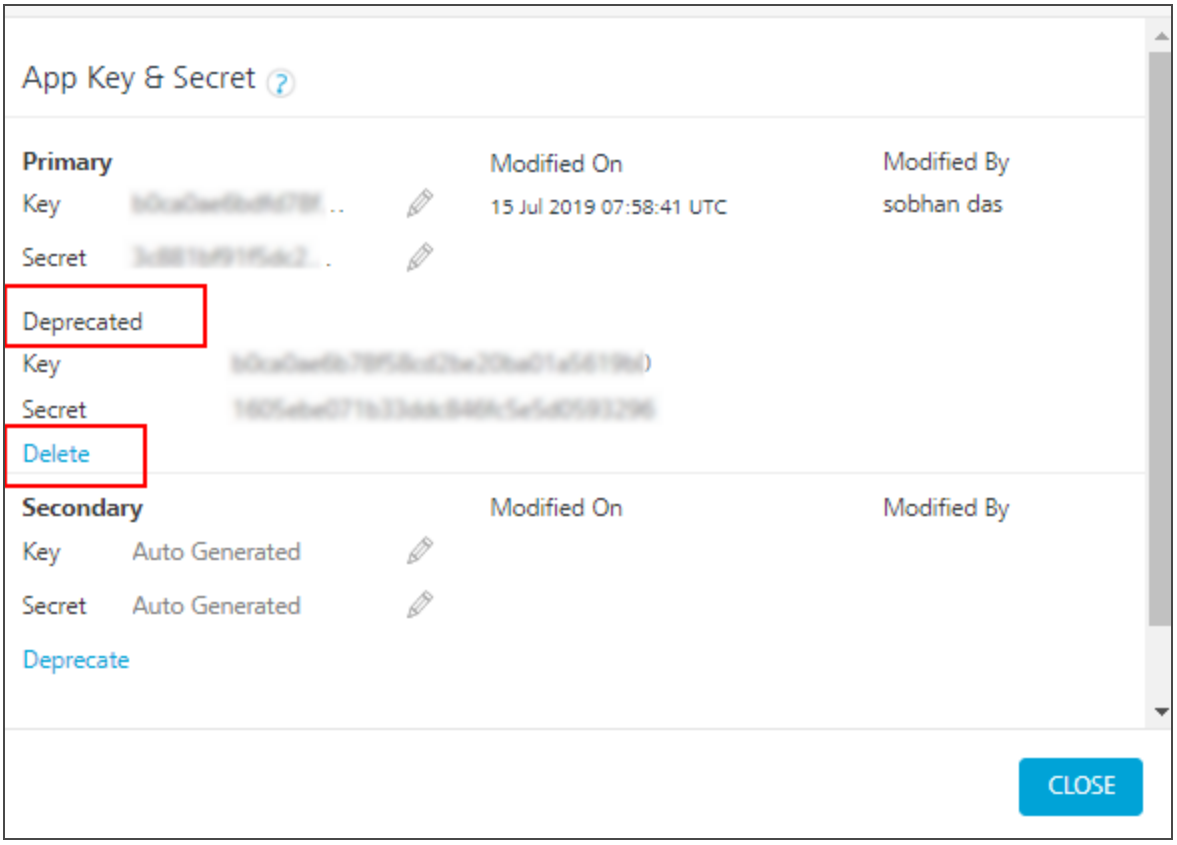
37.6.1.1 Deprecating an App Key/App Secret

Key rotation is supported to help make the app more secure and allow developers to deprecate the active set of app key/ app secret for primary or secondary key-set. While deprecating the active app key/app secret, Kony Fabric ensures that you can either auto-generate a new app key/app secret or manually define your own key secret.

To deprecate an active app key/app secret, follow these steps:

1. Go to Kony Fabric and open the app published to an environment.
2. In the Publish > Service & Web Client page, click **App Key**. The **App Key & Secret** dialog appears and displays the Primary Key/Secret and Secondary Key/Secret details of the app.
3. Click **Deprecate** for Primary or Secondary.
4. The message box appears for your confirmation. Click **YES**. Now the existing app key/app secret are marked as **Deprecated**.

> ***Note:*** You cannot deprecate the active app key/app secret until you delete the existing deprecated app key/app secret.



5. Click **Save** to generate the new app key/app secret and to save these details. Otherwise, you can click the **Edit** button next to the app key/secret to enter the details in the text field. Click the **Tick** button next to the text field to save the details. You have created a new app key/secret.

- In this case, if you click **CLOSE** without saving the new app key/app secret details, your changes are canceled.
- After the new app key/app secret details are saved, the deprecated app key/secret details are marked as **Deprecated**. The **Delete** button appears for the app key/app secret that you have deprecated.

> ***Important:*** Once you delete the deprecated app key/app secret, users cannot access the app using the previous app key/app secret.

> ***Note:*** If you inadvertently deprecate an app key and secret and wishes to reuse the previously used app key/secret, you can do so by manually copy and pasting it to the active app key/secret and saving it.

6. To delete the deprecated app key/app secret permanently, click **Delete**.
7. Click **CLOSE**.

   The new app key/app secret is updated to the app in the run-time server. You do not need to republish the app to the server. Now the services from the app are accessible with the new app key/app secret.

   If you want the new app key/app secret to being part of your client app, you can build the app binary with the new app key/app secret and publish it to your environment. Now users can access the app with the new app key/app secret.

37.7 Reconfiguration at Publish

During app publishing, Kony Fabric allows you to reconfigure the default values of app services (identity and integration) specific to an environment and publish the app to an environment. When you configure required services for an app, the system stores the default app configurations in your current

workspace. A Kony Fabric application stored in the workspace comprises shared and non-shared services. All these services have a certain configuration stored in the workspace. You can reconfigure these services while publishing the app to an environment. The default values of app services will not be changed while reconfiguring services.

> *Important:* For more details about How to Publish an App, refer to **Publish**.

The following services types can be reconfigured:

- App Reconfiguration
- Service Reconfiguration
    - Integration Service Reconfiguration
    - Object Service Configuration
    - Identity Service Reconfiguration

37.7.1 App Reconfiguration

During app publishing, Kony Fabric allows you to reconfigure the default values of app services specific to an environment and publish the app to another environment.

App Key and App Secret are used by a Mobile app to identify and communicate with a published instance of a Kony Fabric app. You can configure specific values or let the system auto-generate those.

> *Important:* For more details about How to Publish an App, refer to **Publish**.

37.7.1.1 Use Cases

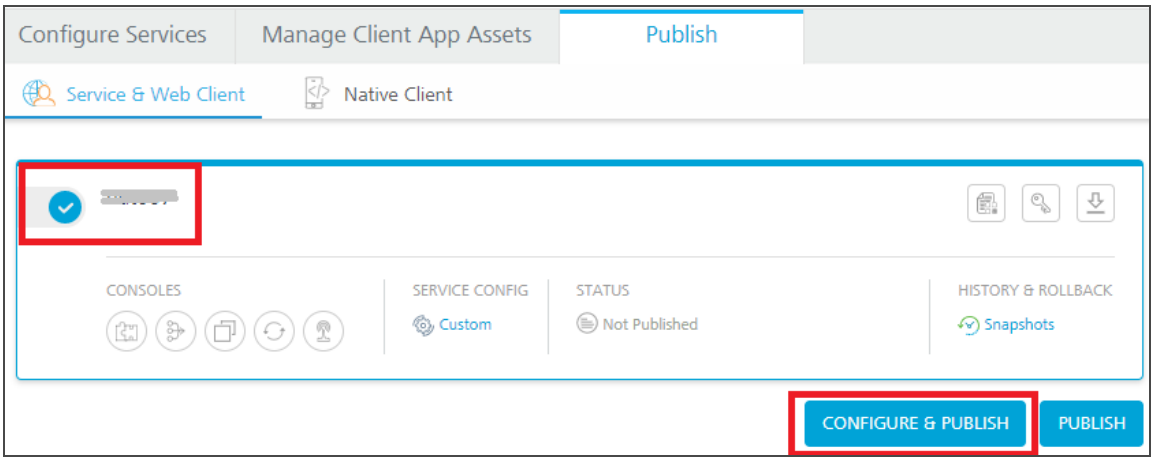
You can use service reconfiguration based on the following scenarios:

A developer can use an app configuration in the current workspace and publish the reconfigured app to another environment. A developer can reconfigure only a few entities (such as base URL, User ID, and password).

- For example, currently an endpoint URL of an integration service of the app is configured as `http://sample.test.com`, which is a test environment. But after an app is published in a production environment, a developer wants the integration service to communicate to a production endpoint - for example, `https://sample.com`. In such cases, a developer reconfigures the endpoint URL from http://sample.test.com to https://sample.com, and publishes the app to a production environment.

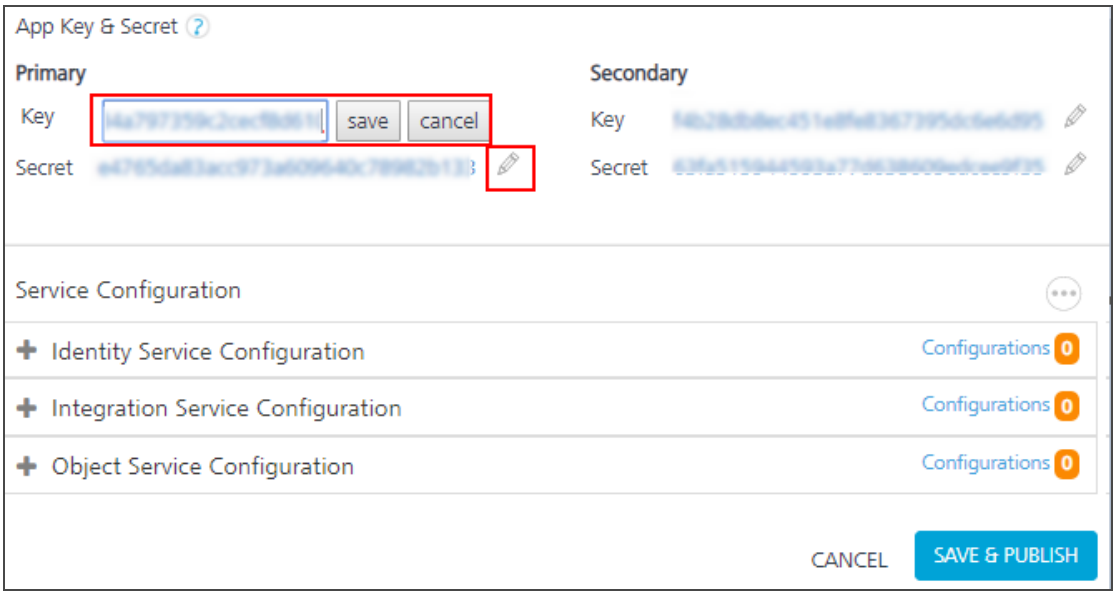
To display the app reconfiguration page while publishing an app, follow these steps:

1. After you complete configuring services in an app, click the **Publish** tab.
2. In the **Publish** page, select an environment. The **CONFIGURE & PUBLISH** and **PUBLISH** buttons are active only after you select an environment.
3. Click **CONFIGURE & PUBLISH** to display the app reconfiguration page.



The **App Configuration** page appears with the following properties:

App Details	Properties
Primary App Key	• Primary App Key/Secret for Web Channel • Secondary App Key/Secret for Native Channel ***Note:*** For more information on separate app key/secret, refer to Separate Appkey/Secret for Native and Web Channels Note that App Key needs to be unique. Additionally App Key and App Secret should only contain alpha numeric characters or '-' and must contain at least 5 and a max of 60 characters.
Primary Secret	
Secondary App Key	
Secondary Secret	



4. To change any value, click the **Edit** button.
   The system displays the original value in the text box, and the **save** and **cancel** buttons next to the text box. The text field is active.
5. Change the value in the text field.
6. Click **save**.

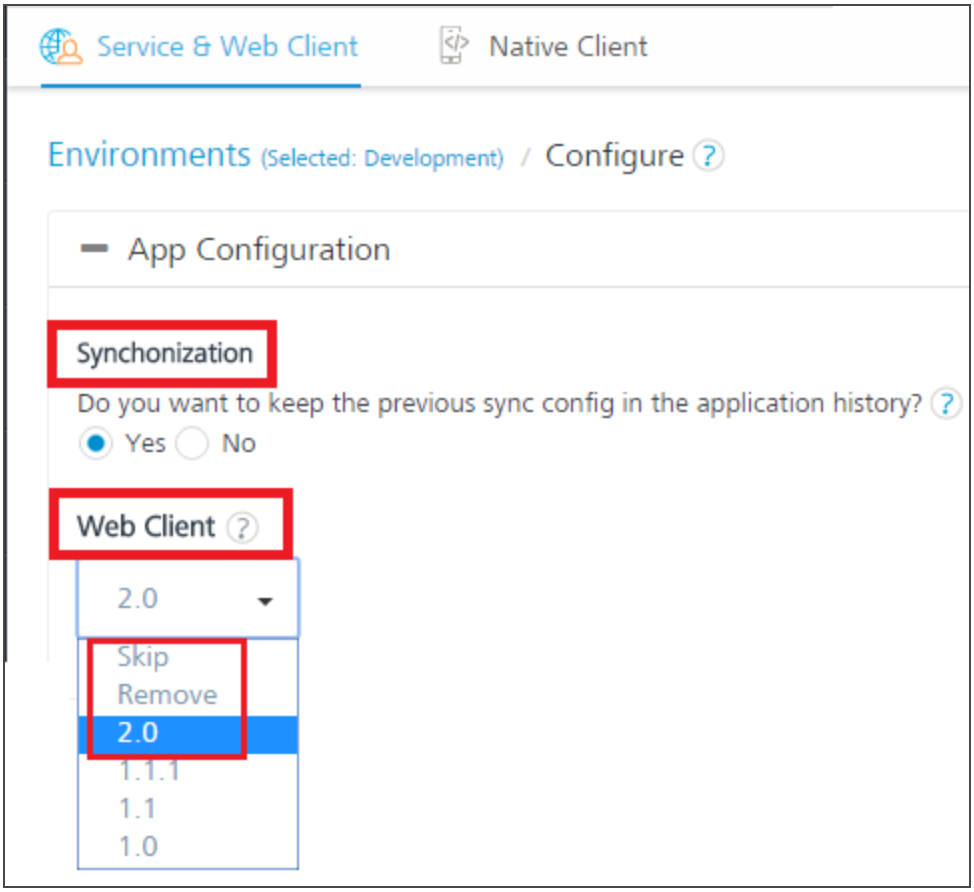
7. If you have configured sync services in your app and also uploaded a binary file for Web client, do the following:

    a. Under **Synchronization**, choose **Yes** to keep the previous sync configuration in the application history. By default, the synchronization is set to **No**.

    > ***Note:*** You are publishing an update to your sync configuration. If you keep the previous sync config in the application history, any devices that already have a local copy of the data can attempt to sync only the schema changes and avoid a full sync. If you do not keep the previous sync config in application history, all devices will be required to fully rebuild their local copy of the data using a full sync. In either case, the client app must be rebuilt with the new sync configuration and updated to all devices.

    b. Under the **Web Client**, Select the required version from the drop-down list. When you select a version and publish the app, the system overrides the existing version of the `.war` file with the selected new version in the Server. If there is no published version, the system adds the new Web client in the server.

    - **Skip**: If you select Skip, Web Client Assets will not be included in this Publish. If you select a version of Web Client, the last published Web Client will be deleted and the said Version of Web Client will be published to Server.
    - **Remove**: If you select Remove, the Web Client is removed if present in the Server.



8. Click **SAVE & PUBLISH** to start the publishing. The process of publishing the app begins.

37.7.2 Service Reconfiguration

In an Enterprise, as a best practice, the development enterprise backend systems are maintained separately from the production enterprise backend systems. Fabric supports the ability to follow the best practice by providing DEV, QA and PROD environments. DEV and QA environments are typically configured to communicate with development enterprise backend and PROD is configured to communicate with Production enterprise backend.

Service Reconfiguration lets you use the same service against different environments by providing the ability to change the connection parameters to a backend while publishing an app to an environment.

37.7.2.1 Use Case

You can use service reconfiguration based on the following scenarios:

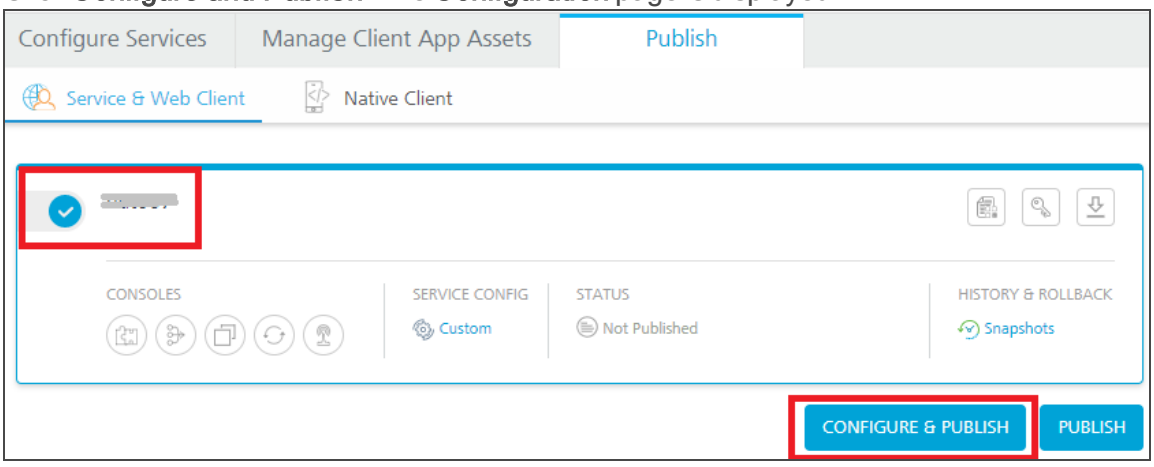
A developer can use an app configuration in the current workspace and publish the reconfigured app to another environment. A developer can reconfigure only a few entities (such as base URL, User ID, and password).

For example, currently, an endpoint URL of an integration service of the app is configured as `https://qa.sample.com`, which is a QA environment. But after an app is published in a PROD environment, a developer wants the integration service to communicate to a production endpoint - for example, `https://sample.com`. In such cases, a developer can reconfigure the endpoint URL from http://sample.test.com to https://sample.com, and publish the app to a PROD environment.

37.7.2.2 Reconfiguring a Service

To reconfigure a service from the **Publish** tab, do the following:

1. From the **Services and Web Client** tab, select the required environment.
2. Click **Configure and Publish**. The **Configuration** page is displayed.



3. Select the required service type from the list. You can also configure the app related information from here. The service types available in the list are as follows:

- Identity Services
- Integration Services
- Object Service

4. The app displays the selected service details with default connection parameters and configurable connection parameters.

The list of the configurable connection parameters vary for each service type. Following is the list of configurable connection parameters with respect to each service type and adapter.

Identity Service

Identity Service Provider	Properties
Okta Login	• Domain name ***Note:*** Refresh the other URLs on the domain name to point to the new reconfigured URL. • Client ID • Client Secret
OAuth Services (Out of the box OAuth connectors like Microsoft, Linkedin)	• Authorize Endpoint • Token Endpoint • Profile Endpoint • Client ID • Client Secret
Custom Identity Service	Custom Identity Service Endpoint

Identity Service Provider	Properties
SAP Service	• Gateway Address • Port • Scheme
Salesforce	OAuth 2.0 and Username/Password: • Salesforce ClientID • Salesforce ClientSecret • Salesforce URL
Microsoft Active Directory	SAML: • Metadata URL (in case Metadata Mode = URL) LDAP/LDAPS: • Domain Name • LDAP URL • Root Domain Azure Active Directory (SAML): • Metadata URL (in case Metadata Mode = URL)
SAML	Metadata URL (in case Metadata Mode = URL)
Open LDAP	• Domain Name • LDAP URL • Root Domain • Bind Username

Integration Service

Adapter type	Properties
XML	• Base URL - you can reconfigure the base URL. • Web Authentication - you can configure the authentication modes. • ignore proxy - you can reconfigure the proxy (for on-premises only).
JSON	• Base URL - you can reconfigure the base URL. • Web Authentication - you can configure the authentication modes. • ignore proxy - you can reconfigure the proxy (for on-premises only).
SOAP	• Base URL - you can reconfigure the base URL. • Web Authentication - you can configure the authentication modes. • ignore proxy - you can reconfigure the proxy (for on-premises only).
SAP	• If you choose Select authentication service as **Use Existing Identity Provider**, you cannot reconfigure any entities. • If you choose Select authentication service as **Specify Login Endpoint**, you can reconfigure the entities such as gateway address, port, and header prefix.

Adapter type	Properties
Salesforce	• If you choose Salesforce Authentication Type as **Use Existing Identity Provider**, you cannot reconfigure any entities. • If you choose Salesforce Authentication Type as **Specify Login Endpoint**, you can reconfigure the entities such as client ID, client secret, username, and password.
Java	None
JavaScript	None
APIProxy	None
Relational Database	• Connection Parameters • Schema Mapping JSON
MongoDB	• Connection Parameters
RAML	• Connection Parameters
OpenAPI (Swagger)	• Connection Parameters • Host URL • Base Path
Salesforce	• Connection Parameters
IBM MQ	• Connection Parameters

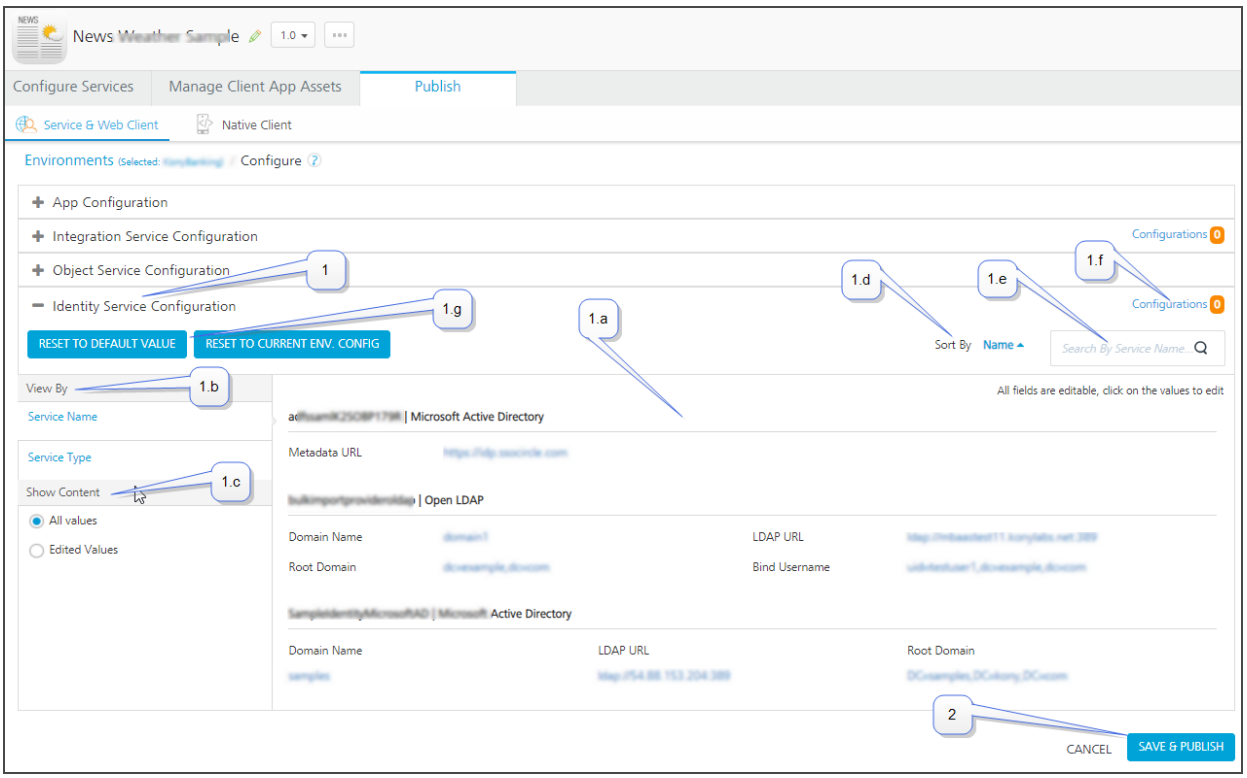
Note: Throttling details for all integration services are displayed under **Publish** > **Configure & Publish** > **Integration Service Configuration**.

Object Service

Adapter type	Properties
SAP	• If you choose Select authentication service as **Use Existing Identity Provider**, you cannot reconfigure any entities. • If you choose Select authentication service as **Specify Login Endpoint**, you can reconfigure the entities such as gateway address, port, and header prefix.
Salesforce	• If you choose Salesforce Authentication Type as **Use Existing Identity Provider**, you cannot reconfigure any entities. • If you choose Salesforce Authentication Type as **Specify Login Endpoint**, you can reconfigure the entities such as client ID, client secret, username, and password.
Relational Database	• Connection Parameters • Schema Mapping JSON
MongoDB	• Connection Parameters
RAML	• Connection Parameters
Integration/Orchestration	• None
Storage	• None

Note: Throttling details for all object services are displayed under **Publish** > **Configure & Publish** > **Object Service Configuration**.

The service details page displays the following fields:



Number	Section	Description
1	**Identity Service Configuration**	Displays the Identity Services configuration details for app reconfiguration.

Number	Section	Description
1. a	**Work area**	Displays the configurable properties of all the services such as Client ID, Client Secret and Domain Name. You can edit the values of the properties and click Save. An undo icon appears next to the value field until you publish the app. You can reset the new value to default value by selecting one of the options in the Reset to Default Values drop-down list. ***Note:*** Based on a combination of selected conditions in the **View By** and **Show Content** sections, the system displays services, entities for all services, and edited values. You can select the following combinations in the **View By** and **Show Content** section: - Service Name and All Values - Service Name and Edited Values - Service Type and All Values - Service Type and Edited Values

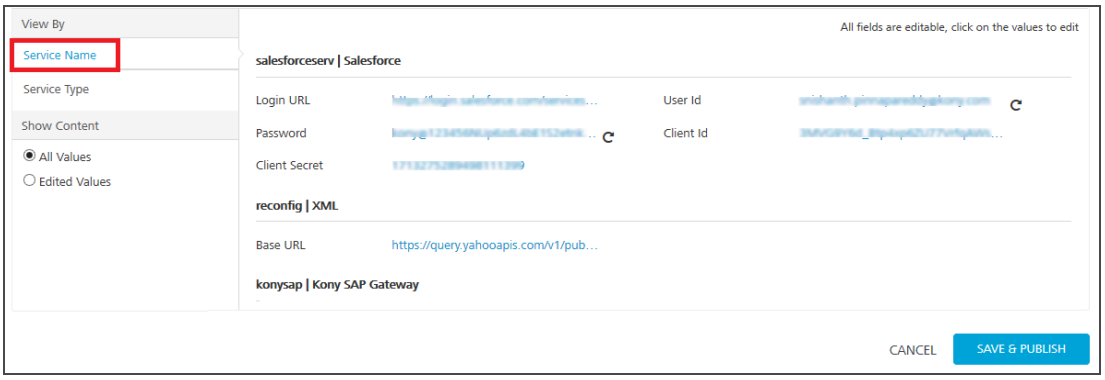
Number	Section	Description
1. b	**View By**	Displays services based on conditions such as service name or service type. • **Service Name**: Displays services and their entities. • **Service Type**: Displays the headers of service types. To view content in a service type, click the **Plus (+)** icon.
1. c	**Show Content**	Displays content in services based on conditions such as all values or edited values. • **All Values**: Displays services and their entities. • **Edited Values**: Displays services and their edited entities.
1. d	**Sort By**	Sorts the services in the work area in A to Z or Z or A order.
1. e	**Search by Service Name**	Filters the services based on service name.
1. f	**Configurations**	Compares the reconfigured services.
1. g	**Reset to Default Values** • **Reset to Default Value** • **Reset to Current Env. Config**	Resets the values of reconfigured services.

Number	Section	Description
2	**SAVE & PUBLISH**	Saves and publishes the reconfigured app to an environment.

You can do the following from the service details page to configure the connection parameters:

- In **View By**, you can select the required option to change the order in which you view the services. The options available are as follows:
  - **Service Name**: Click **Service Name** if you want to view the services in an alphabetical order based on the service names. The system displays all the details of each service in this view.

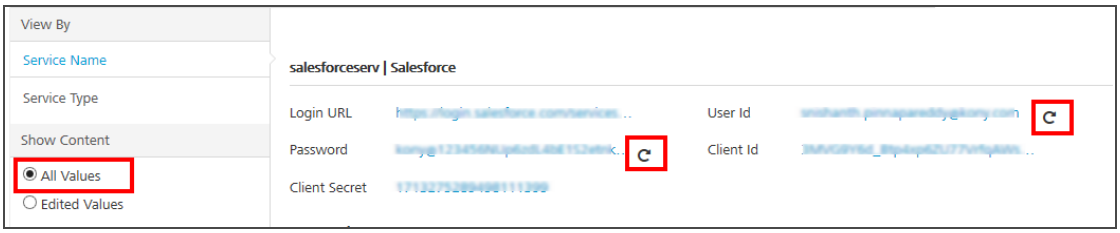
    > ***Note:*** By default, the list view is set to **View By as Service Name** and **Show Content as All Values**.

  - **Service Type**: Click **Service Type** if you want to group the services based on the service

type. To view details in each service type, click the **Plus** icon and expand the service.

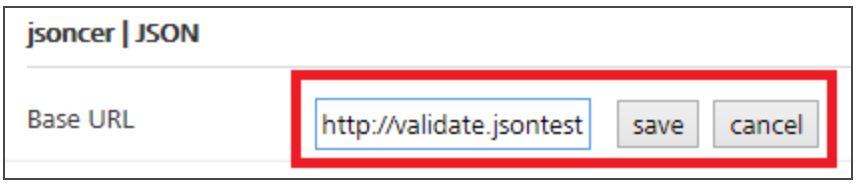
- In **Show Content**, you can select the required option to change the view of the content within the service based on their values. The options available are as follows:
  - **All Values**: Click **All Values** if you want to view all values including reconfigured values for the selected filter in the **View By** section.
    For example, if you change User ID and Password for Salesforce service, shown below.



  - **Edited Values**: Click **Edited Values** if you want to view only the reconfigured values for the selected filter in the **View By** section.
    For example, if you change User ID and Password for Salesforce service, shown below.

- In the work area, follow these steps to reconfigure values in the services:
  a. To change any value, click the value. For example, to change the base URL, click it.

The system displays the value in the text box, and the **save** and **cancel** buttons next to the text box.



  b. Edit the value in the text box . For example, change `com` to `net`.

> ***Important:*** When you reconfigure a value, click the **save** button to save the changes. If you click anywhere else on the page, the changes will be lost, and you will get only the previous data.

  c. Click **save**. The system adds an **undo** button next to each reconfigured values.

- Click **SAVE & PUBLISH** to start the publishing.

You can also do the following tasks from the service details page:

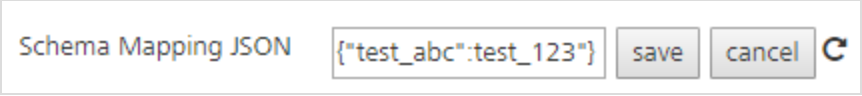
- Compare Reconfigured Integration Services
- Reset Values for Reconfigured Integration Services

37.7.2.3 Reconfigure the Schema Name in Relational Database Connectors

In **Integration Service Configuration**, you can reconfigure the schema name in Relational Database Connectors using JSON. To reconfigure the schema name, follow these steps:

- Under **Schema Mapping JSON**, click against the service to enter the schema name.

  For example, To change the schema name from dev to QA, enter the JSON value and click **Save**.

  - `test_abc` - Schema on which operations are selected.
  - `test_123` - Schema on which run time operations must be performed.

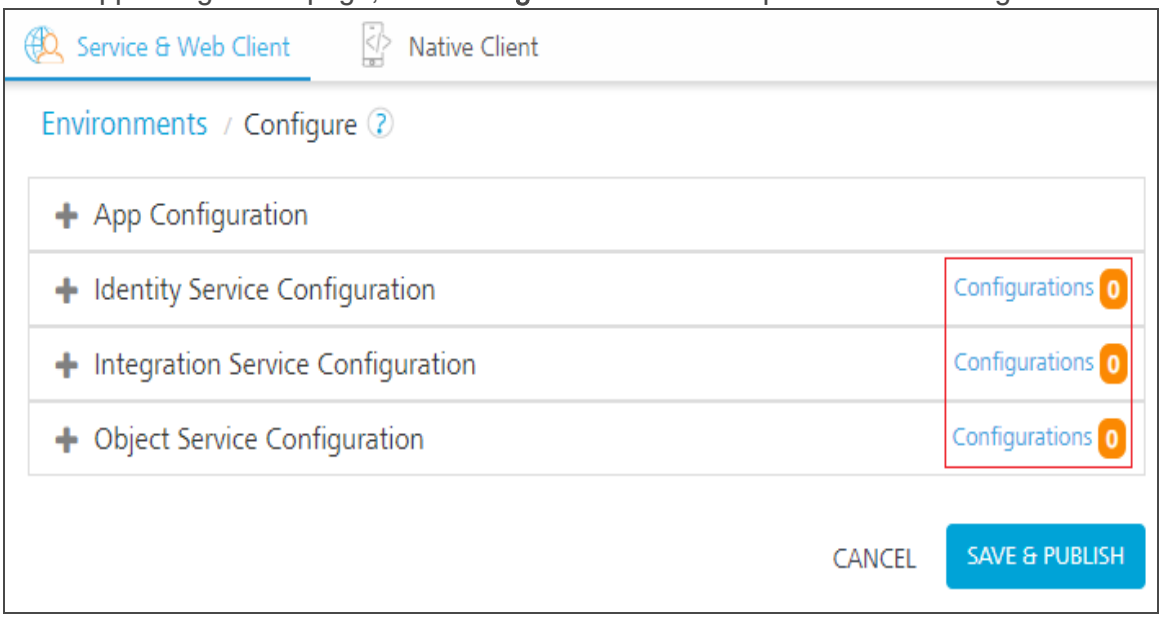
  The system adds an **undo** button next to each reconfigured value.

- Click **Save and Publish** to start the publishing.

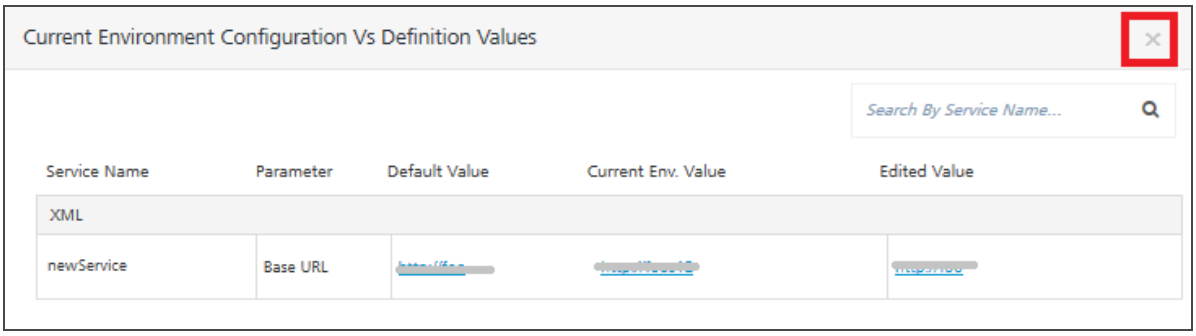
37.7.2.4 Compare Reconfigured Services

In the service details page, after you reconfigure values of services, you can compare the details of all services. You cannot edit any fields in the compare dialog.

1. In the App configuration page, click **Configurations** on the required service configuration row.



The system displays the **Current Environment Configuration Vs Definition Values** dialog box.



- **Service Name**: Displays the service name.
- **Parameter**: Displays the field name that is reconfigurable.
- **Default Value**: Displays the value configured in the design time.
- **Current Env. Value**:Displays reconfigured value in the current environment.

- **Edited Value**: Displays the edited value for the selected environment.

> ***Important:*** In the **Current Environment Configuration Vs Definition Values** dialog box, the values in the **Current Env. Value** and **Edited Value** fields are updated to an environment only when you click **SAVE & PUBLISH**.

2. Click the **Close** button at the top-right corner to exit the dialog.

37.7.2.5 Reset Values for Reconfigured Services

In the service details page, after you reconfigure values of services, you can reset the values to the default configuration or to the current environment configuration.

> ***Important:*** When you modify a field and save the service, the system overrides old values with the new values in the current environment clipboard.

To reset to default values, follow these steps:

- **Reset to Default Value**: After you reconfigure a value in the work area, and when you choose **Reset to Default Value**, the system resets the reconfigured values to the default values that are configured in the current workspace - for example, Developer environment.
- **Reset to Current Environment Config**: Contains last saved values. After you reconfigure a value in the work area, when you choose **Reset to Current Environment Config**, the system resets all values to the current environment (last saved values).

37.8 Managing Service Profiles

The Service Configuration Profile Import and Export capability enables you to exchange service configuration parameters of an app between two environments.
Every service profile includes configuration parameters of all the services that are linked to an app. The details of the service profile are available in the form of a JSON file. You can export a service profile from one environment, edit the required service parameters in the JSON file, and then import the updated service profile to another environment.

Service profiles can be versioned in a Git repository and these versions can be used for app upgrades. This provides full portability of a service profile to deploy the profile to new clouds.

Service profiles support is available for Kony Fabric Console and MFCLI. The Export and Import features are in the **Publish > Service & Web Client > CONFIGURE and PUBLISH** page.

37.8.1 Use Cases

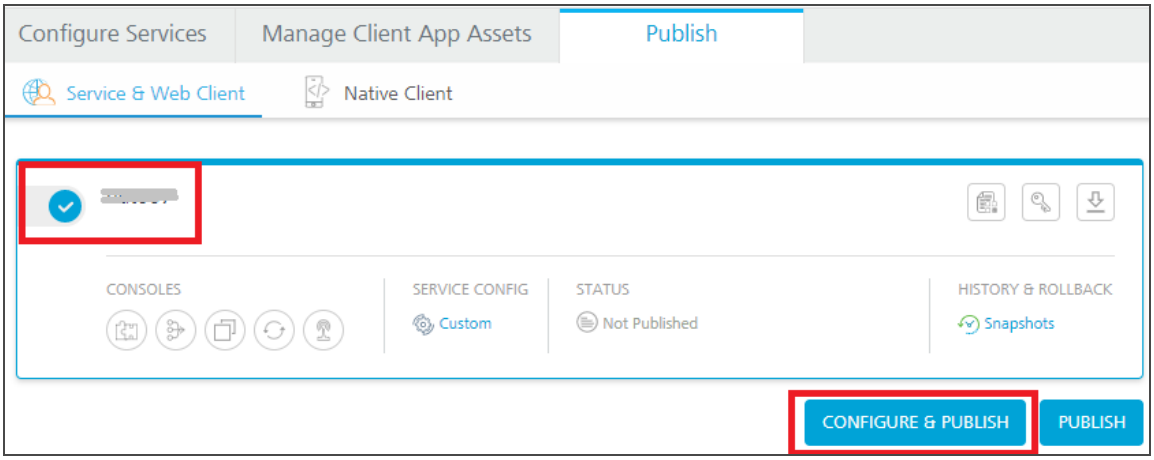
You can use service reconfiguration based on the following scenario:

- The use case: Currently, developers have to recreate endpoint configurations as they move from one environment to another environment. For example, you can set up a service profile for a Development environment. When you want to move into QA, you need to reconfigure all the service endpoints. Instead, Kony provides developers the ability to export service configuration profile so that the service profile can be imported into a new environment.

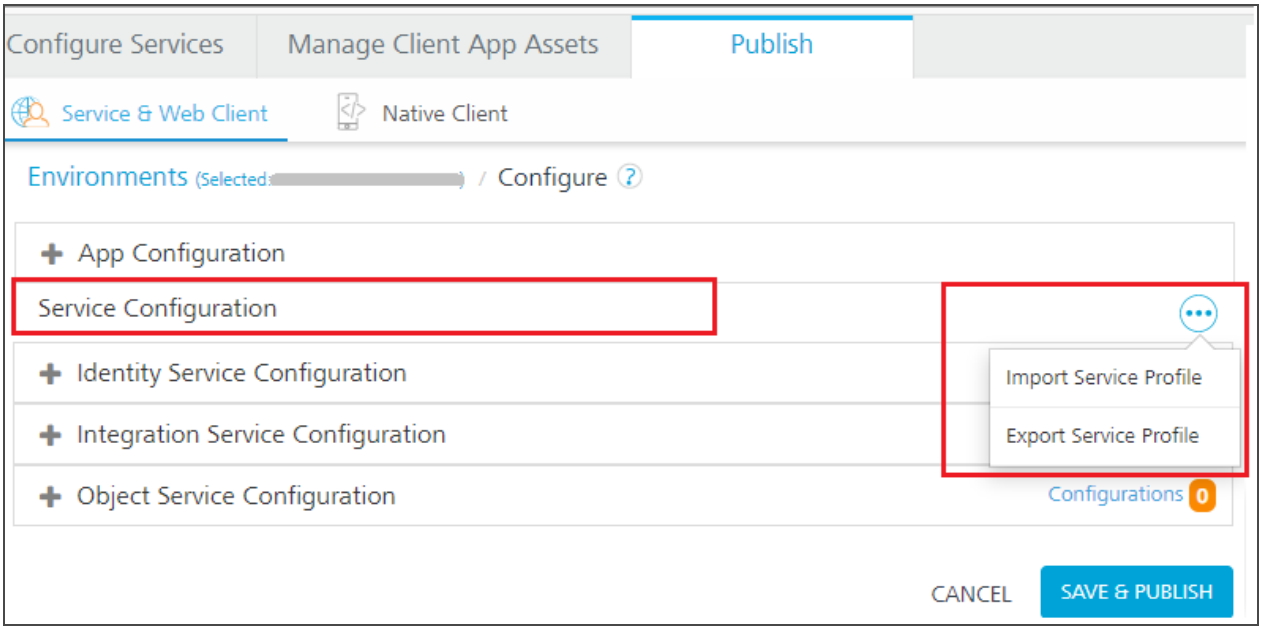
37.8.2 Export/Import Service Profiles using Kony Fabric

To export or import a service configuration profile from the **Publish** tab, do the following:

1. In the Publish > Service & Web Client page, select the required environment.
2. Click **CONFIGURE & PUBLISH**.



3. In **Service Configuration**, click the **More Options** button. The Import Service Profile and Export Service Profile options are shown in the list.



- To export the service profile, click the **Export Service Profile** option. The service profile details are downloaded in a JSON file.

- To import the service profile, click the **Import Service Profile** option and navigate to the required JSON file of a service profile.

  > ***Important:*** After you import a service profile, the existing configuration parameters get modified. This modification is based on the data imported before the **SAVE & PUBLISH** button was clicked.

4. Click **SAVE & PUBLISH** to start publishing the updated details in the service profile to the selected environment. The process of publishing the app begins.

37.8.3 Export/Import Service Profiles using MFCLI

You can export or import service profiles by using MFCLI commands for continuous integration.

37.8.3.1 export-config command

The `export-config` helps you to export the Service Profile for the existing services (Integration, Identity and Object), for a given app and an environment. The profile is downloaded as per the specified .json file.

- To export a service profile from a Cloud (manage.kony.com) environment

  ```
  java -jar mfcli.jar export-config -u <user> -p <password> -t
  <account id> [-f <file name>  [-a <app name>] [-v <app version>]
  [-e <environment name>]
  ```

  For example:

  ```
  java -jar mfcli.jar export-config -u abc@kony.com -p password -t
  100054321 -f "C:\\tmp\\Sample.json" -a MyApp_23 -v 1.0 -e
  MyCloudEnvironment
  ```

- To export a service profile from an on-premise installation

```
 java -jar mfcli.jar export-config -u <user> -p <password> -au
<Identity URL> -cu <Console URL> [-f <file name>  [-a <app name>]
[-v <app version>] [-e <environment name>]
```

For example:

```
java -jar mfcli.jar export-config -u abc@kony.com -p password -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -f
"C:\\tmp\\Sample.json" -a MyApp_23 -v 1.0 -e MyCloudEnvironment
```

The following arguments are supported for the export-config command:

```
For example, to get the summary help on all the commands,
java -jar mfcli.jar export-config help
Usage: Run the self-executable JAR with relevant arguments:
```

Arguments	Description
-t, --account	Nine-digit ID of the Kony Cloud account (visible on top right corner in Console), for example, 100054321. Not relevant for an on-premise installation.
* -a, --app	Name of the app for which Service Configuration profile is to be exported.
-cu, --console	URL of Kony Fabric Console (without context path), relevant for on-premise installation only. For example, http://10.10.24.78:8081

Arguments	Description
* -e, --environment	Name of the environment for which a Service Configuration profile is to be exported.
* -f, --file	Name of the file to import or export. For example, C:\\tmp\\Sample.json.
-au, --identity	URL of Kony Fabric Identity Services (without context path), relevant for on-premise installation only. For example, http://10.10.24.79:8080
--mfa	If specified, Multi-Factor Authentication is enabled. The secret key for MFA required for generating one-time password (OTP) needs to be specified in the properties file. The default value is set to `false`
-p, --password	Password for the Kony user account. This could be plain text or, encrypted using 'encrypt' command. This is mandatory.
-u, --user	Kony user required for authentication. For example, abc@kony.com. This is mandatory.
-v, --version	Version of the app for which a Service Configuration profile to be exported. The default version is set to `1.0`

37.8.3.2 import-config command

The `import-config` command helps you to import the specified Service Profile in a .json file into the given app and in an environment.

- To import a service profile to a Cloud (manage.kony.com) environment

```
java -jar mfcli.jar import-config -u <user> -p <password> -t
<account id> [-f <file name> [-a <app name>] [-v <app version>]
[-e <environment name>]]
```

For example:

```
java -jar mfcli.jar import-config -u abc@kony.com -p password -t
100054321 -f "C:\\tmp\\Sample.zip" -a MyApp_23 -v 1.0 -e
MyCloudEnvironment
```

- To import a service profile to an on-premise installation

```
 java -jar mfcli.jar import-config -u <user> -p <password> -au
<Identity URL> -cu <Console URL> [-f <file name>  [-a <app name>]
[-v <app version>] [-e <environment name>]
```

For example:

```
java -jar mfcli.jar import-config -u abc@kony.com -p password -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -f
"C:\\tmp\\Sample.zip" -a MyApp_23 -v 1.0 -e MyCloudEnvironment
```

  - The following arguments are supported for the import-config command:

```
 For example, to get summary help on all the commands,
java -jar mfcli.jar import-config help

Usage: Run the self-executable JAR with relevant arguments:
```

Arguments	Description
-t, --account	Nine-digit ID of the Kony Cloud account (visible on top right corner in Console), for example, 100054321. Not relevant for an on-premise installation.
* -a, --app	Name of the app for which a Service Configuration profile is to be exported.
-cu, --console	URL of Kony Fabric Console (without context path), relevant for on-premise installation only. For example, http://10.10.24.78:8081
* -e, --environment	Name of the environment for which a Service Configuration profile is to be exported.
* -f, --file	Name of the file to import or export. For example, C:\\tmp\\Sample.json.
-au, --identity	URL of Kony Fabric Identity Services (without context path), relevant for on-premise installation only. For example, http://10.10.24.79:8080
--mfa	If specified, Multi-Factor Authentication is enabled. The secret key for MFA required for generating one-time password (OTP) needs to be specified in the properties file. The default value is set to: `false`

Arguments	Description
-p, --password	Password for the Kony user account. This could be plain text or, encrypted using 'encrypt' command. This is mandatory.
-u, --user	Kony user required for authentication, For example, abc@kony.com. This is mandatory.
-v, --version	Version of the app for which a Service Configuration profile to be exported. The default version is set to `1.0`.

38. Continuous Integration with Kony Fabric

Continuous Integration is a key modern day software development practice. Any software development project doing Agile development probably would be doing Continuous Integration (CI) too.

Kony Fabric being a SDLC tool, supports Continuous Integration. Kony Fabric has REST APIs for import/export of an application and publish/unpublish of an application to facilitate CI. You can consume these REST APIs directly if needed. For this, refer to the Continuous Integration with Kony Fabric APIs documentation.

While deploying the application to different environments, you may need to reconfigure certain properties, such as the end-point URL to which the application should talk in the production environment. For these kind of reconfiguration, you can use the Reconfiguration at Publish support. You can set environment specific properties as one time activity using Kony Fabric Console and these will automatically get applied during publish in CI flow.

38.1 Kony Fabric Command Line Utility

Using REST APIs may not be ideal in some cases. For example, a developer uses a shell script for Continuous Integration, then calls REST APIs and does all the response handling. The process can become tedious and time-consuming. To make such scenarios easier, we offer Kony Fabric Command Line Utility. The utility helps you export/import or publish/unpublish an application on on-premise Kony Fabric and Kony Cloud environments.

38.1.1 Download Links

Kony Fabric Command Line Utility can be downloaded from Kony download center.

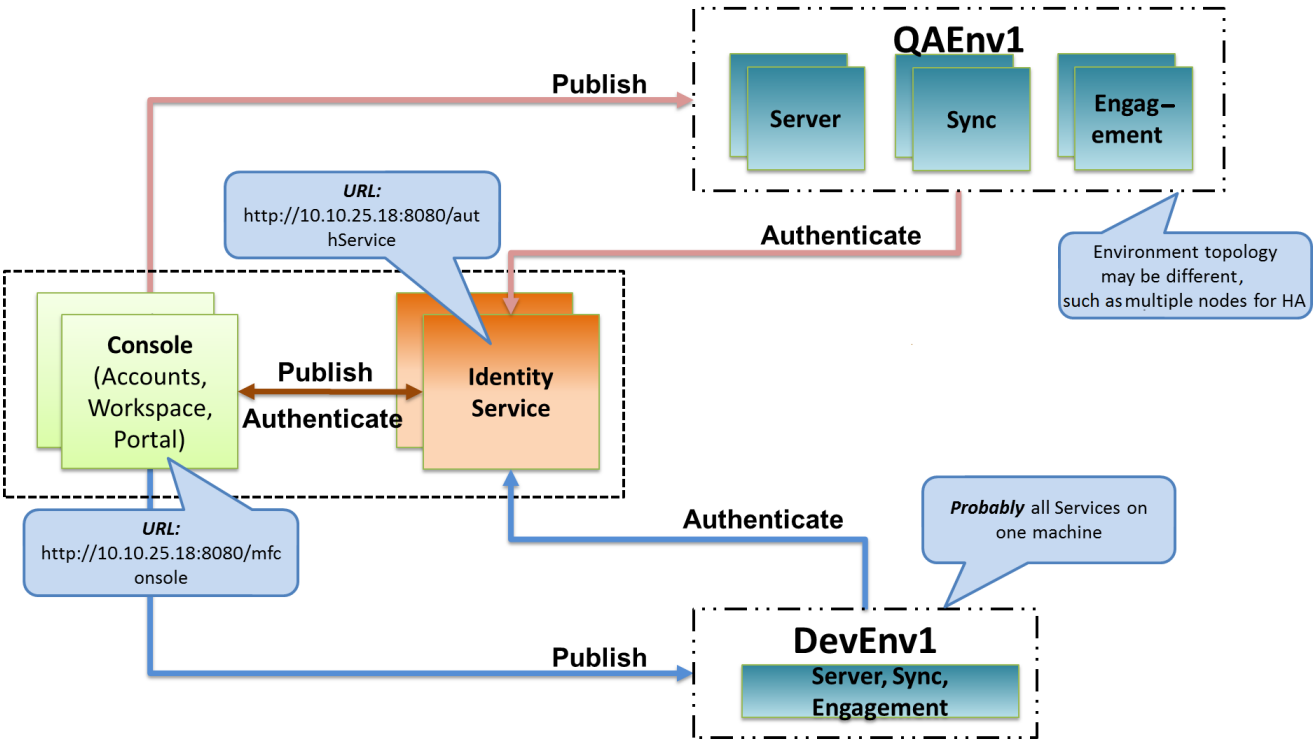
38.1.2 Examples

This section will help you better understand the utility. For example, company MoBoCo is developing the app MoBoApp. Some of the app's developers use the development environment called DevEnv1. As part of the continuous integration job that runs every night, the team wants to push the MoBoApp from DevEnv1 to QAEnv1 . The team wants automation and testing to occur at night and the following

day. Consider the following scenarios.

Scenario 1 - Promoting an app from the local development environment to local QA environment every night

There is **one Kony Fabric Console connected to two environments - DevEnv1 and QAEnv1**. The following image shows the Kony Fabric topology:



Using the preceding topology to promote the MoBoApp from DevEnv1 to QAEnv1 every night, a user can do the following:

```
 java -jar mfcli.jar Publish -u ciuser@moboco.com -p ciuserpwd -e
QAEnv1 -a MoBoApp -au http://10.10.25.18:8080 -cu
http://10.10.25.18:8080
```

Here, `ciuser@moboco.com` is the user with password `ciuserpwd`. The user's account helps to promote the `MoBoApp` to `QAEnv1`. Identity (Auth) Service and Console are running on `http://10.10.25.18:8080`. You can also encrypt the password before exporting an app.

> ***Note:*** The publish command could be used to publish a new application or to overwrite an existing application.

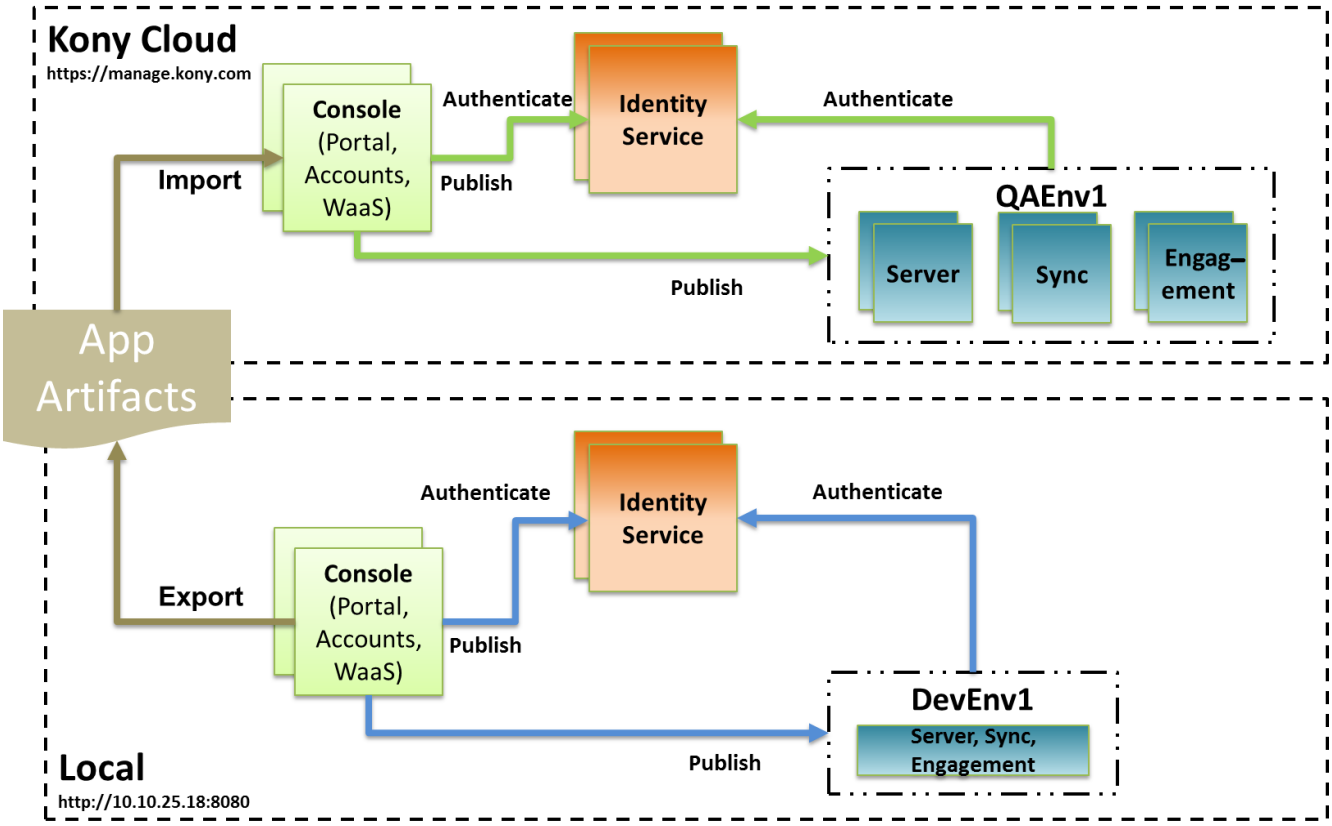
The following example shows how the publish command is used:

```
 To publish an application from an on-premise installation,
java -jar mfcli.jar Publish -u [user] -p [password] -e [environment
name] -a [app name] -au [Identity Service URL] -cu [Console URL]
```

If the end-point URL or any other property is different in QA environment, you can use the Application Reconfiguration support to set these properties at one time action in Kony Fabric Console. The environment specific properties will automatically get applied every time you publish.

Scenario 2 - Promoting an app from the local development environment to QA environment running on Kony Cloud every night

This case is more complex. Developers are working against a local environment (DevEnv1), but QA is testing through an environment in Kony Cloud (QAEnv1). There are **two MF Consoles**. The example is a hybrid scenario of on-premise and the cloud. The following image shows the Kony Fabric topology:



In this example, a user needs to perform the following four steps:

1. Encrypt the password of a user who needs to promote the `MoBoApp` to `QAEnv1account`.
2. Export the app from DevEnv1 (as a .zip file).
3. Import the app (as a .zip file) into Kony Cloud account.
4. Publish the app to QAEnv1.

38.1.3 Minor difference between on-premise Kony Fabric and Kony Cloud

The command line utility supports on-premise Kony Fabric and Kony Cloud. However, there are minor differences in command-line arguments. For on-premise, the utility needs to know the URL of the

Identity Services and Console. For Kony Cloud, since the URL is constant (https://manage.kony.com), and is hard-coded into the utility for convenience. However, Kony Cloud is a multiple account (tenant) service, and a user must specify the account ID using -t option. The Kony Cloud account ID is a nine-digit number (such as 100054321) and is located in the top-right corner of the console.

In the scenario, a user must perform the following steps:

1. **Encrypt the password**.

   To encrypt the password, run the following command.

   ```
   java -jar mfcli.jar  encrypt password ciuserpwd
   ```

   An encrypted password is generated for the preceding ciuserpwd. Use the encrypted password in the following commands in the scenario.

2. **Export the app from DevEnv1 (as a zip file)**

   To export the app, a user must run the following command.

   ```
   java -jar mfcli.jar Export -u ciuser@moboco.com -p encrypted_
   password -a MoBoApp -f .\MoBoApp.zip -au http://10.10.25.18:8080
   -cu http://10.10.25.18:8080.
   ```

   In the scenario, `ciuser@moboco.com` is the user with password `encrypted_password`. The user's account helps to export the `MoBoApp` as `MoBoApp.zip`. The Identity (Auth) Services and console are running on `http://10.10.25.18:8080`.

   The following is an example of how the export command is used:

   ```
   To export an application from an on premise installation,

   java -jar mfcli.jar Export -u [user] -p [encrypted_password] -a
   [app name] -f [file name] -au [Identity URL] -cu [Console URL]
   ```

3. **Import the app (as a zip file) into Kony Cloud Account**

```
 java -jar mfcli.jar Import -u ciuser@moboco.com -p encrypted_
password -a MoBoApp -f .\MoBoApp.zip -t 100054321
```

In the example, ciuser@moboco.com is the user with encrypted_password. The user's account helps to import the MoBoApp from MoBoApp.zip.

> ***Note:*** Since no URL is provided for Identity Service and console, it is assumed that the URL is Kony Cloud (https://manage.kony.com). Since Kony Cloud is a multiple tenants (account) service, a user must specify the account ID using -t. The account ID in the preceding example is 100054321.

> ***Note:*** In the example, it is assumed a user is trying to overwrite the existing MoBoApp in the account. If this is new application, then do not specify any name with -a option.

The following is an example of how the import command is used:

```
To import an application to manage.kony.com,

java -jar mfcli.jar import -u <user> -p <password> -t <account
id> [-f <file name> | -r <directory name>] [-a <app name>] [-v
<app version>] [-ay] [-to <time in secs>]
```

4. **Publish the app to QAEnv1**

```
 java -jar mfcli.jar Publish -u ciuser@moboco.com -p encrypted_
password -e QAEnv1 -a MoBoApp -t 100054321
```

This method is similar to local publish, except the -t option specifies the account ID.

The following is an example of how the publish command is used:

```
 To publish an application to manage.kony.com,

java -jar mfcli.jar Publish -u [user] -p [encrypted_password] -e
[environment name] -a [app name] -t [account id]
```

If the end-point URL or any other property is different in QA environment running on Kony Cloud, you can use the Application Reconfiguration support to set these properties at one time action in Kony Fabric Console. The environment specific properties will automatically get applied every time you publish.

38.1.4 Miscellaneous Features

38.1.4.1 Get Usage

Get the usage:

```
 For example, to get summary help on all the commands,
java -jar mfcli.jar help

Following commands are supported:-

    account-config     Gets Auth url information for the Kony Fabric
installation. Applicable for on-premise installation only.
    appinfo             Displays the App key, App Secret and Service
URL for an application
    authenticate         Authenticates user with given credentials and
returns the Identity token
    binary-upload      Uploads a client binary to Fabric. Click here
for more details on the binary-upload command

build               Command to build Visualizer apps for different
channels. Click here for more details on the build command
build-cancel       Command to cancel build for viz project
```

```
    build-download      Command to download artifacts for given build
    build-status        Command to check build status or fetch status
url for given build
    build-trigger       Command to trigger build for viz project
    build-upload        Command to upload viz project for build

    diagnostics         Gets the diagnostics information from the Kony
Fabric installation. Applicable for on-premise installation only.

    encrypt             Generates an encrypted password from plain text
password using default key. To use custom encryption key, set the
property password.encryption.key.
    export              Exports the specified application as a .zip
file or a directory
    export-config-properties Exports the configurable properties as a
specified .zip file or directory. Click here for more details on the
export-config-properties command.

export-dashboards Export the custom dashboards from Fabric.
    export-reports      Export custom reports for a specified
application from Fabric.
    export-service      Exports the specified service as a .zip file or
a directory
    export-config   Exports the existing integration, identity and
object Service Configuration profile, for the given app and
environment. Profile is downloaded as the specified .json file. Click
here for more details on the export-config command
    generate-docs       Generates HTML documentation for the specified
Kony Fabric application along with the OpenAPI (Swagger) file
    healthcheck         Gets the health check information for the
environment.
    help                Help on the tool or specific command
    import              Imports the specified .zip file or the
```

```
directory as an application
    import-config-properties Imports the specified .zip file or the
directory as configurable properties file. Click here for more details
on the import-config-properties command.
import-dashboards  Import the custom dashboards into Fabric.
    import-reports     Import the custom reports for a specified
application into Fabric.
    import-service     Imports the specified .zip file or the
directory as a service
    import-config   Imports the specified Service Configuration
profile .json file,into the given app and environment. Click here for
more details on the import-config command.
    license            Gets the license information from the Kony
Fabric installation. Applicable for on-premise installation only.
    lock-config        Command to lock services given a lock
configuration json and an App zip. Creates a clone of the provided app
with lock configuration applied. Click here for more details on lock-
config command
    merge-service-zip  Merges a given service package with a template
package.
    native-publish     Publishes the native binaries to the
environment. Click here for more details on native-publish command

    publish             Publishes the application to the environment.
    publish-cancel     Cancel an in-progress publish/unpublish of an
application to an environment.
    publish-status     Queries the publish/unpublish status of an
application in an environment.
    set-appversion     Sets a particular version as default among all
successfully published versions of an app.
    unlock-config      Command to remove all lock configurations from
the provided App zip. Creates a clone of the provided app with locks
removed. Click here for more details on unlock-config command
    unpublish          Unpublishes the application from the
```

```
environment.
    wrap                Wraps an existing client binary with Kony
Fabric SDK for getting analytics. Applicable for Kony Cloud only.
    wrap-delete         wrap-delete command delete the analytics app
and the stats captured will be not be accessible. Applicable for Kony
Cloud only.
    wrap-fetch          Fetch wrapped client binary. Applicable for
Kony Cloud only.


Usage: Run the self-executable JAR with relevant arguments.
 -u [user]              : Kony user required for authentication, for
example, abc@kony.com

 -p [password]          : Password of the Kony user. This could be plain
text or encrypted using
       'encrypt' command.

 -t [account id]        : 9 digit id of the Kony Cloud account (visible
in top right corner in Console), for example, 100054321. Not relevant
for an on-premise installation.

 -a [app name]          : Name of the app to be
published/unpublished/exported/imported. If importing a new app, this
should not be specified.

 -e [environment name]: Name of the environment to publish
to/unpublish from. Not relevant for import/export.

-f [file name]          : Name of the exported file or file to import.
For example, c:\\tmp\\app.zip. Not relevant for publish/unpublish.

--release-lock         : Forcibly release lock taken by previous
```

```
operation (use with caution) when doing publish/unpublish. Default:
false

-au [Identity URL] : URL of Kony Fabric Identity Services, relevant
for on premise installation only. For example, http://10.10.24.79:8080

- cu [Console URL]   : URL of Kony Fabric Console, relevant for on
premise installation only. For example, http://10.10.24.78:8081

-r, --directory    : Name of the directory to export to or import
from. For example, C:\\src\\MyApp. Either -f or -r has to be
specified.

--skipwebapp       :   If true, skips the publish of web app/zip during
app publish. Default: false

--mfa
       If specified, multi-factor authentication is enabled. The
secret key for
       multi-factor authentication required for generating one time
password (OTP) needs to
       be specified in the properties file.
       Default: false
       For more details to enable MFA, refer Support for Multi-Factor
Authentication from MFCLI

  -v, --version
       Version of the app to be imported, defaults to the version info
present
       in the package. If version info is not present in package,
defaults to 1.0.
       Default: 1.0
```

```
To get more information on a specific command, type 'java -jar
mfcli.jar help <command>'
For example, to get detailed help on a particular command including
examples on publish,
java -jar mfcli.jar help publish


To get details of an application from for Kony Cloud (manage.kony.com)
environment
java -jar mfcli.jar appinfo -u <user> -p <password> -t <account id> -a
<app name> [-v <app version>] -e <environment name>
java -jar mfcli.jar appinfo -u abc@kony.com -p password -t 100054321 -
a MyApp -v 2.0 -e MyEnv


To get details of an application from on-premise installation
java -jar mfcli.jar appinfo -u <user> -p <password> -au <Identity URL>
-cu <Console URL> -a <app name> [-v <app version>] -e <environment
name>
java -jar mfcli.jar appinfo -u abc@kony.com -p password -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -a MyApp -v 2.0 -e
MyEnv


To publish an application to manage.kony.com,
java -jar mfcli.jar Publish -u [user] -p [password] -e [environment
name] -a [app name] -t [account id] [--skipwebapp]
java -jar mfcli.jar Publish -u abc@kony.com -p password -e MyEnv -a
MyApp -t 100054321 --skipwebapp

To publish an application to an on-premise installation,
java -jar mfcli.jar Publish -u [user] -p [password] -e [environment
name] -a [app name] -au [Identity URL] -cu [Console URL] [--
```

```
skipwebapp]
java -jar mfcli.jar Publish -u abc@kony.com -p password -e MyEnv -a
MyApp -au http://10.10.24.79:8080 -cu http://10.10.24.78:8081 --
skipwebapp

To unpublish an application from manage.kony.com,
java -jar mfcli.jar Unpublish -u [user] -p [password] -e [environment
name] -a [app name] -t [account id]
java -jar mfcli.jar Unpublish -u abc@kony.com -p password -e MyEnv -a
MyApp -t 100054321

To unpublish an application from an on-premise installation,
java -jar mfcli.jar Unpublish -u [user] -p [password] -e [environment
name] -a [app name] -au [Identity URL] -cu [Console URL]
java -jar mfcli.jar Unpublish -u abc@kony.com -p password -e MyEnv -a
MyApp -au http://10.10.24.79:8080 -cu http://10.10.24.78:8081

To export an application from manage.kony.com,
java -jar mfcli.jar Export -u [user] -p [password] -a [app name] -t
[account id] -f [file name] | -r <directory name>
java -jar mfcli.jar Export -u abc@kony.com -p password -a MyApp -t
100054321 -f "c:\\tmp\\ExportedApp.zip"

To export an application from an on-premise installation,
java -jar mfcli.jar Export -u [user] -p [password] -a [app name] -f
[file name] -au [Identity URL] -cu [Console URL]
java -jar mfcli.jar Export -u abc@kony.com -p password -a MyApp -f
"c:\\tmp\\ExportedApp.zip" -au http://10.10.24.79:8080 -cu
http://10.10.24.78:8081

To import an application to manage.kony.com,
java -jar mfcli.jar import -u <user> -p <password> -t <account id> [-f
<file name> | -r <directory name>] [-a <app name>] [-v <app version>]
```

```
[-ay] [-to <time in secs>]

java -jar mfcli.jar Import -u abc@kony.com -p password -a MyApp -t
100054321 -f "c:\\tmp\\AppToImport.zip"


-v, --version If present overwrites only an existing version of the
apps. It there is no existing app version, the import command is
terminated.

-ay, --async If present imports the app asynchronously, polls and
responds the status.
Default: false

-to, --timeout If present Timeout in seconds for asynchronous import.
Polls the status of import until the given timeout.
Default: 300

To import an application to an on-premise installation,
java -jar mfcli.jar import -u <user> -p <password> -au <Identity URL>
-cu <Console URL> [-f <file name> | -r <directory name>] [-a <app
name>] [-v <app version>] [-ay] [-to <time in secs>]

java -jar mfcli.jar Import -u abc@kony.com -p password -a MyApp -f
"c:\\tmp\\AppToImport.zip" -au http://10.10.24.79:8080 -cu
http://10.10.24.78:8081

To encrypt a password,
For example, to encrypt password 'mypassword',
  java -jar mfcli.jar encrypt mypasswordOr,
  java -DKONY_MFCLI_ENCRYPTION_KEY=2d04b412-64cc-4066-a6c8-
9c1417b9b85f -jar mfcli.jar encrypt mypassword
```

38.1.4.2 Unpublish Command

The Unpublish feature helps you undo the publish command. The syntax for the Unpublish command is that same as the publish command's. **For republish, do not unpublish**. To overwrite an existing publish, republish by executing the publish command again. When a user does unpublish, all of the app's metadata are removed. The next publish creates fresh metadata, such as a new app key/secret. In case of republishing, when a user does publish multiple times to republish an app, all of the app's metadata are retained.

38.1.4.3 Authenticate API

If you have log in credentials, this authenticate command helps you to get the Auth Token out of your login credentials. For example, you can use this command to get a valid Kony Identity token and be able to use that token with subsequent REST API calls to interact directly like using Engagement service APIs or AppFactory APIs.

The API supports for Kony Auth and External auth services.

```
for Kony Cloud (manage.kony.com) environment,
java -jar mfcli.jar authenticate -u <user> -p <password> -t <account
id>
java -jar mfcli.jar authenticate -u abc@kony.com -p password -t
100054321
```

```
for on-premise installation,
java -jar mfcli.jar authenticate -u <user> -p <password> -au <Identity
URL> -cu <Console URL>
java -jar mfcli.jar authenticate -u abc@kony.com -p password -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081
```

```
To use external authentication
java -jar mfcli.jar authenticate -u <user> -p <pasword> -t <accountid>
--external-auth
```

38.1.4.4 Force Release Lock

This utility guarantees a consistent operation during the preceding import/export/publish/unpublish commands. The utility ensures that only one such operation per app occurs at a given time. To do this, the utility takes a lock in the database. If there is a software defect, the DB lock is not freed. A user can use `--release-lock` to force the lock release. This option should be used with caution because it can cancel any existing app publish.

38.2 Continuous Integration with Kony Fabric APIs

In the section, you can learn about the Continuous Integration (CI) capabilities by using APIs supported by Kony Fabric. Without using Kony Fabric Console, developers can automate the process of importing an app, exporting an app, and publishing an app by using APIs supported in CI. The details of the APIs are captured in the subsequent sections.

The following are two ways of publishing apps in Kony Fabric:

- Using Publish Apps in Kony Fabric Console.
- Using script by calling publish APIs through a command line interface (CLI) - for example, cURL or a .Java file.

> ***Note:*** cURL - a command line tool for getting or sending files using URL syntax. The user guide uses the cURL command to represent a mobile device making HTTPS API calls to a Kony Fabric environment. cURL is typically pre-installed on Linux and Mac systems. For Windows, go to http://curl.haxx.se/download.html, download cURL, and the SSL libraries required to connect to HTTPS URLs. For cURL commands and documentation, refer to http://curl.haxx.se/docs/

> ***Important:*** All the publish APIs in this section are auth protected.

The following APIs are supported for continuous integration of an app:

1. Fetch the Auth Token - Get Authentication Token Information via API
2. Export and App - Export the Kony Fabric App From Console via API
3. Import an App - Import a Kony Fabric App (.zip file) via API
4. Publish via API

38.2.1 Fetch the Auth Token

To fetch the claims token, invoke the Auth service login API and extract the “claims_token”.”value” from the response.

The API helps you log in as a management user and get back session token that can be used for invoking management APIs.

A successful response includes a session token.

If a user's credentials are incorrect, the log-in fails, and the system generates an HTTP 401 error.

Signature

```
https://accounts.auth.konycloud.com/login
```

Sample cURL Script

```
token="$(curl -H "Accept: application/json" --data "userid=
[abc@kony.com]&password=[pwd]" -X POST
"https://accounts.auth.konycloud.com/login" | awk -F 'claims_
token.*?value":"' '{print $2}' | awk -F '",' '{print $1}')"
```

Method

POST

Request Headers

Form post parameters

- userid={user id}
- password={password}

Sample Response:

```
 Login Success Response (HTTP Response Code 200 OK):
{
    "profile": {
        "email": "",
        "userid": "",
        "firstname": "",
        "lastname": ""
    },
    "refresh_token": "{refresh_token}",
    "claims_token": {
        "value": "{session_token}",
        "exp": 1402941484000 /* expiry time of session in seconds
since epoch */
    }
}

Sample error response: (HTTP Response Code 401 Unauthorized):
{
    "domain": "AUTH",
    "code": -4,
    "mfcode": "Auth-4",
    "message": "Invalid User Credentials",
    "details": {
        "message": "Invalid User Credentials",
        "errcode": 0,
        "errmsg": "Invalid User Credentials"
    },
```

```
    "requestid": "77cf1b1b-3bc5-4d73-84ec-806dcd60cf56",
    "httpstatus": "Unauthorized"
}
```

38.2.2 Export an App

The Apps Export API helps you export all the services and properties of Kony Fabric custom application from a specified workspace.

When you call the Apps Export API, the system exports the application in Export app structure.

Signature

```
 https:// {workspaceid}.workspace.konycloud.com/api/v1/ws/
{workspaceid}/apps/export?appName={appName}
```

Sample cURL Script

```
token="$(curl -H "Accept: application/json" --data "userid=
[abc@kony.com]&password=[pwd]" -X POST
"https://accounts.auth.konycloud.com/login" | awk -F 'claims_
token.*?value":"' '{print $2}' | awk -F '"," '{print $1}')"
```

Method

GET

Request Query Parameters (Mandatory)

- workspaceid : The accounts that can be accessed by a user are listed in the top right corner of Kony Fabric Console. The workspaceid is the account ID that corresponds to the selected account.
- appName : { Name of the Kony Fabric custom application that a user wants to export}

Request Headers Parameters (Mandatory)

X-Kony-Authorization : {Auth Service Token}

Response:

- **Sample Success Response**:

```
HTTP 200 OK
```

- **Sample Success Response Body**:

```
The export API returns a zip stream that can be converted to a required .zip file. For more details, refer to Export app structure.
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Failure Response Body Format**:

```
{
"domain": "WAAS"
"code": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
"httpstatus": "[Http Status String]"
}
```

- **Example Failure Response Body**:

```
{
    "domain": "WAAS",
    "code": -9,
    "message": "Invalid App Credentials",
    "details": {
        "message": "Invalid App Credentials",
        "errcode": 0,
        "errmsg": null
    },
    "httpstatus": "Unauthorized"
}
```

38.2.3 Import an App

The App Import API in Workspace service imports the App configuration in the package in a zip file (application zip). An application zip has app configuration, services, and messaging configuration. While importing, if these details are already present in the app, these details will be updated. Otherwise, these details are created.

Signature

```
 https:// {workspaceid}.workspace.konycloud.com/api/v1/ws/
{workspaceid}/apps/import?appName={appName}
```

Sample cURL Script

```
curl -F "appZip=@C:/Users/KH1895/Desktop/file.zip" -H "X-Kony-
Authorization:[auth token]" -X POST " https://
[workspaceid].workspace.konycloud.com/api/v1/ws/[workspaceid]
/apps/import"

curl -F "appZip=@C:/Users/KH1895/Desktop/file.zip" -H "X-Kony-
```

```
Authorization:[auth token]" -X POST " https://
[workspaceid].workspace.konycloud.com/api/v1/ws/[workspaceid]
/apps/import?appName=[app name]&suppressWarnings=true"
```

Method

POST

Request Query Parameters (Mandatory)

- workspaceid : The accounts that can be accessed by a user are listed in the top right corner of Kony Fabric Console. The Account ID corresponds to the selected account.

Request Query Parameters (Optional)

- appName : Name of the Kony Fabric custom application the user wants to create or update.
- suppressWarnings : When importing apps with identity services, the identity services are not imported if `suppressWarnings` flag is not set. The required services must be first created or updated manually and then the app needs to be imported. The query param must be set to true (suppressWarnings=true) when importing apps. When `suppressWarning` is true and if an identity provider exists in the workspace, the existing provider is linked to the app. Otherwise, the import fails.

Request Headers (Mandatory)

X-Kony-Authorization : {Auth Service Token}

Request Body

```
 Content-Type : multipart/form-data
{
   "appZip" : < Zip file content of the app. The zip file needs to be
in this format >
}
```

Response

- **Sample Success Response**:

```
HTTP 200 OK
```

- **Sample Success Response Body**:

  Returns the details of an app that is imported into the workspace in the following format:

```
 appBaseUrl - <baseurl>/api/v1/ws/{workspace id}/apps/{app id}

{
     "id": "<app id>",
      "type": "<type of app>",
       "name": "<app name>",
       "icon": "<app icon>",
        "description": "<app description>",
        "created_at": "<time of creation of app>",
        "created_by": "<puid of the owner who created the app>",
        "updated_at": "<time of updation of app>",
        "updated_by": "<puid of the owner who updated the app>",
        "_links":
         {
             …
         }
}
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Failure Response Body Format**:

```
 {
"domain": "WAAS"
"code": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
"httpstatus": "[Http Status String]"
}
```

- **Example Failure Response Body**:

```
 {
    "domain": "WAAS",
    "code": -9,
    "message": "Invalid App Credentials",
    "details": {
        "message": "Invalid App Credentials",
        "errcode": 0,
        "errmsg": null
    },
    "httpstatus": "Unauthorized"
}
```

38.2.4 Publish via APIs

Publishing a Kony Fabric app is a multi-step workflow. During publishing an app, the `next_step` object in the response of the publish API decides whether the next step to be executed.

a. Fetch Environment GUID of an Environment via API

b. Fetch App ID of the Imported Kony Fabric App via API

c. Publish a Kony Fabric App via API

- Initiate Publish
- Continue Publish

d. Unpublish a Kony Fabric App via API

- Initiate Unpublish
- Continue Unpublish

38.2.4.1 Fetch Environment Globally Unique Identifier (GUID)

To publish or unpublish an app to or from an environment, the environment must be uniquely identified through a GUID. To retrieve a GUID of an environment, call the following API and extract the “environment_guid” of the relevant one.

The API fetches the details of the environments that a user can access.

The `environment_guid` value retrieved from the response of the API and will be used for publishing or unpublishing an app.

Signature

```
https://manage.kony.com/api/v1_0/environments
```

Method

GET

Request Headers Parameters (Mandatory)

- X-Kony-Authorization : {Auth Service Token}
- Accept : {application/json}

Response:

- **Sample Success Response**:

```
HTTP 200 OK
```

- **Sample Success Response Body**:

```
[
    {
        "environment_guid": "b8ee8e32-ca1e-471d-b737-
cf792b182712",
        "name": "mbaas",
        …
    },
    {
        "environment_guid": "60ebcf2f-e327-492e-92ae-
e5914ecfeeed",
        "name": "reconfig",
        …
    },
    …
]
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

38.2.4.2 Fetch App Globally Unique Identifier (GUID)

This API fetches the list of apps.

Signature

```
https://{workspaceid}.workspace.konycloud.com/api/v1/ws/{workspaceid}
/apps
```

Method

GET

Request Query Parameters (Mandatory)

- workspaceid : The accounts that can be accessed by a user are listed in the top right corner of Kony Fabric Console. The Account ID corresponds to the selected account.

Request Headers Parameters (Mandatory)

- X-Kony-Authorization : {Auth Service Token}
- Accept : {application/json}

Response:

- **Sample Success Response**:

```
HTTP 200 OK
```

- **Sample Success Response Body**:

```
 Returns list of Kony Fabric custom apps in the specified
workspace.
{
 [
  {
       "id": "<app id>",
       "type": "<type of app>",
       "name": "<app name>",
       "description": "<app description>",
        "created_at": "<time of creation of app>",
        "created_by": "<puid of the owner who created the app>",
        "updated_at": "<time of updation of app>",
        "updated_by": "<puid of the owner who updated the app>",
        "_links":
         {
```

```
                …
            }
        }
    ],
    "count": "< number of custom Kony Fabric apps returned in this
API call >",
    "page": < page number >,
    "_links": {
            …
    },
    "totalcount": "< total number of custom Kony Fabric apps in
the workspace >",
    "pagesize": <page size>
}
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Failure Response Body Format**:

```
{
"domain": "WAAS"
"code": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
"httpstatus": "[Http Status String]"
}
```

- **Example Failure Response Body**:

```
{
    "domain": "WAAS",
    "code": -9,
    "message": "Invalid App Credentials",
    "details": {
        "message": "Invalid App Credentials",
        "errcode": 0,
        "errmsg": null
    },
    "httpstatus": "Unauthorized"
}
```

38.2.4.3 Publish a Kony Fabric App via API

38.2.4.4 Initiate Publish

The API helps you start publishing the services inside the application to an environment with the environment `id {env_guid}` that you received. For more details on how to get the environment `ID`, refer to Fetch Environment GUID.

When you call this API, the system starts publishing the app, and locks the app from publishing by different users until the publish completes or fails.

Signature

```
https://api.kony.com/api/v1_0/environments/{env_guid}/publish
```

Method

POST

Request Headers (Mandatory)

- X-Kony-Authorization : {Auth Service Token}
- Content-Type : {application/json or application/x-www-form-urlencoded}
- Accept : {application/json}
- Referer : https://manage.kony.com (The refer generates the URL required for next step of publishing.)

Request Headers (Optional)

- X-Kony-RequestId : {Request Id}
- X-HTTP-Method-Override

Request Body

```
 Mandatory

app_id : Id of the Application to be published. For details on how to fetch the app id, refer to Fetch App ID
```

```
Optional

action : release_lock (This action releases the previous locks on the app before attempting publish). If the action parameter is not provided, the value will be set to start by default.
```

Response

- **Sample Success Response**:

```
HTTP 201 CREATED
```

- **Sample Success Response Headers**:

```
"Set-Cookie" : "<Cookies value>"
```

- **Sample Success Response Body**:

```
{
        "_links" : {
                "self" : {
                        "href" : "https://accounts.kony.com/api/v1_0/environme
guid/publish/apps/{publish_id}"
                }
        }
        "status" : true,
        "message" : "Success",
        "publish_id" : "<GUID>",
        "next_step" : true,
        "next_step_description" : "<Message>"
}
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Sample Failure Response Body**:

```
 {
"domain": "accounts"
"mfcode": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
```

```
      "requestid": "Request Id"
"httpstatus": "[Http Status String]"
}
```

> ***Note:*** After a successful response from the initiate publish operation API, users must call the following PUT request. The PUT API should contain a header `Cookie` with the value from `Set-Cookie` header of the API response.

38.2.4.5 Continue Publish

The API helps you continue publishing the services inside the application to an environment with the `id {env_guid}` a user can access. For more details on how to get the `id {env_guid}`, refer to Fetch Environment GUID.

Signature

```
 https://api.kony.com/api/v1_0/environments/{env_guid}/publish/
{publish_id}
```

Method

PUT

Request Headers (Mandatory)

- X-Kony-Authorization : {Auth Service Token}
- Content-Type : {application/json or application/x-www-form-urlencoded}
- Accept : {application/json}
- **Cookie : {Value from Set-Cookie header in the response of POST /publish API}**
  You can get the set-cookie header from the Initiate Publish APIs

- Referer : https://manage.kony.com (The refer generates the URL required for next step of publishing.)

Request Headers (Optional)

- X-Kony-RequestId : {Request Id}
- X-HTTP-Method-Override

Request Body

```
Optional

action : cancel  | continue ( Default value is “continue” )
```

Response

- **Sample Success Response**:

```
HTTP 200 OK
```

- **Sample Success Response Body**:

```
{
       “_links” : {
                “self” : {
                        “href” : “https://accounts.kony.com/api/v1_0/environme
guid/publish/apps/{publish_id}”
                }
       }
       "status" : true,
       "message" : "Success",
       "publish_id" : "<GUID>",
```

```
        "next_step" : true,
        "next_step_description" : "<Message>"
}
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Sample Failure Response Body**:

```
{
"domain": "accounts"
"mfcode": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
        "requestid": "Request Id"
"httpstatus": "[Http Status String]"
}
```

38.2.5 Unpublish

Similar to publish, unpublishing of a Kony Fabric app is a multi-step workflow. During unpublishing an app, the `next_step` object in the response of the unpublish API decides whether the next step to be executed.

38.2.5.1 Initiate Unpublish

Signature

```
https://api.kony.com/api/v1_0/environments/{env_guid}/unpublish
```

Method

POST

Request Headers (Mandatory)

- X-Kony-Authorization : {Auth Service Token}
- Content-Type : {application/json or application/x-www-form-urlencoded}
- Accept : {application/json}
- Referer : https://manage.kony.com (The refer generates the URL required for next step of unpublishing.)

Request Headers (Optional)

- X-Kony-RequestId : {Request Id}
- X-HTTP-Method-Override

Request Body

```
Mandatory

app_id : Id of the Application to be published. For details on how to fetch the app id, refer to Fetch App ID
```

```
Optional

action : release_lock (This action releases the previous locks on the app before attempting publish). If the action parameter is not provided, the value will be set to unpublish by default.
```

Response

- **Sample Success Response**:

```
HTTP 201 CREATED
```

- **Sample Success Response Headers**:

```
“Set-Cookie” : “<Cookies>”
```

- **Sample Success Response Body**:

```
{
       “_links” : {
              “self” : {
                     “href” : “https://accounts.kony.com/api/v1_0/environme
guid/unpublish/apps/{publish_id}”
              }
       }
       "status" : true,
       "message" : "Success",
       "publish_id" : "<GUID>",
       "next_step" : true,
       "next_step_description" : "<Message>"
}
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Sample Failure Response Body**:

```
{
"domain": "accounts"
"mfcode": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
       "requestid": "Request Id"
"httpstatus": "[Http Status String]"
}
```

> ***Note:*** After a successful response from the initiate unpublish operation API, users must call the following "PUT" API. The PUT API should contain a header `Cookie` with the value from the `Set-Cookie` header of the API response.

38.2.5.2 Continue Unpublish

The API helps you to continue unpublishing the services inside the application to an environment with the `id {env_guid}` that the user can access. For more details on how to get the `id {env_guid}`, refer to Fetch Environment GUID.

Signature

```
 https://api.kony.com/api/v1_0/environments/{env_guid}/unpublish/
{publish_id}
```

Method

PUT

Request Headers (Mandatory)

- X-Kony-Authorization : {Auth Service Token}
- Content-Type : {application/json or application/x-www-form-urlencoded}

- Accept : {application/json}
- Cookie : {Value from Set-Cookie header in the response of POST /unpublish API}
- Referer : https://manage.kony.com (The refer generates the URL required for next step of unpublishing.)

Request Headers (Optional)

- X-Kony-RequestId : {Request Id}
- X-HTTP-Method-Override

Request Body

```
Optional

action : cancel  | continue ( Default value is “continue” )
```

Response

- **Sample Success Response**:

```
HTTP 200 OK
```

- **Sample Success Response Body**:

```
{
       “_links” : {
                “self” : {
                        “href” : “https://accounts.kony.com/api/v1_0/environme
guid/unpublish/apps/{publish_id}”
                }
       }
```

```
        "status" : true,
        "message" : "Success",
        "publish_id" : "<GUID>",
        "next_step" : true,
        "next_step_description" : "<Message>"
}
```

- **Sample Failure Response - if a user does not have access to the workspace**:

```
HTTP 401 Unauthorized
```

- **Sample Failure Response - if export failed due to an internal failure**:

```
HTTP 500 Internal Server Error
```

- **Sample Failure Response Body**:

```
{
"domain": "accounts"
"mfcode": [error-code], /* a number/string based on the domain */
"message": "Message",
"details": [Details of Error Message],
        "requestid": "Request Id"
"httpstatus": "[Http Status String]"
}
```

38.3 Continuous Integration for Binary-Upload and Native-Publish

Binary-upload and Native-publish commands are introduced in MFCLI V8 SP4.

- `binary-upload` command uploads a client binary to Kony Fabric. Supported platforms for client binaries are ios_phone, ios_tablet, android_phone, android_tablet, windows_phone, and web_phone.
- `native-publish` command publishes the native binaries to the environment.

38.3.1 binary-upload command

- To upload a native client binary to a Cloud (manage.kony.com) environment

```
 java -jar mfcli.jar binary-upload -u <user> -p <password> -t
<account id> -a <app name> [-v <app version>] -bp <Path to
binary> -plat <platform name> --binary-version <Binary version> -
-description <Description> --display-name <Display Name> --
appIcon <Path to app icon> [-ssdirpath <Path to screenshots
directory>]
```

```
 For example:

    java -jar mfcli.jar binary-upload -u abc@kony.com -p password
-t 100054321 -a MyApp -v 1.0 -bp "D:\\WorkDir\\iosClient.ipa" -
plat ios_phone --binary-version 1.0.2 --description "This is
iphone binary" --display-name "sampleDisplayName" --appIcon
"D:\\WorkDir\\icon1.png" -ssdirpath "D:\\WorkDir\\screenshotsDir"
```

- To upload a native client binary to an on premise installation

```
 java -jar mfcli.jar binary-upload -u <user> -p <password> -au
<Identity URL> -cu <Console URL> -a <app name> [-v <app version>]
-bp <Path to binary> -plat <platform name> --binary-version
<Binary version> --description <Description> --appIcon <Path to
app icon> [-ssdirpath <Path to screenshots directory>]
```

```
For example:
   java -jar mfcli.jar binary-upload -u abc@kony.com -p password
-au http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -a MyApp
-v 1.0 -bp "D:\\WorkDir\\iosClient.ipa" -plat ios_phone --binary-
version 1.0.2 --description "This is iphone binary" --display-
```

```
name "sampleDisplayName" --appIcon "D:\\WorkDir\\icon1.png" -
ssdirpath "D:\\WorkDir\\screenshotsDir"
```

- The following arguments are supported for the binary-upload command:

```
 For example, to get summary help on all the commands,
java -jar mfcli.jar help binary-upload

Usage: Run the self-executable JAR with relevant arguments.
  Options:
      -t, --account
       9 digit id of the Kony Cloud account (visible in top
right corner in Console) for e.g. 100054321. Not relevant for
an on-premise installation.
  * -a, --app
       Name of the app to which binary to be uploaded.
    --appIcon
       Path to app icon
  * -bp, --binary-path
       Path to the client binary file .ipa or .apk
  * --binary-version
       The version of the client binary.
       Default: 1.0.0
    --channels
       Comma separated channel names of web build. Valid
values are web_phone,web_tablet,responsive_web
    -cu, --console
       URL of Kony Fabric Console (without context path),
relevant for on-premise installation only. For e.g.
http://10.10.24.78:8081
  * --description
       Description of client binary.
  --display-name
       Display name of client binary.
```

```
    -au, --identity
       URL of Kony Fabric Identity Services (without context
path), relevant for on-premise installation only. For e.g.
http://10.10.24.79:8080
    --mfa
       If specified, multi-factor authentication is enabled.
The secret key for multi-factor authentication required for
generating one time password (OTP) needs to be specified in
the properties file.
       Default: false
    -p, --password
       Password for the Kony user. This could be plain text
or, encrypted using 'encrypt' command. This is mandatory.
  * -plat, --platform
       The platform of the client binary.
       Possible Values: [ios_phone, ios_tablet, android_
phone, android_tablet, windows_phone, web_phone]
    -ssdirpath, --screenshot-directory-path
       Path to the screenshots folder.
    -u, --user
       Kony user required for authentication, for e.g.
abc@kony.com. This is mandatory.
    -v, --version
       The app version for which the binary to be uploaded.
       Default: 1.0
```

38.3.2 native-publish command

The native-publish command helps you publish the native client binaries to the environment.

- To publish native client binaries to a Cloud (manage.kony.com) environment

```
 java -jar mfcli.jar native-publish -u <user> -p <password> -t
<account id> -a <app name> [-v <app version>] -e <environment
name> -cbmeta <client binary meta>
```

```
 For example:

    java -jar mfcli.jar native-publish -u abc@kony.com -p
password -t 100054321 -a MyApp -v 2.0 -e MyEnv -cbmeta "{"ios_
phone" : 1.2.1, "android_phone" : 1.2.0}"
```

- To publish native client binaries to an on premise installation

```
 java -jar mfcli.jar native-publish -u <user> -p <password> -au
<Identity URL> -cu <Console URL> -a <app name> [-v <app version>]
-e <environment name> -cbmeta <client binary meta>
```

```
 For example:

    java -jar mfcli.jar native-publish -u abc@kony.com -p
password -au http://10.10.24.79:8080 -cu http://10.10.24.78:8081
-a MyApp -v 2.0 -e MyEnv -cbmeta "{"ios_phone" : 1.2.1, "android_
phone" : 1.2.0}"
```

  - The following arguments are supported for the native-publish command:

```
 For example, to get summary help on all the commands,
java -jar mfcli.jar help native-publish

Usage: Run the self-executable JAR with relevant arguments.
  Options:
      -t, --account
       9 digit id of the Kony Cloud account (visible in top
right corner in Console) for e.g. 100054321. Not relevant for
```

```
an on-premise installation.
  * -a, --app
       Name of the app to be published/unpublished.
  * -cbmeta, --clientbinary-meta
       Json string of native channel name vs binary version
    -cu, --console
       URL of Kony Fabric Console (without context path),
relevant for on-premise installation only. For e.g.
http://10.10.24.78:8081
  * -e, --environment
       Name of the environment to publish to/unpublish from.
    -au, --identity
       URL of Kony Fabric Identity Services (without context
path), relevant for on-premise installation only. For e.g.
http://10.10.24.79:8080
    --mfa
       If specified, multi-factor authentication is enabled.
The secret key for multi-factor authentication required for
generating one time password (OTP) needs to be specified in
the properties file.
       Default: false
    -p, --password
       Password for the Kony user. This could be plain text
or, encrypted using 'encrypt' command. This is mandatory.
    --release-lock
       Forcibly release lock taken by previous operation (use
with caution) when doing a publish/unpublish.
       Default: false
    -u, --user
       Kony user required for authentication, for e.g.
abc@kony.com. This is mandatory.
    --verbose
```

```
      If specified, prints details of all steps.
      Default: false
   -v, --version
      Version of the app to be published/unpublished.
      Default: 1.0
```

38.4 Continuous Integration for Cloud Build

Build is a command introduced in MFCLI V8 SP4. You can use the command to build and generate native app binaries for the selected native platforms. The Build command helps you build native client binaries on Cloud.

38.4.1 build command

To build and generate native client binaries on a Cloud (manage.kony.com) environment

```
  java -jar mfcli.jar build -u <user> -p <password> -t <account id> -e
<environment name> -pp <properties path> -sp <src-code path> -od
<output directory> [ -pt <timeout in seconds>]

java -jar mfcli.jar build -u abc@kony.com -p password -t 100054321 -e
MyEnv -pp "D:\WorkDir\prop.json" -sp "D:\WorkDir\src.zip" -od
"D:\WorkDir\output" -pt 20
```

- The following arguments are supported for the build command:

```
 build   Command to build Visualizer apps for different channels.

Usage: Run the self-executable JAR with relevant arguments.
  Options:
      -t, --account
       9 digit id of the Kony Cloud account (visible in top right
```

```
corner in Console) for e.g. 100054321. Not relevant for an on-
premise installation.
    -e, --environment
       Name of the environment to use for building.
    -mpt, --max-poll-timeout
       Timeout in minutes to be used to stop polling after given
minutes.
       Default: 30
    --mfa
       If specified, multi-factor authentication is enabled. The
secret key for multi-factor authentication required for
generating one time password (OTP) needs to be specified in the
properties file.
       Default: false
  * -od, --out-dir
       Output directory where the build artifacts should be
saved.
    -p, --password
       Password for the Kony user. This could be plain text or,
encrypted using 'encrypt' command. This is mandatory.
    -pt, --poll-timeout
       Timeout in seconds to be used for polling build status.
       Default: 30
  * -pp, --props-path
       Path to the .json file which will be used as payload.
  * -sp, --src-path
       Path to the source code zip.
    -u, --user
       Kony user required for authentication, for e.g.
abc@kony.com. This is mandatory.
```

38.4.2 Additional commands supported for Build command

You can perform individual stages of build process by using the following sub-commands of the build command:

```
    build-cancel        Command to cancel build for viz project
    build-download      Command to download artifacts for given build
    build-status        Command to check build status or fetch status
url for given build
    build-trigger       Command to trigger build for viz project
    build-upload        Command to upload viz project for build
```

38.4.2.1 1. build-upload command

To upload artifacts from a local path to a workspace, run the following build command:

```
 java -jar mfcli.jar build-upload -u <user> -p <password> -t <account
id> -e <environment name> -sp <src-code path> -pp <props path>
```

```
 For example:

java -jar mfcli.jar build-upload -u abc@kony.com -p password -t
100054321 -e MyEnv -sp "D:\WorkDir\src.zip" -pp "D:\WorkDir\prop.json"
```

38.4.2.2 2. build-trigger command

To start build binaries, run the following build command:

```
 java -jar mfcli.jar build-trigger -u <user> -p <password> -t <account
id> -e <environment name> -bid <build identifier> -pp <props path>
```

```

For example:

java -jar mfcli.jar build-trigger -u abc@kony.com -p password -t
100054321 -e MyEnv -bid sample-guid -pp "D:\WorkDir\prop.json"
```

38.4.2.3 3. build-download command

After a client binary is uploaded, to download artifacts to your local system, run the following build-download command:

```
 java -jar mfcli.jar build-download -u <user> -p <password> -t
<account id> -e <environment name> -dl <download link> -od <out dir
path> -f <file name> [ --relative ]
```

```
 For example:

java -jar mfcli.jar build-download -u abc@kony.com -p password -t
100054321 -e MyEnv -od "D:\WorkDir\src.zip" -dl "http://example.com/"
-f Demo.apk --relative
```

- The following are additional arguments you must pass with build-download command:

```
  * -dl, --download-link
       Link of the file to be download.
    -e, --environment
       Name of the environment used for building.
  * -f, --file-name
       Name for the file to be saved as, after downloading.
    --mfa
       If specified, multi-factor authentication is enabled. The
secret key for multi-factor authentication required for
generating one time password (OTP) needs to be specified in the
properties file.
       Default: false
  * -od, --out-dir
       Path to the directory where downloaded files will be kept.
    -p, --password
       Password for the Kony user. This could be plain text or,
encrypted using
```

```
       'encrypt' command. This is mandatory.
    --relative
       Flag to indicate, whether download link is relative or
actual.
       Default: false
```

38.4.2.4 build-status command

To view an app build progress during any of the stages, run the following build command:

```
 java -jar mfcli.jar build-status -u <user> -p <password> -t <account
id> -u <user> -p <password> -t <account id> -e <environment name> -bid
<build identifier> -ps -pt <timeout in seconds>
```

```
 For example:
java -jar mfcli.jar build-status -u abc@kony.com -p password -t
100054321 -u abc@kony.com -p password -t 100054321 -e MyEnv -bid
sample-guid -ps -pt 30
```

- The following are additional arguments you must pass with build-status command:

```
 * -bid, --build-identifier
Build identifier for the build obtained via build-upload command.

-e, --environment
Name of the environment which was used for building.

-ps, --poll-status
Flag to poll the status url as well after fetching.
Default: false

-pt, --poll-timeout
Timeout in seconds to be used for polling build status.
Default: 30
```

38.4.2.5 build-cancel command

To cancel an app build process during any of the stages, run the following build command:

```
  java -jar mfcli.jar build-cancel -u <user> -p <password> -t <account
id> -e <environment name> -qid <queue id>
```

```
 For example:
  java -jar mfcli.jar build-cancel -u abc@kony.com -p password -t
100054321 -e MyEnv -qid sample-queue-id
```

- The following are additional arguments you must pass with build-cancel command:

```
 * -qid, --queue-id
Queue identifier for the build obtained via build-trigger
command.
```

38.5 Support for Multi-Factor Authentication from MFCLI

The Multi-Factor Authentication (MFA) feature helps you activate a user account for an added security authentication on Cloud. If you activate MFA for your account, and you use MFCLI, you must provide additional details such as a **secret key** along with other attributes.

38.5.1 How to Enable Multi-Factor Authentication (MFA)

To run MFCLI in MFA mode, you need to pass the `--mfa` argument. If MFA is enabled, you must also provide the secret key for multi-factor authentication, which is required to generate one time password (OTP). You must supply the secret key through a property file.

38.5.1.1 How to configure a secret key in a .properties file

You need to provide the secret key value for the `mfa.secret.key` in a `.properties` file, for example, Sample_mfcli.properties.

```
mfa.secret.key = jrlr dm5t vlze clew b64f cptg fwhs d6f2
```

For more details on secret key for MFA, refer to How to Generate a Secret Key.

38.5.1.2 How to invoke MFCLI in MFA mode

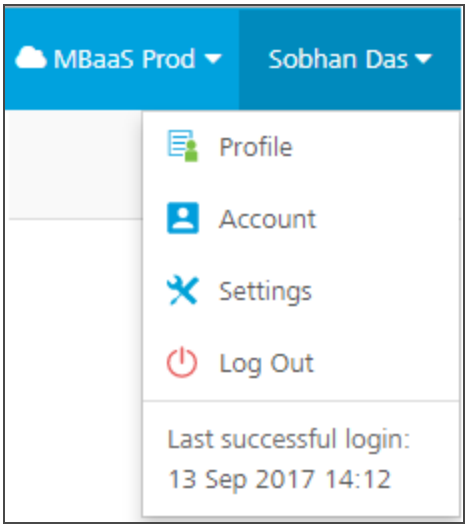
To invoke MFCLI in MFA mode, pass the command as follows:

```
 java -DKONY_MFCLI_PROPERTIES_FILE=\Sample_mfcli.properties -jar
mfcli.jar <command> --mfa .........
```

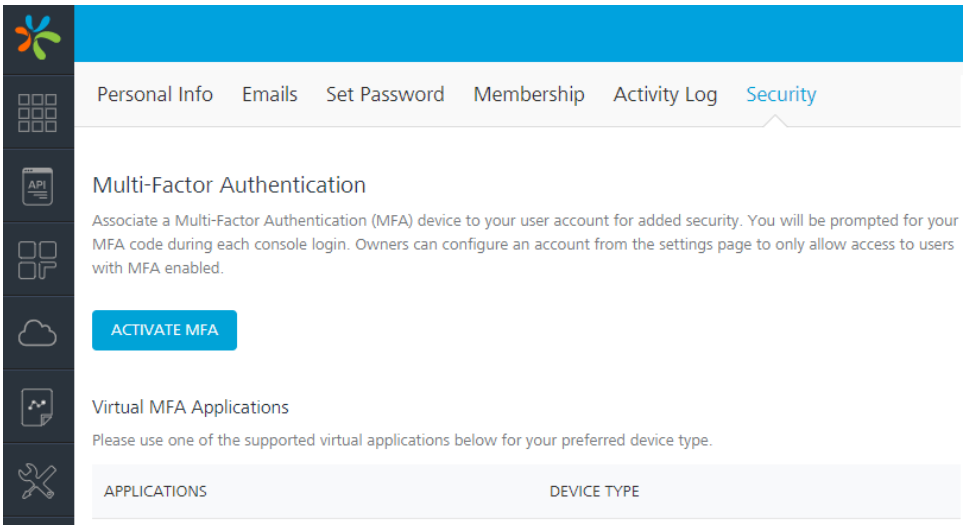
38.5.1.3 How to Generate a Secret Key

To generate a secret key, follow these steps:

1. Click **Profile** in the user account drop-down menu.

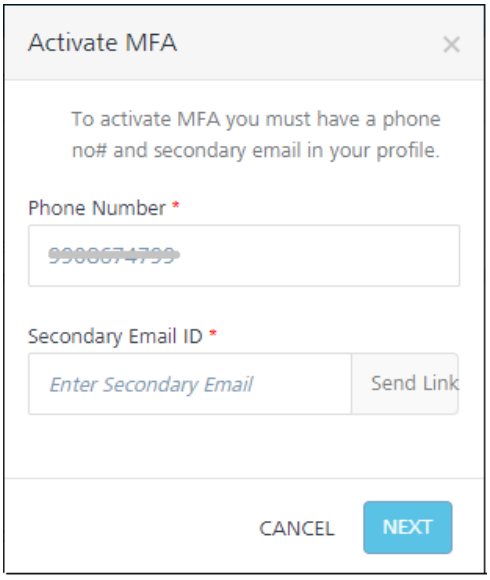


2. Click the **Security** tab.

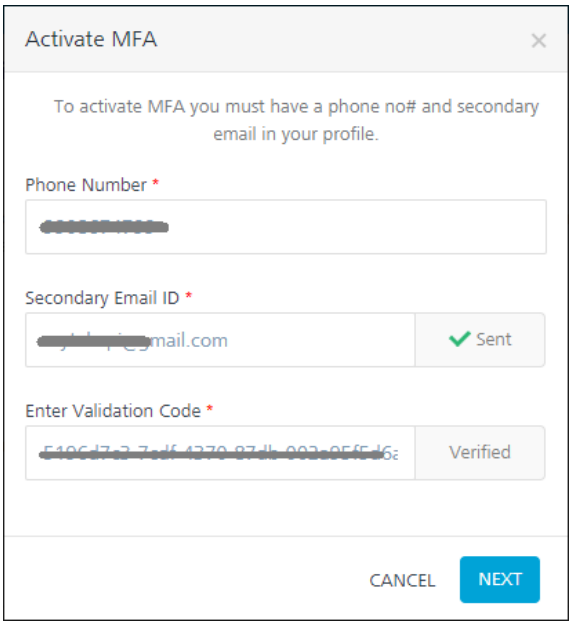


> ***Important:*** The Secret key can only be obtained during registration. If a user is already registered, deactivate the user and then activate the user again to get the secret key.

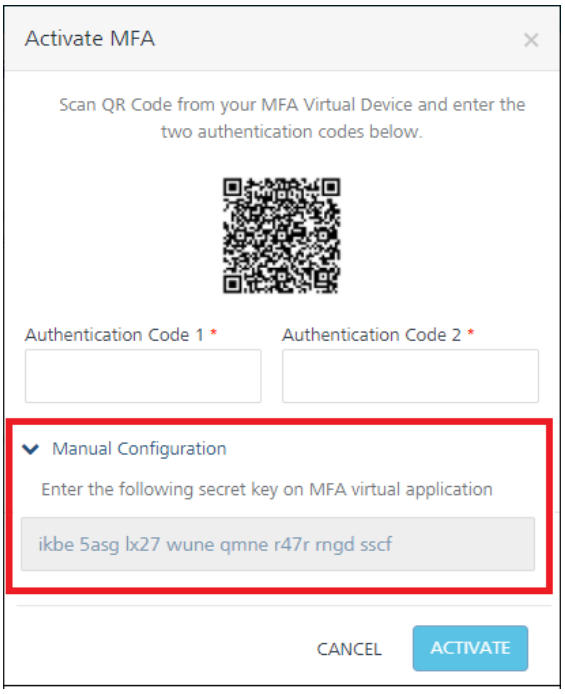
3. Click **ACTIVATE MFA**.



4. In the **Activate MFA** dialog, do the following:
    a. Enter the phone number.
    b. Enter the secondary email and click **Send Link**. An email from Kony Accounts is sent to your registered secondary email ID. The email contains a security code. Also, the **Enter Validation Code** field is enabled under the **Secondary Email ID** field.
    c. In the **Enter Validation Code**, enter the security code and validate the code by clicking the **Verify** button.



d. Once the code is verified, click **NEXT**.



e. Expand the **Manual Configuration**, and then copy the secret key.

f. Update the secret key in the `Sample_mfcli.properties` file.

You must supply the `Sample_mfcli.properties` file as `-d` parameter for MFCLI.

5. Click **ACTIVATE**.

38.6 Export and Import Custom Reports and Custom Dashboards through MFCLI

Export and Import operations of Custom Reports and Custom Dashboards are supported in Kony Fabric On-Premise and Kony Fabric Cloud environments from V8 SP4 onwards.

> ***Note:*** The export and import operations are applicable for only those Custom Reports and Custom Dashboards that are saved in the `shared` folder.

38.6.1 Export Operation

Exporting Custom Reports or Custom Dashboards that are created in an account gives a `.zip` file as an output.

The output of the export operation provides the following information:

- Number of custom reports that were successfully exported.
- Number of custom reports that failed to be exported.

38.6.1.1 Export Custom Reports

You can export Custom Reports from the following environments:

- Kony Fabric Cloud Environment
- Kony Fabric On-Premise Environment

Kony Fabric Cloud Environment

To export custom reports of the standard domain from Kony Fabric Cloud environment, run the following command.

```
 java -jar mfcli.jar export-reports -u <user> -p <password> -t
<account id> -r <directory name>
```

Example

```
 java -jar mfcli.jar export-reports -u abc@kony.com -p abc123 -t
100054321 -r C:\tmp\Sample
```

In this scenario, `abc@kony.com` is the user name with password as `abc123` and `100054321` is the account ID. This command exports the custom reports as `StandardDomainReports_2018_12_28_15_00_10.zip` to `C:\tmp\Sample`.

> ***Note:*** The file name of the .zip file will be a concatenation of `StandardDomainReports` and the time stamp.

To export custom reports of the custom domain from Kony Fabric Cloud environment, run the following command.

```
 java -jar mfcli.jar export-reports -u <user> -p <password> -t
<account id> -a <app name> [-v <app version>] -r <directory name> -e
<environment name>
```

Example

```
 java -jar mfcli.jar export-reports -u abc@kony.com -p abc123 -t
100054321 -a MyApp -v 1.0 -r C:\tmp\Sample -e LocalDevEnv
```

In this scenario, `abc@kony.com` is the user name with password as `abc123` and `100054321` is the account ID. This command exports the custom reports of `MyApp` version `1.0` from `LocalDevEnv` environment as `MyApp_2018_12_28_15_00_10.zip` to `C:\tmp\Sample`.

> ***Note:*** The file name of the .zip file will be a concatenation of the application name and the time stamp.

Kony Fabric On-Premise Environment

To export custom reports of the standard domain from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar export-reports -u <user> -p <password> -au
<Identity URL> -cu <Console URL> -r <directory name>
```

Example

```
 java -jar mfcli.jar export-reports -u abc@kony.com -p abc123 -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -r C:\tmp\Sample
```

In this scenario, `abc@kony.com` is the user name with password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command exports the custom reports as `StandardDomainReports_2018_12_28_15_00_10.zip` to `C:\tmp\Sample`.

> ***Note:*** The file name of the .zip file will be a concatenation of `StandardDomainReports` and the time stamp.

To export custom reports of the custom domain from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar export-reports -u <user> -p <password> -au
<Identity URL> -cu <Console URL>  -a <app name> [ -v <app version> ] -
r <directory name> -e <environment name>
```

Example

```
 java -jar mfcli.jar export-reports -u abc@kony.com -p abc123 -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -a MyApp -v 1.0 -r
C:\tmp\Sample -e LocalDevEnv
```

In this scenario, `abc@kony.com` is the user name with password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command exports the custom reports of `MyApp` version `1.0` as `MyApp_2018_12_28_15_00_10.zip` to `C:\tmp\Sample`.

> ***Note:*** The file name of the .zip file will be a concatenation of the application name and the time stamp.

38.6.1.2 Export Custom Dashboards

You can export Custom Dashboards from the following environments:

- Kony Fabric Cloud Environment
- Kony Fabric On-Premise Environment

Kony Fabric Cloud Environment

To export custom dashboards from Kony Fabric Cloud environment, run the following command.

```
 java -jar mfcli.jar export-dashboards -u <user> -p <password> -t
<account id>  -r <directory name>
```

Example

```
 java -jar mfcli.jar export-dashboards -u abc@kony.com -p abc123 -t
100054321  -r C:\tmp\Sample
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123`, and `100054321` is the account ID. This command exports the custom dashboards as `CustomDashboardReports_2018_12_28_15_00_10.zip` to `C:\tmp\Sample`.

> ***Note:*** The file name of the .zip file will be a concatenation of the `CustomDashboardsReports` and the time stamp.

Kony Fabric On-Premise Environment

To export custom dashboards from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar export-dashboards -u <user> -p <password> -au
<Identity URL> -cu <Console URL>  -r <directory name>
```

Example

```
 java -jar mfcli.jar export-dashboards -u abc@kony.com -p abc123 -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081  -r C:\tmp\Sample
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command exports the custom dashboards as `CustomDashboardReports_2018_12_28_15_00_10.zip` to `C:\tmp\Sample`.

> ***Note:*** The file name of the .zip file will be a concatenation of the `CustomDashboardsReports` and the time stamp.

38.6.2 Import Operation

Import Operation takes the `.zip` file as input, which is the output of the Custom Reports or Custom Dashboards export operation.

The output of the import operation provides the following information:

- Number of custom reports that were successfully imported.
- Number of custom reports that failed to be imported.
- Number of skipped custom reports during import (the reports, that do not belong to the input parameter application).

> ***Important:***
> -The database type of the import and export operations must be same. If the database type differs, the operation will fail. For example, if the custom reports are exported from MySQL database, the custom reports must also be imported to MySQL database.
> -The environment used for the import and export operations must be same. For example, if the custom reports are exported from Kony Fabric Cloud environment, the custom reports must also be imported to Kony Fabric Cloud environment.

38.6.2.1 Prerequisites

You must meet the following prerequisites before importing Custom Reports and Custom Dashboards:

- The application for which the custom reports have to be imported must be published prior to performing the import operation.
- You must create Custom Metrics for an application after the app is published. The **name**, **type**, and **order** of the custom metrics must match the application from which the custom reports were exported.

  For example, if `Application1`, from which the custom reports were exported has the following custom metrics:

  - Metrics1: Long
  - Metrics2: String
  - Metrics3: Boolean
  - Metrics4: String

  Then, before you import the custom reports into `Application2`, create the same custom metrics in `Application2` preserving the **name**, **type**, and **order** as used in `Application1`.
- Import the custom reports respective to the custom dashboards, prior to importing the custom dashboards.

38.6.2.2 Import Custom Reports

You can import Custom Reports from the following environments:

- Kony Fabric Cloud Environment
- Kony Fabric On-Premise Environment

Kony Fabric Cloud Environment

To import custom reports of the standard domain from Kony Fabric Cloud environment, run the following command.

```
 java -jar mfcli.jar import-reports -u <user> -p <password> -t
<account id> -f <file name>
```

Example

```
 java -jar mfcli.jar import-reports -u abc@kony.com -p abc123 -t
100054321 -f C:\sample\MyApp_2018_12_28_15_00_10.zip
```

In this scenario, `abc@kony.com` is the user name with password as `abc123` and `100054321` is the account ID. This command imports the custom reports from `C:\sample\MyApp_2018_12_28_15_00_10.zip`.

To import custom reports of the custom domain from Kony Fabric Cloud environment, run the following command.

```
 java -jar mfcli.jar import-reports -u <user> -p <password> -t
<account id>  -a <app name> [ -v <app version> ] -f <file name> -e
<environment name>
```

Example

```
 java -jar mfcli.jar import-reports -u abc@kony.com -p abc123 -t
100054321  -a MyApp -v 1.0 -f C:\sample\MyApp_2018_12_28_15_00_10.zip
-e LocalDevEnv
```

In this scenario, `abc@kony.com` is the user name with password as `abc123` and `100054321` is the account ID. This command imports the custom reports of `MyApp` version `1.0` from `C:\sample\MyApp_2018_12_28_15_00_10.zip`.

Kony Fabric On-Premise Environment

To import custom reports of the standard domain from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar import-reports -u <user> -p <password> -au
<Identity URL> -cu <Console URL> -f <file name>
```

Example

```
 java -jar mfcli.jar import-reports -u abc@kony.com -p abc123 -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -f
C:\sample\MyApp_2018_12_28_15_00_10.zip
```

In this scenario, `abc@kony.com` is the user name with password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command imports the custom reports from `C:\sample\MyApp_2018_12_28_15_00_10.zip`.

To import custom reports of the custom domain from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar import-reports -u <user> -p <password> -au
<Identity URL> -cu <Console URL>  -a <app name> [ -v <app version> ] -
f <file name> -e <environment name>
```

Example

```
 java -jar mfcli.jar import-reports -u abc@kony.com -p abc123 -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081  -a MyApp -v 1.0 -
f C:\sample\MyApp_2018_12_28_15_00_10.zip -e LocalDevEnv
```

In this scenario, `abc@kony.com` is the user name with password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command imports the custom reports of `MyApp` version `1.0` from `C:\sample\MyApp_2018_12_28_15_00_10.zip`.

38.6.2.3 Import Custom Dashboards

You can import Custom Dashboards from the following environments:

- Kony Fabric Cloud Environment
- Kony Fabric On-Premise Environment

Kony Fabric Cloud Environment

To import custom dashboards from Kony Fabric Cloud environment, run the following command.

```
 java -jar mfcli.jar import-dashboards -u <user> -p <password> -t
<account id>  -f <file name>
```

Example

```
 java -jar fcli.jar import-dashboards -u abc@kony.com -p abc123 -t
100054321  -f C:\sample\CustomDashboardReports_2018_12_28_15_00_10.zip
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123` and `100054321` is the account ID. This command imports the custom dashboards from `C:\sample\CustomDashboardReports_2018_12_28_15_00_10.zip`.

Kony Fabric On-Premise Environment

To import custom dashboards from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar import-dashboards -u <user> -p <password> -au
<Identity URL> -cu <Console URL>  -f <file name>
```

Example

```
 java -jar mfcli.jar import-dashboards -u abc@kony.com -p abc123 -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081  -f
C:\sample\CustomDashboardReports_2018_12_28_15_00_10.zip
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command imports the custom dashboards from `C:\sample\CustomDashboardReports_2018_12_28_15_00_10.zip`.

38.7 Export and Import Configurable Parameters for App Services through MFCLI

Configurable Parameters provide an interface to define a set of key-value pairs at the server and the client level. You can access the configured server and client properties from the custom code. The configured properties are available to custom code such as preprocessor, postprocessor and, Java services at run time. Any updates made to the configured properties are reflected in the custom code.

> ***Note:*** For more information on how to configure Server Properties and Client App Properties in the App Services Console, refer to Configurable Parameters.

You can export and import your configuration parameters from one environment to another environment. When you export configuration parameters, the App Services Console downloads a .zip file or folder, which contains the parameters list in two .CSV files such as `clientAppProperties.csv` and `serverProperties.csv`. Each of these CSV files contains the key-value pairs list of the configurable parameters. Only by using MFCLI command, you can export or import configuration parameters in a folder format. You will use the .ZIP file or a folder to import the configurations parameter into another environment.

> ***Note:*** If the name of the keys in the imported file matches with the existing names, you will see a conflict message while importing with the list of keys that have the conflict. Click **Upload to overwrite the existing keys and values with the new keys and values** or click **Cancel to stop the upload**.

> ***Note:*** If the imported file contains same key name with different key values, the last key value takes precedence for that key name.

You can export and import your configuration parameters from one environment to another by either of the two methods:

- **By using the App Services Console.** For more information, refer to the **Export the key-value pair list** and **Import the key-value pair list** sections in the App Services Console > Configurable Parameters.
- **By using the MFCLI commands for continuous integration**. This support is available for Kony Fabric On-Premise and Kony Fabric Cloud environments from V8 SP4 onwards.
  The following sections detail how to export and import Configurable Parameters by using MFCLI commands.

38.7.1 Export Operation for Configurable Parameters - App Services

The `export-config-properties` command exports the configurable parameters configured in App Services Console in a specified .zip file or directory.

38.7.1.1 Export Configurable Parameters - App Services

You can export configurable parameters from the following environments:

- Export Configurable Parameters from Kony Fabric Cloud Environment
- Export Configurable Parameters from Kony Fabric On-Premise Environment

Export Configurable Parameters from Kony Fabric Cloud Environment

To export configurable parameters from Kony Fabric Cloud environment, run the following command.

```
java -jar mfcli.jar export-config-properties -u <user> -p <password> -
t <account id> [-f <file name> | -r <directory name>] -e
<environmentName>
```

> ***Note:*** To export the Configurable Parameters to a zip file, use the `[-f "zipfilename.zip]` parameter in the command.
>
> To export the Configurable Parameters to directory, use the `[-r <directory name>]` parameter in the command.

Example

```
java -jar mfcli.jar export-config-properties -u abc@kony.com -p
password -t 100054321 -f "C:\\tmp\\Sample.zip" -e "TestEnv"
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123` and `100054321` is the account ID. This command exports the configurable parameters to `C:\\tmp\\Sample.zip`.

Export Configurable Parameters from Kony Fabric On-Premise Environment

To export configurable Parameters from Kony Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar export-config-properties -u <user> -p <password>
-au <Identity URL> -cu <Console URL> [-f <file name> | -r <directory
name>] -e <environmentName>
```

> ***Note:*** To export the Configurable Parameters to a zip file, use the `[-f "zipfilename.zip]` parameter in the command.
>
> To export the Configurable Parameters to directory, use the `[-r <directory name>]` parameter in the command.

Example

```
java -jar mfcli.jar export-config-properties -u abc@kony.com -p
password -au http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -f
"C:\\tmp\\Sample.zip" -e "TestEnv"
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command exports the configurable parameters to `C:\tmp\Sample.zip`.

38.7.2 Import Operation for Configurable Parameters - App Services

The `import-config-properties` command takes a `ZIP` file as input to import the configurable parameters from the `ZIP` to the App Services Console. A `ZIP` file is an output of the `export-config-properties` command.

38.7.2.1 Import Configurable Parameters - App Services

You can import Configurable Parameters from the following environments:

- Import Configurable Parameters to Kony Fabric Cloud Environment
- Import Configurable Parameters to Kony Fabric On-Premise Environment
- Delete Configurable Parameters from Kony Fabric Environment

Import Configurable Parameters to Kony Fabric Cloud Environment

To import Configurable Parameters to Kony Fabric Cloud environment, run the following command.

```
java -jar mfcli.jar import-config-properties -u <user> -p <password> -t <account id> [-f <file name> | -r <directory name>]  -e <environmentName>
```

> ***Note:*** To import the Configurable Parameters from a zip file, use the `[-f "zipfilename.zip]` parameter in the command.
>
> To import the Configurable Parameters from a directory, use the `[-r <directory name>]` parameter in the command.

Example

```
java -jar mfcli.jar import-config-properties -u abc@kony.com -p abc123 -t 100054321 -f "C:\\tmp\\Sample.zip" -e "TestEnv"
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123` and `100054321` is the account ID. This command imports the Configurable Parameters from `C:\\tmp\\Sample.zip` to the environment.

Import Configurable Parameters to Kony Fabric On-Premise Environment

To import Configurable Parameters to Kony Fabric On-Premise environment, run the following command.

```
java -jar mfcli.jar import-config-properties -u <user> -p <password> -au <Identity URL> -cu <Console URL> [-f <file name> | -r <directory name>] -e <environmentName>
```

> ***Note:*** To import the Configurable Parameters from a zip file, use the `[-f "zipfilename.zip]` parameter in the command.
>
> To import the Configurable Parameters from a directory, use the `[-r <directory name>]` parameter in the command.

Example

```
java -jar mfcli.jar import-config-properties -u abc@kony.com -p abc123
-au http://10.10.25.18:8080 -cu http://10.10.25.18:8081 -f
"C:\\tmp\\Sample.zip" -e "TestEnv"
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command imports the Configurable Parameters from `C:\\tmp\\Sample.zip` to the environment.

38.7.3 Delete Operation for Configurable Parameters - App Services

Delete Configurable Parameters from Kony Fabric Cloud Environment

To delete Configurable Parameters from Kony Fabric Cloud environment, run the following command.

```
java -jar mfcli.jar import-config-properties -u <user> -p <password> -
t <account id> [-f <file name> | -r <directory name> -e
<environmentName> -delete
```

Example

```
java -jar mfcli.jar import-config-properties -u abc@kony.com -p abc123
-t 100054321 -f "C:\\tmp\\Sample.zip" -e "TestEnv" -delete
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123` and `100054321` is the account ID. This command deletes the Configurable Parameters specified in the `C:\\tmp\\Sample.zip` file from the "TestEnv" environment.

Delete Configurable Parameters from Kony Fabric On-Premise Environment

To delete Configurable Parameters from Kony Fabric On-Premise environment, run the following command.

```
java -jar mfcli.jar import-config-properties -u <user> -p <password> -
au <Identity URL> -cu <Console URL> [-f <file name> | -r <directory
name>] -e <environmentName> -delete
```

Example

```
java -jar mfcli.jar import-config-properties -u abc@kony.com -p abc123
-au http://10.10.25.18:8080 -cu http://10.10.25.18:8081 -f
"C:\\tmp\\Sample.zip" -e "TestEnv" -delete
```

In this scenario, `abc@kony.com` is the user name with the password as `abc123`. The Identity (Auth) Services and Console are running on `http://10.10.25.18:8080` and `http://10.10.25.18:8081`. This command deletes the Configurable Parameters specified in the `C:\\tmp\\Sample.zip` file from the "TestEnv" environment.

38.8 Configuring Read-only Fields for Object Services through MFCLI

You can configure the fields in a data model of Object Services as read-only fields. MFCLI is enhanced to support locking of fields in an object service attached to an app. So that, the locked fields (base fields) cannot be modified by the other Fabric Cloud users. In this case, other Cloud users can modify only the custom fields.

The MFCLI provides two new commands `lock-config` (to lock fields) and `unlock-config` (to unlock fields). The `lock-config` command must always be attached with a supportive JSON configuration file, which includes configurations/rules/conditions to lock fields of Objects services.

38.8.1 Use Case

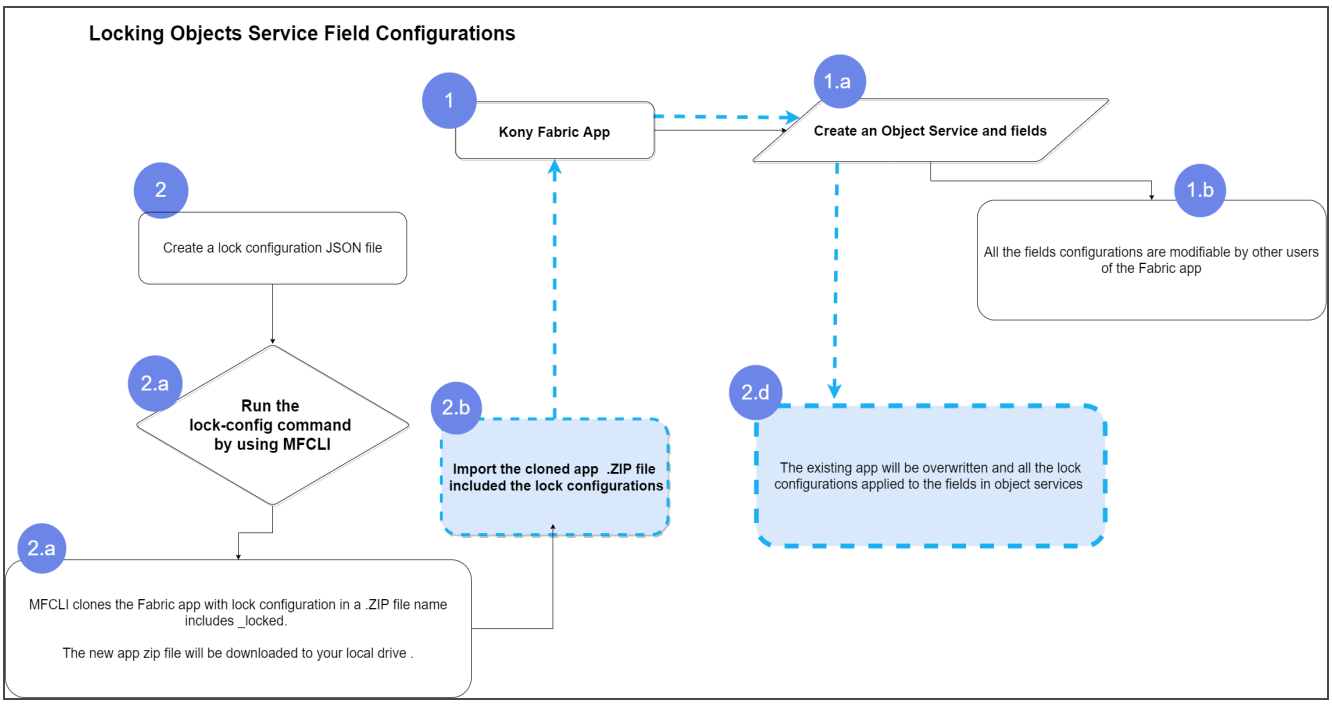
When customers have a large number of fields in their database, including generic and special fields, the maintenance of these fields becomes an issue. The issue becomes more complicated when multiple stakeholders work on the same set of fields. To improve the upgradability and maintainability of Fabric apps (including the Objects Services), the fields can be split into two categories: read-only fields (base fields) and read-write fields (custom fields). Only the selected fields (custom fields) can be modified by other stakeholders. To achieve this scenario, you can use the lock config command of MFCLI.

38.8.2 How Locking Fields Works

After you create your data model for an object service in an app in Fabric Console, the fields of the service can be modified when the app is accessed by other Fabric users. If you want to make any of these fields read-only, you can lock these fields by using the `lock-config` command of MFCLI. This command must include a valid `lockconfig.json` file. A lockconfig.json file must be defined with the required fields of an object that need to be locked. After you run the `lock-config` command successfully, the Fabric Console downloads a zip file to your original app file location. The name of the locked app file name is `<OriginalFabricAppName>_locked.zip`. The new zip file contains the locked fields of objects.

Use the locked zip file to import the locked fields into your existing app. When you import the locked zip file, the existing app is overwritten. The locked fields of your objects are categorized as the **BASE FIELDS** and the rest of the fields in the objects are categorized as **CUSTOM FIELDS**.

Refer to the following diagram that explains how to lock fields of object services as read-only `(2.d)` fields:



38.8.3 Prerequisites

• Quantum Fabric Console Version V9 GA
• Fabric App with Objects Services
• MFCLI version <version number> onwards

- LockConfigTemplate.json file
- Structure of Lock Configuration JSON file

```
LoockConfigTemplate.json
{
    "serviceLockConfigs": {
        "objectServices": {
            "<object_service_name>": {
                "version": "<version>",
                "paths": [
                    "objects/<object_name>/fields/<field_name>",
                    "objects/<object_name>/fields/*",
                    "objects/<object_name>/fields",
                    {
                        "path": "objects/<object_name>/fields",
                        "values": [
                            "<field1_name>",
                            "<field2_name>"
                        ]
                    }
                ]
            },
            "<object_service2_name>": {},
            "<object_service3_name>": {}
        }
    }
}
```

- `"objects/<object_name>/fields"` = The path marks all the existing fields as read-only and no new fields can be created.
- `" objects/<object_name>/fields/*"` = This path with an `asterisk (*)` marks all the existing fields in an object as **read-only** but new fields can be created.
- `"objects/<object_name>/fields/<field_name>"` = The path marks only the specified field as read-only.
    - You can also provide a path and array of specific values to mark as read-only, as follows:

```
                    {
```

38.8.4 Lock Fields Operation

The `lock-config` command locks the fields of objects defined in the configuration JSON file (for example SampleLockConfig.json)

You can export configurable parameters from the following environments:

- Locking fields of objects from Fabric Cloud Environment
- Locking fields of objects from Fabric On-Premise Environment

38.8.4.1 Locking Fields of Objects from Fabric Cloud Environment

To Lock fields of objects from Fabric Cloud environment, run the following command.

```
java -jar mfcli.jar lock-config -u <user> -p <password> -t <account
id> -f <lock config file path> -a <app path> -la <locked app name>
```

Example

```
java -jar mfcli.jar lock-config -u abc@kony.com -p password -t
100054321 -f ~/../lockConfig.json -a ~/../App.zip -la my_locked_app
```

In this scenario, `lockCofing.JSON` is the lock configuration file and `App.zip` is the Fabric app. This command locks the fields of objects services in the App.zip based on the JSON file, and creates a clone of the provided Fabric app with lock configuration applied. For example, `C:\\tmp\\App_locked.zip` file created in the same Fabric app path.

> ***Note:*** If you want to export the locked fields with a different app name, use the `[-la "NewAppName.zip]` parameter in the command. This is an optional.

38.8.4.2 Locking Fields of Objects from Fabric On-Premise Environment

To Lock fields of Objects from Fabric On-Premise environment, run the following command.

```
 java -jar mfcli.jar lock-config -u <user> -p <password> -au <Identity
URL> -cu <Console URL> -f <lock config file path> -a <app path> -la
<locked app name>
```

> ***Note:*** If you want to export the locked fields with a different app name, use the `[-la NameAppName.zip]` parameter in the command. This is an optional.

Example

```
java -jar mfcli.jar lock-config -u abc@kony.com -p password -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -f
~/../lockConfig.json -a ~/../App.zip -la my_locked_app
```

In this scenario, `lockCofing.JSON` is the lock configuration file and `App.zip` is the Fabric app. This command locks the fields of objects services in the App.zip based on the JSON file, and creates a clone of the provided Fabric app with lock configuration applied. For example, `C:\\tmp\\App_locked.zip` file created in the same Fabric app path.

> ***Note:*** The locked fields are exported in a .zip file to the original app file location in your system. The new zip file will be a concatenation of the application name and the `_locked` extension.

38.8.5 Removing all Locked Fields of Object Services

Command to remove all lock configurations from the provided App zip creates a clone of the provided app with locks removed.

The `unlock-config` command removes the write-protection from the fields and objects based on the locked zip file.

You can unlock fields and objects from the following environments:

- Unlock Fields and Objects from Fabric Cloud Environment
- Unlock Fields and Objects from Fabric On-Premise Environment

38.8.5.1 Remove locked configurations of Objects Services from Fabric Cloud Environment

To remove the locked configurations of object services from Fabric Cloud environment, run the following command.

```
java -jar mfcli.jar unlock-config -u <user> -p <password> -t <account
id> -a <app name> -ua <unlocked app name>
```

Example

```
java -jar mfcli.jar unlock-config -u abc@kony.com -p password -t
100054321 -a ~/../App.zip -ua my_unlocked_app
```

In this scenario, the following are the mandatory steps:

- The `-a` is the path of the Fabric app with lock configurations. This is a mandatory parameter.

This command removes all locked configurations for the fields of objects services in the App.zip, and creates a clone of the provided Fabric app with lock configuration applied. For example, `C:\\tmp\\App_locked.zip` file created in the same Fabric app path.

> ***Note:*** The `-ua (--unlocked-app)`, is the name of the new app zip with lock config removed. for example, AppName_unlocked. This is an optional.

38.8.5.2 Remove locked configurations of Objects Services from Fabric On-Premise Environment

To remove the locked configurations of object services from Fabric On-Premise environment, run the following command.

```
java -jar mfcli.jar unlock-config -u <user> -p <password> -au
<Identity URL> -cu <Console URL> -a <app name> -ua <unlocked app name>
```

Example

```
java -jar mfcli.jar unlock-config -u abc@kony.com -p password -au
http://10.10.24.79:8080 -cu http://10.10.24.78:8081 -a ~/../App.zip -
ua my_unlocked_app
```

In this scenario, the following parameters are mandatory:

- The `-a` is the path of the Fabric app with lock configurations.

This command removes all locked configurations for the fields of objects services in the App.zip, and creates a clone of the provided Fabric app with lock configuration applied. For example, `C:\\tmp\\App_locked.zip` file is created in the same Fabric app path.

38.9 MFCLI command to Merge Exported Definitions

The MFCLI tool is enhanced to enable importing and merging application templates into the base application functionality. Option to merge template service zip package with the base service zip package is provided in Kony Fabric console.

Use the following command to merge the service packages. An output zip file (merged package from the base service package and the template service package) is created.

```
 java -jar mfcli.jar merge-service-zip --base-service-package-path
<file path> --template-service-package-path <file path> --output-
directory <directory path> [ --output-package-name <package name> ]  [
--override-existing ]
```

`baseServicePackagePath` - Path of the zip file that points to the base service zip.

`templateServicePackagePath` - Path of the zip file that points to the template service zip.

`outputServicePackagePath` - Path where the output zip is created if the merge is successful.

`output-package-name` - Used to name the merged zip file.

> ***Note:*** A random name is generated if the name is not provided.

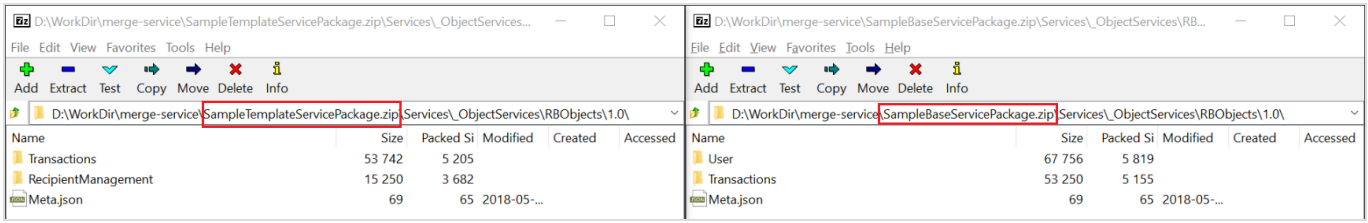
`override-existing` - If the parameter is provided in the command structure, the output file with the specified file name overwrites the existing file name (if the file exists). Else, it displays an error.

During execution of the merge process, you can view the list of conflicts and set of specific services from both the packages. In case of a conflict due to service being available in both the base and template, the version from the template package overwrites the base version.

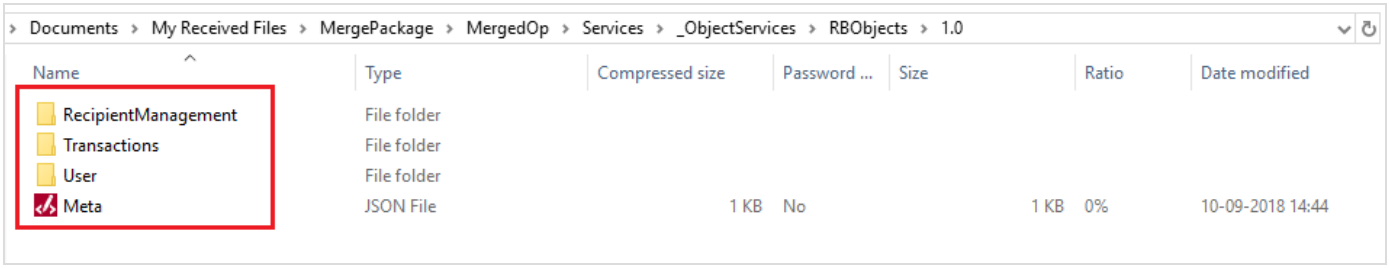
```
D:\WorkDir\merge-service>java -jar mfcli.jar merge-service-zip --base-service-package-path SampleBaseServicePackage.zip --template-service-package-path SampleTemplateServic
ePackage.zip --output-directory ./
Merging service package from base zip and template zip...
Copying file from template zip: Services/_Identity/
Copying file from template zip: Services/_Identity/conflictingIdentity2/
Copying file from template zip: Services/_Identity/conflictingIdentity2/Meta.json
Copying file from base zip: Services/_Identity/newidentity/
Copying file from base zip: Services/_Identity/newidentity/Meta.json
Copying file from base zip: Services/_Identity/nonconflicting1/
Copying file from base zip: Services/_Identity/nonconflicting1/Meta.json
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Endpoints/default.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Fiserv.wsdl
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getPFMTrasactions.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getRecipient.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getPayOffBalance.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getAccountsYTD.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getScheduledTransactions.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getScheduledTransactionByCIF.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/addRecipient.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getMemberTransactions.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/updateAccount.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getRecentTransactions.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getLoanAccountDetails.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/doRepetitiveTransfer.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/doTransfer.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getPersonalDetails.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getPendingTransactions.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getTaxIdData.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/deleteRecipient.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getTransactions.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getAccountsNonloggedin.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/getAccounts.xml
Copying file from template zip: Services/_Integration/FiservDNAIntServices/1.0/Operations/cancelScheduledTransfer.xml
Copying file from template zip: Services/_Orchestration/FiservDNAOrchServices/1.0/getUserDetails.xml
Copying file from template zip: Services/_Orchestration/FiservDNAOrchServices/1.0/createTransfer.xml
Copying file from template zip: Services/_Orchestration/FiservDNAOrchServices/1.0/getScheduledTransactions.xml
Copying file from template zip: Services/_Orchestration/FiservDNAOrchServices/1.0/Meta.json
Copying file from template zip: Services/_ObjectServices/RBObjects/1.0/Transactions/Operations/getMapping.json
Copying file from template zip: Services/_ObjectServices/RBObjects/1.0/Transactions/Operations/createMapping.json
```

After the merge is successful, an output service package is created with services from both the base and template packages.

Base and Template package files before merging



Output package created after merging is successful



39. Kony SDKs

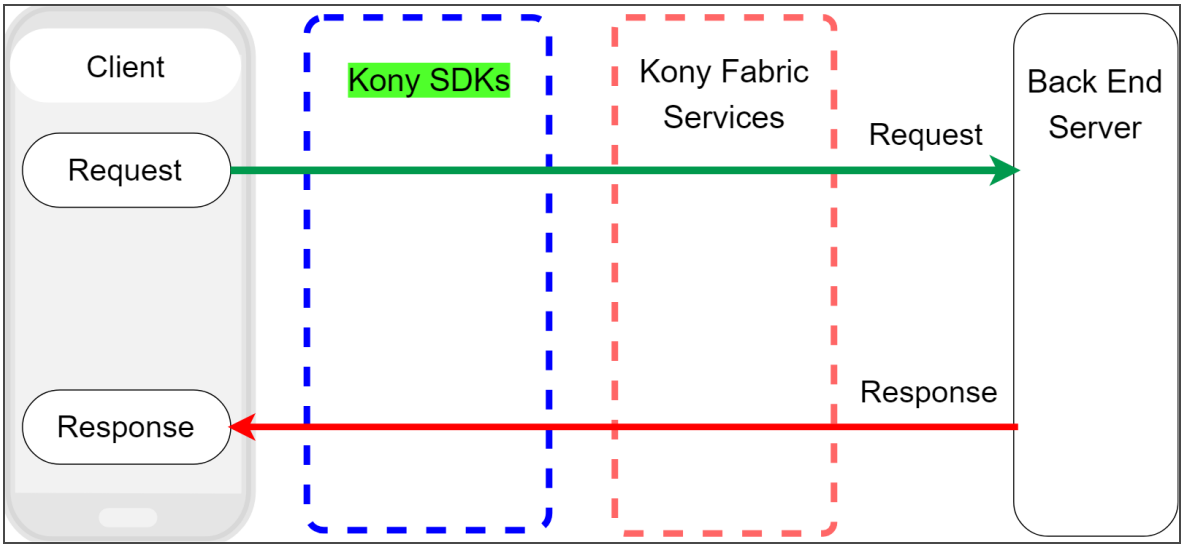
SDKs integrate Kony Fabric services with your client applications. You can use your preferred development language or platform to build your apps.

Before using Kony SDKs, Kony Fabric app must be published to an environment.

> ***Note:*** Kony provides different client SDKs to connect applications to Kony Fabric services. The SDKs are available for Native iOS, Native Android, .NET, Cordova (PhoneGap), JavaScript, and the SDK built into the Kony Visualizer IDE.
>
> Due to differences in the various platforms, the functionality varies for each client SDK. Please refer to the documentation of specific SDKs to view the features supported by each of them and how to use them.

39.1 Workflow of Kony SDKs

The following workflow describes the integration of Kony SDKs layer along with various stages of client app development.



39.2 Supported Kony SDKs

Kony Fabric client SDKs are supported for the following languages.

Kony SDKs			
SDKs	**Description**	**Prerequisites**	**Documentation**
Kony Visualizer	Kony Visualizer SDK (`kony-ide-sdk`) is bundled with Visualizer. Kony Visualizer SDK help you to integrate Kony Fabric services into client apps for multiple platforms	• Kony Visualizer 7.0 and above.	
JavaScript	Plain JS SDK (`kony-plainJS-sdk`) helps you to integrate Kony Fabric services into client apps developed with Vanilla JS	Supported Web browsers for Plain JS: • Firefox 31.0 • Google Chrome 36.0 • Internet Explorer 10.1 • Opera 23.0	

Kony SDKs			
SDKs	**Description**	**Prerequisites**	**Documentation**
PhoneGap (Cordova)	Kony PhoneGap SDK (`kony-phonegap-sdk`) helps you to integrate Kony Fabric services into client apps developed with PhoneGap (Cordova)	• Cordova 3.4.0-rc.2	
iOS	Kony iOS SDK (`kony-ios-sdk`) helps you to integrate Kony Fabric services into client apps developed with Xcode	• Minimum iOS version: iOS7 • Mavericks OS X or Mountain Lion OS X • Xcode 5.1	
Android	Kony Android SDK (`kony-android-sdk`) helps you to integrate Kony Fabric services into client apps developed with Android Studio	• Minimum Android version: Android 4.0 (API Level 14) • Java Development Kit (JDK) 1.6 - Oracle Technology NetworkJavaJava SE SupportDownloads > Java SE 6 Downloads • An IDE, such as Eclipse, with ADT installed or Android Studio 1.5 or 2.1.	

Kony SDKs			
SDKs	**Description**	**Prerequisites**	**Documentation**
.NET (Visual Studio)	Kony .NET (Visual Studio) SDK (`kony-windows-sdk`) helps you to integrate Kony Fabric services into client apps developed with .Net (Windows)	• .NET Framework 4.6.1 or higher • NuGet Packages: ◦ Acr.DeviceInfo 6.5.0 or higher ◦ Newtonsoft.Json 12.0.1 or higher ***Note:*** Kony does not explicitly test or certify the .NET SDK with Xamarin, but it is designed to be compatible with Xamarin.	SDK .NET (Visual Studio)

39.3 Kony Visualizer SDK

These steps show how to add the Kony-IDE-SDK to your project and set up Kony Fabric Client.

- Prerequisites

  Kony Visualizer 7.0 and higher versions.

- Downloading Kony IDE SDK
- Initializing the Kony SDK Client SDK
- Setting User ID
- HTTP Message Body Integrity

- Server Event APIs
- Invoking an Identity Service
- Invoking an Integration Service
- Invoking an Integration Service with Passthrough
- Invoking a Configuration Service
- Invoking a Logic Service
- Invoking a Messaging Service
- Invoking a Metrics Service
- Invoking Offline Objects APIs

  Offline objects is a new capability of Object Services in Kony V8 that provides a simplified approach to synchronizing data to a client app for offline access. The APIs can be used at different levels in your applications. All Offline Objects API's have KNYMobileFabric as a namespace.
  For more information, see Kony Offline Objects API Reference Guide.

- Invoking Sync APIs

  For information on sync APIs, refer Sync Framework Documentation.

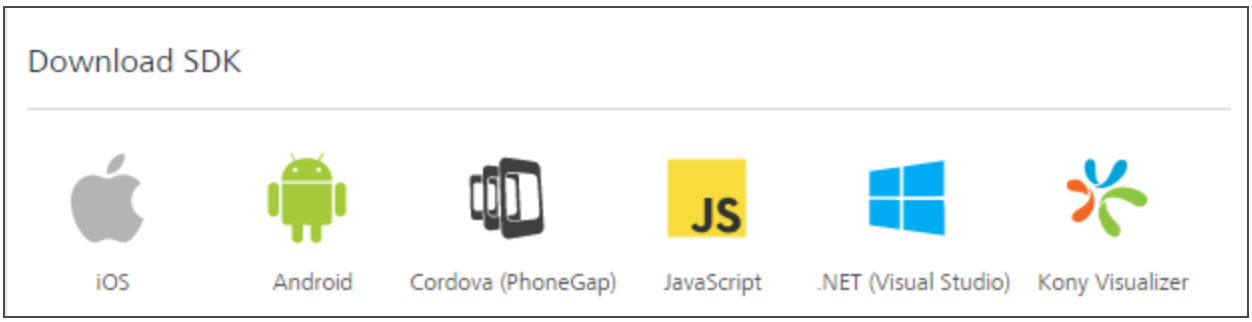
- Invoking an Object Service
- Cache Service Response for Integration and Object Service
- Using Log SDK
- Binary APIs
- API Reference

  To view the API Reference for Kony JS, click Kony IDE docset.

39.3.1 Downloading Kony IDE SDK Files

The Kony SDK is now included as part of the Kony development tools and the SDK version is managed using the update feature within the software.

If you need to install the Kony development tools, please download the updates from Kony downloads site.



39.3.2 Initializing the Kony JS Client SDK

The initialization method fetches the Kony Fabric configuration and saves it in the cache. The application uses the cached configuration. It is a synchronous call.

You can initialize the Kony JS Client SDK in the following ways:

- Linking your Fabric application with Kony Visualizer.
- Manual initialization through code.
- Manual initialization by using setClientParams.

39.3.2.1 Link Fabric Application with Kony Visualizer

To initialize the Kony JS Client SDK, link your Kony Fabric Application with Kony Visualizer by following these steps.

1. Open Kony Visualizer Enterprise.
2. Click **Login** on the upper right corner and sign in to your Kony Fabric account.

3. Go to **Project** > **Settings** > **Fabric**. Select a Could Account and Environment from the **Fabric Details** section and click **Done**.
4. Now, Click on the DATA & SERVICES tab (top right), click on More options, and select **Link to Existing App**. The **Fabric Applications** screen appears.
5. Click **Associate** against the Fabric Apps that you want to link to Quantum Visualizer.

> ***Important:*** Note that the default versions of the Fabric app are shown in the above screen-shot and the same is linked for auto init. Please select the version other than default version if you want to have app associated with another Fabric app's version.

After you link your Fabric application with Visualizer, the Kony Fabric SDK is initialized automatically when the app starts.

Note:

- The initialized SDK is available for use within the application with the variable name **KNYMobileFabric**.
- If the operating system launches an application in the background, the session registers as a Non-Interactive session.

  For example,

  - On Push Notifications
  - Content Provider Calls
  - Registering for Geo-location Updates.
- If the application launches for interaction, the session registers as an Interactive session.

  For example,

- When a user launches the application.
- When another app or component launches the application.

39.3.2.2 Manual Initialization Through Code

To initialize the Kony JS Client SDK, run the following code:

```
//Sample code to initialize [[[Undefined variable
MyVariables.Quantum]]] Fabric Client

var appkey = 'your-app-key';
var appsecret = 'your-app-secret';
var serviceUrl = 'your-service-url';
var initOptions = {'MFAppVersion': '<your-Fabric-app-version>'};
//This is an optional argument to be used if you want to switch to
Fabric-app version other than default version.
// Get an instance of SDK
var client = new kony.sdk();
// initialize SDK
client.init(appkey, appsecret, serviceUrl, function(response) {
   kony.print('Init success: ' + JSON.stringify(response));
}, function(error) {
   kony.print('Init failed: ' + JSON.stringify(error));
}, initOptions);
```

39.3.2.3 Manual Initialization by using setClientParams

You can also initialize the Kony JS Client SDK by using setClient parameters. However, a few data types are not captured in the metrics of the app.

Setting clientParams in the manual init by using setClientParams API allows Kony Fabric SDK to send client parameters such as application ID and application name to the parameters set from Kony Visualizer. It also allows these parameters to be consistent across different data collection points such as integration services, application events, and custom metrics.

To initialize the Kony JS Client SDK, run the following code:

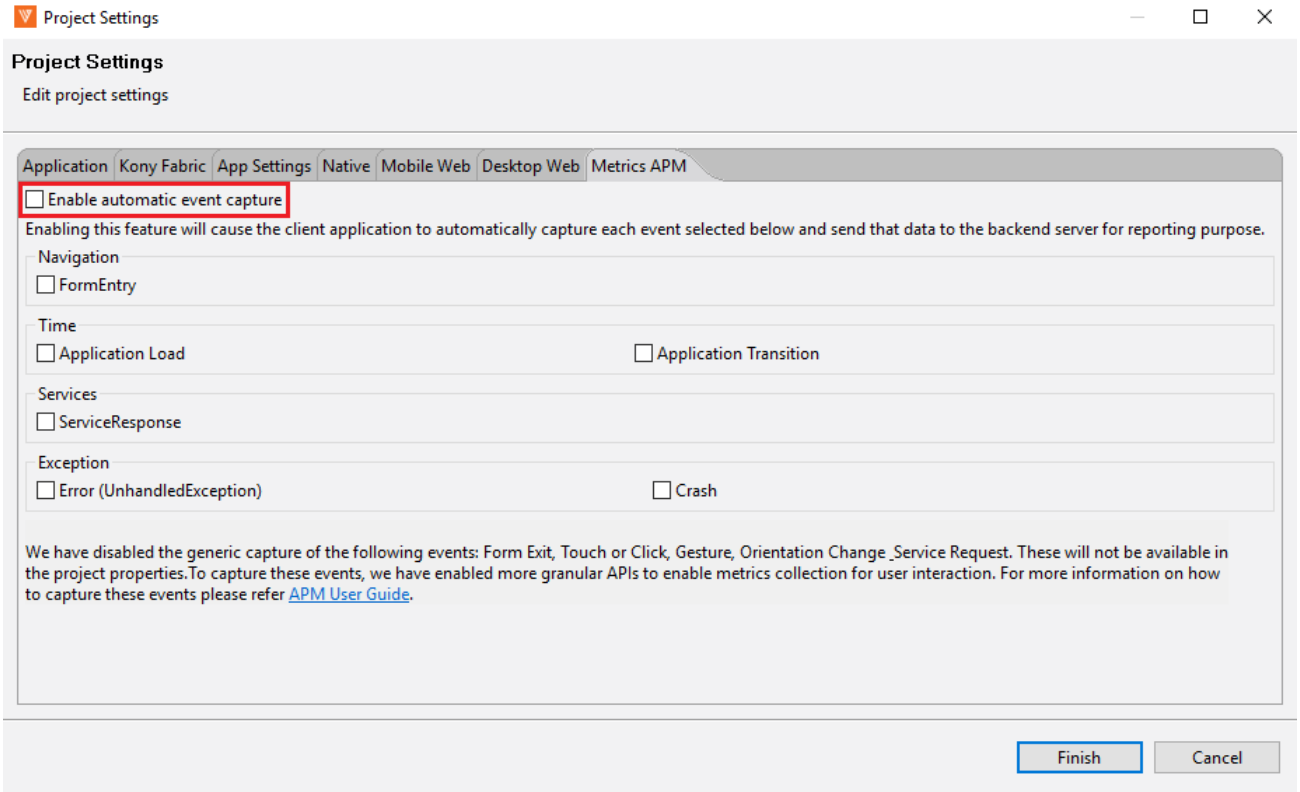
```
//Sample code to initialize Kony Fabric Client manually
// Get an instance of SDK
// Assume SDK already initialized
var client = kony.sdk.getCurrentInstance();
var clientParams = {};
// appConfig is a global variable that holds the application's
configuration.
clientParams.aid = appConfig.appId;
clientParams.aname = appConfig.appName;
client.setClientParams(clientParams);
```

39.3.2.4 Automatic Event Capture

This feature enables the client application to automatically capture data of the selected events and send it to the server for reporting purpose.

To enable Automatic Event Capture for your Fabric application, follow these steps.

1. Go to **File** > **Settings**.
2. Under Settings, in the **Metrics APM** tab, select the **Enable automatic event capture** check box.



3. Now, select the events of which you want to capture the data.
4. Click **Finish** to save your changes.

When an application starts, Kony SDK registers a session and sends its information to the Kony Fabric Server. If the device is offline, or the server is not reachable, the session's information persists on the device until it can successfully send the information to the Kony Fabric Server.

For more information on application session, refer to Standard Reports and Dashboard Guide.

This feature is disabled for the following events:

- Form Exit
- Touch or Click
- Gesture

- Orientation Change
- Service Request

For information about how to capture the mentioned events, refer to APM User Guide.

39.3.2.5 Configuring Global Parameters

The global parameter APIs helps you to set, remove, and edit global header params, query params, and body params in order to send the subsequent SDK network calls.

Enum globalRequestParamType

The globalRequestParamType enum specifies the following param types:

- headers
- query params
- body params

getGlobalRequestParams

The getGlobalRequestParams method returns the global params of a specific type that have been set.

Syntax

```
getGlobalRequestParams(paramType);
```

Parameters

Name	Type	Description
paramType	globalRequestParamType	The type of the param.

Return Value

This method returns a list of all network params in the following format:

```
({<ParamName>, <ParamValue>, <ParamType>},{})
```

The param type can be either header, post param, or body.

Example

```
// This example returns all global header params.
// Get an instance of SDK
// Assume SDK already initialized
var client = kony.sdk.getCurrentInstance();
client.getGlobalRequestParams(client.globalRequestParamType.headers);
```

setGlobalRequestParam

The setGlobalRequestParam method adds a param with the specified name, value, and type, to the network calls. If a param with the same name already exists, this API overrides that param. If the same param name is manually added to a network call, the local param will have a higher priority than the global param. After a param is added, it is sent globally to all network calls from licensing and the SDK.

Syntax

```
setGlobalRequestParam(paramName, paramValue, paramType);
```

Parameters

Name	Type	Description
paramName	String	Name of the parameter
paramValue	String	Value of the parameter
paramType	globalRequestParamType	The type of the parameter

Example

```
// This example sets a global header with header name testHeader and
value testValue.
// Get an instance of SDK
// Assume SDK already initialized
var client = kony.sdk.getCurrentInstance();
```

```
client.setGlobalRequestParam("testHeader", "testValue",
client.globalRequestParamType.headers);
```

removeGlobalRequestParam

The removeGlobalRequestParam method removes the specified param from network calls.

Syntax

```
removeGlobalRequestParam(paramName, paramType);
```

Parameters

Name	Type	Description
paramName	String	The name of the parameter.
paramType	globalRequestParamType	The type of the parameter.

Example

```
// This code example removes the global header with header name
testHeader.
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
client.removeGlobalRequestParam("testHeader",
client.globalRequestParamType.headers);
```

resetGlobalRequestParams

The resetGlobalRequestParams method resets all the global request params.

Syntax

```
resetGlobalRequestParams()
```

Parameters

None

Example

```
// This example resets all the global request params
// Get an instance of SDK
// Assume SDK already initialized
var client = kony.sdk.getCurrentInstance();
client.resetGlobalRequestParams();
```

39.3.3 Setting User ID

The **setUserID** API sets the user ID for the data gathered from an application. The user ID allows the data to be tracked on a user basis for broad analysis like how many different users used the application. It also helps to track activities of a specific user, which can help in seeing what activities were done before a crash, or what events led to a transaction not passing through. The user ID allows the same user to be tracked across different devices as well.

```
// Sample code for the setUserID API
kony.setUserID("userID");
```

> ***Note:*** **KNYMetricsService.setUserId** has been removed from apps built with Kony Visualizer Enterprise.

39.3.4 HTTP Message Body Integrity

The Client App Security feature helps to secure data exchanged between a client app and a server app. Enterprise class applications may need to ensure that network traffic being exchanged between the server and client app is not tampered with. This feature detects and reports network traffic tampering on the data exchanged between the server and client app.

> ***Important:*** HTTP Integrity does not support Scheduler job.

39.3.4.1 Best Practices - HTTP Message Body Integrity

As a security best practice, it is recommended that you have different Kony Fabric applications with different security keys for Native and Web.

39.3.4.2 Error Message - HTTP Message Body Integrity

While exchanging data between a client app and a server app and if the data is tampered, the following error message appears:

- Error code:`1019`
- Error message: `Http message Body Integrity Check failed`

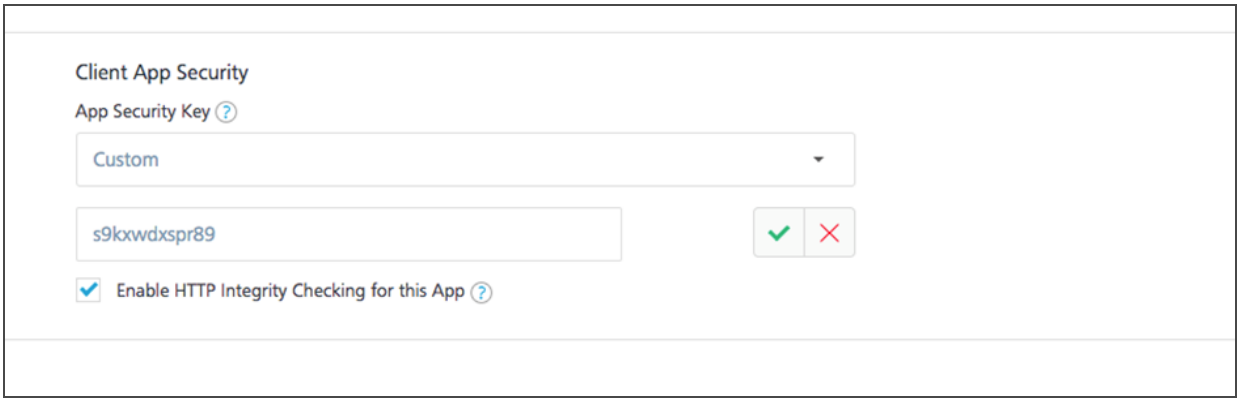
39.3.4.3 Limitations - HTTP Message Body Integrity

- HTTP message body integrity does not support **Metrics and Binary** features.

> ***Note:*** You can enable HTTP Message Body Integrity by using Kony Fabric > Identity > SERVICE CONFIGURATION.

39.3.4.4 Usage of Custom App Security Key

Custom App Security Key that is configured in Kony Fabric must be provided to the SDK prior to initializing the SDK using a preshow action.



Syntax

```
setAppSecurityKey(customSecurityKey)
```

Parameters

Input Parameter	Type	Description	Required
customSecurityKey	String	Custom security key defined in Kony Fabric by the user. It contains alphanumeric characters.	Yes

Return Type

Type	Value
Boolean	True
JSON	JSON object with keys, **errmsg** and **errcode**.

Sample Code

```
// Note: This piece of code must be called from preAppInit (i.e before SDK init)
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var response = client.setAppSecurityKey("customSecurityKey");
if (response !== null && response === true) {
    kony.print("Custom security key is set successfully");
} else {
    kony.print(response.errmsg + " and " + response.errcode);
}
```

Error Codes

Error Code	Error Message
1023	Security Key must be a non empty string.

Note:

- Client and server apps must be in sync with each other.
  - If the **Custom Security Key** is enabled in Kony Fabric, add it in the client app.
  - If the **Custom Security Key** is disabled in Kony Fabric, remove it from the client app.
- The Integrity Check fails in the following scenarios:
  - If the **Custom Security Key** is enabled on the client app and disabled on the server.
  - If the **Custom Security Key** is enabled on the server and disabled on the client app.
- Rebuild the client application in the following scenarios of Fabric Console.
  - If HTTP Integrity is toggled.
  - If the app security key is toggled between App Secret and Custom.

39.3.5 Server Event APIs

Server Events is a capability of the Kony Fabric runtime server that lets the backend services to generate and subscribe to events. Server Events help you to generate events asynchronously such as processing a submitted order and invoking a time-taking activity where the client app does not need to wait for the response.

Note:

- The Server Events APIs are supported in Android, iOS, and SPA/DW platforms.
- Only one connection per application and one callback for event notification is allowed.

39.3.5.1 subscribeServerEvents API

The **subscribeServerEvents** API opens a connection and sends a subscription message for the topics you have provided. After subscribing, the application starts receiving the ServerEvents' messages for the subscribed topics.

Syntax

```
KNYMobileFabric.subscribeServerEvents(topicsToSubscribe,
subscribeOptions);
```

Parameters

Parameter	Type	Description
topicsToSubscribe	String	Specifies the events to be subscribed.
subscribeOptions	JSON	It determines the success and failure of the subscription. This parameter must contain the following functions: • **onEventCallback**: If there is a reply from the server, the onEventCallback function is invoked. • **onFailureCallback**: If the subscription fails, the onFailureCallback function is invoked.

Sample Code

```
var eventsToSubscribe = ["service1/operation1", "service1/operation2",
"service2/operation1"];

subscribeOptions = {

    "onEventCallback": function(message) {
```

```
        //Handle the server event notification

    },

    "onFailureCallback": function(error) {

        //Handle the subscription failure, majorly due to websocket
failure.

    }

};

KNYMobileFabric.subscribeServerEvents(eventsToSubscribe,
subscribeOptions);
```

39.3.5.2 unsubscribeServerEvents API

The **unsubscribeServerEvents** API is used to unsubscribe the ServerEvents' messages for the topics you have provided.

> ***Note:*** If you unsubscribe from all the topics, we recommend you to close the existing connection.

Syntax

```
KNYMobileFabric.unsubscribeServerEvents(topicsToUnsubscribe,
unsubscribeOptions);
```

Parameters

Parameter	Type	Description
topicsToUnsubscribe	String	Specifies the events to be unsubscribed.

Parameter	Type	Description
UnsubscribeOptions	JSON	This parameter must contain the following functions: • **closeConnection**: If you want to close the existing connection, closeConnection is invoked. • **onCloseCallback**: This function is invoked when the closeConnection function is set to true and websocket is closed successfully.

Sample Code

```
var eventsToUnsubscribe = ["service1/operation1",
"service1/operation2", "service2/operation1"];

unsubscribeOptions = {

    "closeConnection": true, //pass this value as true if you want to
close the existing connection.

    "onCloseCallback": function(error) {

        //callback will be invoked if closeConnection is set to true
and websocket connection is closed successfully.

    }
```

```
};

KNYMobileFabric.unSubscribeServerEvents(eventsToUnsubscribe,
unsubscribeOptions);
```

39.3.5.3 publishServerEvents API

The publishServerEvents API publishes events to server from the client SDK API.

Syntax

```
KNYMobileFabric.publishServerEvents(eventsToPublish);
```

Parameters

Parameter	Type	Description
eventsToPublish	String	Specifies the events to be published.

Sample Code

```
var eventsToPublish = [{
    "topic": "transaction/deposit",

    "data": {
        "amount": "1500",
        "user": "clientevents",
        "account": "1000",
        "transaction": "deposit"
    }

}];

KNYMobileFabric.publishServerEvents(eventsToPublish);
```

39.3.5.4 Frequently Asked Questions

1. Can I subscribe to a topic more than once?

   Yes, you can subscribe to events for a topic multiple times. The events will be notified each time an application has subscribed to that topic.

2. Can I unsubscribe to a partial list of topics?

   Yes, you can unsubscribe to a partial list of topics.

3. Can I use different callbacks for every subscription?

   Currently, the API maintains only one callback and the events for all topics will be notified through the same callback. When a different callback is passed with another subscription, all the events will start invoking the latest callback provided in the subscription.

39.3.6 Invoking an Identity Service

The following are the methods you can use for an identity service.

- Login with provider type as Basic
- Login with provider type as OAuth/SAML
- Login with provider type as OAuth 2.0 with Deep link URL
- Get Backend Token
- User Profile
- Get Provider Name
- Get Provider Type
- Use Persisted Login
- Logout

39.3.6.1 Login with provider type as Basic

```
// Sample code to authenticate to Kony Fabric client

var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var options = {};
options["userid"] = "userid";
options["password"] = "password"; // Optional values for login
options["loginOptions"]["include_profile"] = false;
options["loginOptions"]["isSSOEnabled"] = false;
options["loginOptions"]["continueOnRefreshError"] = false; //Throws
error if previous IdentitySession has expired
options["loginOptions"]["persistLoginResponse"] = false;
options["loginOptions"]["isOfflineEnabled"] = false;
identitySvc.login(options, function(response) {
    kony.print("Login Success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Login Failure: " + JSON.stringify(error));
});
```

Note:

- The **isSSOEnabled** flag set to true, indicating that Single Sign-On (SSO) is enabled. This parameter must be passed every time the user logs in. The parameters **userid** and **password** are required only for the first login. For more information about SSO, refer to Single Sign-On.

Supported Platforms for SSO	
Platform	SDK Version
Android	7.3 and higher versions
iOS	7.3 and higher versions
Web	V8 SP4 and higher versions

Support for Multi-Login with SSO is available from V8 SP4 and higher versions.

- The earlier code sample provides the following information:
    - The **include_profile** parameter is set to true. This parameter specifies whether to encode the user profile as part of the claims token.
    - The **continueOnRefreshError** flag is set to false. The default value is **true**. If the **continueOnRefreshError** is set to **false** and the previous identity provider session has expired, error 1017 is thrown. The error message is "Transient Login failed. Previous Identity Token expired in backend."
    - From version 7.2.5 onward Kony Fabric apps allow users to use services protected by different Identity services in the same application session as long as users have authenticated to the Identity service. This allows a user to login to Google and Facebook, for example, via identity services and use integration services dependent on those identity services in the same application session, without having to logout of either.
    - The **isOfflineEnabled** flag is set to true. The default value is false. This option allows users to use previous login information when they are in offline mode. Typically the login details are lost when the app is in offline mode. This option stores the user ID and password to be used if authentication is needed when there is no network connection.

      To use the **isOfflineEnabled** option, you must enable the "Bundle OpenSSL Library"

option in Project Settings. For information about this setting see Additional Settings.

This option is valid for iOS and Android platforms only.

This option is available for basic login only. It is not available for OAuth/SAML login.

- The **persistLoginResponse** flag is set to true. The default value is false. This option allows users to login to an app once and reuse the response across multiple app sessions. Typically the login details are lost when the app is closed. The **persistLoginResponse** option stores the claims token in the data store of the client device. This allows the app developer to use the token to invoke the integration or object services without prompting the user to authenticate again. If the token has expired, the login will fail.

  To use this option, the Kony Fabric server must be configured so that the Identity session is enabled for the entire time the persisted response is needed. For more information about configuration settings see How to Configure App Session Settings.

  In the Android platform, you must also enable the "Bundle OpenSSL Library" option in Project Settings. For information about this setting see Additional Settings.

  To retrieve the claims token from the data store, use the usePersistedLogin() API.

Important:

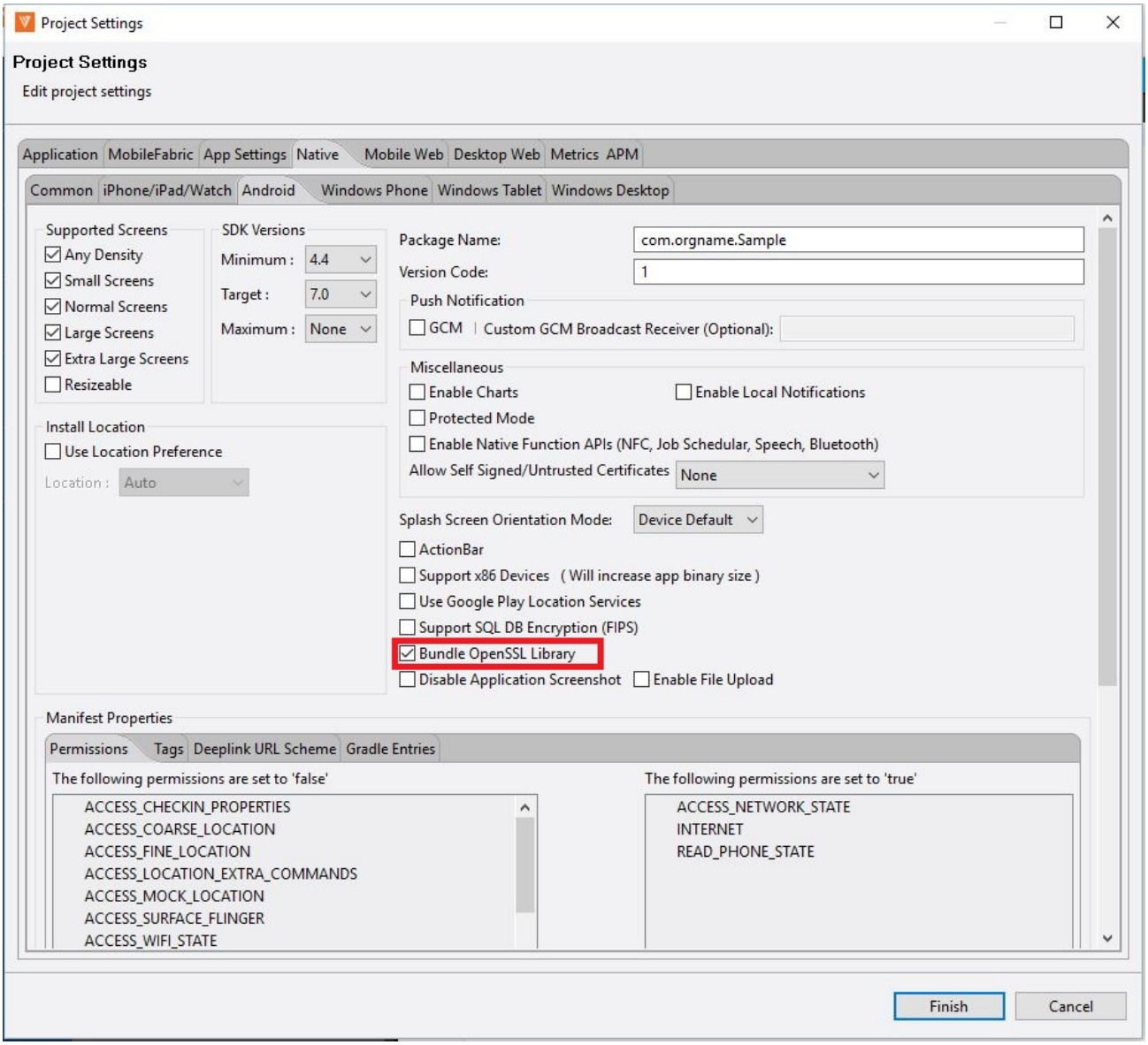
- When you select Kony User Repository as the identity type, the system does not allow you to provide an identity name.
- To use Kony User Repository as authentication service, the value for `providerName` must be set as `userstore`. If you set it with any other value (for example, Kony User Repository, User Store or Cloud Repository), the system throws an error.

Additional Settings

To use the **isOfflineEnabled** and **persistLoginResponse** options in the Androd platform, you must ensure that the "Bundle Open SSL Library" option is enabled in the Kony Visualizer Project Settings. To do this, perform the following steps.

1. On the File menu, click **Settings** to open the Project Settings dialog box.
2. Click the **Native** tab.
3. Click the**Android** sub-tab.
4. Check the option for "Bundle OpenSSL Library".

The page will look like the following example.



39.3.6.2 Login with provider type as OAuth/SAML

```
// Sample code to authenticate to Kony Fabric Client
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var options = {};
options.include_profile = true;
//browserWidget is mandatory if using the MVC Architecture
options.browserWidget = myForm.myBrowserWidget;
 var loginOptions = {};
loginOptions.isSSOEnabled = true;
loginOptions.continueOnRefreshError = false;
loginOptions.persistLoginResponse = true;
options.loginOptions = loginOptions;
identitySvc.login(options, function(response) {
    kony.print("Login success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Login failure: " + JSON.stringify(error));
});
```

Note:

- The client is the **kony.sdk();** object.

The earlier code provides the following information:

- The **isSSOEnabled** flag set to true, indicating that Single Sign-On (SSO) is enabled. This parameter must be passed every time the user logs in. For more information about SSO, refer to Single Sign-On.
- The **include_profile** parameter set to **true**. This parameter specifies whether to encode the user profile as part of the claims token.

- The **continueOnRefreshError** flag set to **false**. The default value is **true**. This flag throws error 1017 with error message "Transient Login failed, Previous Identity Token expired in backend". This error is thrown when the **continueOnRefreshError** is set to **false** and the previous identity provider session has expired.

  From version 7.2.5 onward Kony Fabric apps allow users to use services protected by different Identity services in the same application session as long as users have authenticated to the Identity service. This allows a user to login to Google and Facebook, for example, via identity services and use integration services dependent on those identity services in the same application session, without having to logout of either.

- The **persistLoginResponse** flag is set to true. The default value is false. This option allows users to login to an app once and reuse the response across multiple app sessions. Typically the login details are lost when the app is closed. The **persistLoginResponse** option stores the claims token in the data store of the client device. This allows the app developer to use the token to invoke the integration or object services without prompting the user to authenticate again. If the token has expired, the login will fail.

- When SSO is enabled, log out with SLO = true does not log out the OAuth provider.

  To use this option, the Kony Fabric server must be configured so that the Identity session is enabled for the entire time the persisted response is needed. For more information about configuration settings see How to Configure App Session Settings.

  For the Android platform, you must also enable the "Bundle OpenSSL Library" option in Project Settings. For information about this setting see Additional Settings.

- To retrieve the claims token from the data store, use the usePersistedLogin() API.

  From version 7.2.02 onward Kony Fabric SDK provides an option to customize the OAuth login form using a user-defined form with a browser widget inside it. The browserWidget parameter accepts a kony.ui.Browser instance. The SDK will run oAuth related logic on the provided browserWidget instance. This parameter enables a developer to provide a customized oAuth UI. If the provided value is not defined or if it is not an instance of the

kony.ui.browserWidget, the SDK will create its own browser window with default configuration/customization for oAuth Login.

- The browserWidget field is mandatory for Kony Applications built by using the MVC architecture on Kony Visualizer.

39.3.6.3 Login with provider type as OAuth 2.0 with Deep link URL

```
// Sample code to authenticate to Kony Fabric Client

var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var username = "username_for_identity_service";
var password = "password_for_identity_service";
var options = {};
options["userid"] = username;
options["password"] = password;
options["UseDeviceBrowser"] = true;
// This parameter in options will open the login url in native
browser.
// This is a deeplink url, where the control will be redirected after
login.
//#ifdef android
options.success_url = "Deep link url registered for android";
//#else
options.success_url = "Deep link url registered for rest";
//#endif
identitySvc.login(options, function(response) {
    kony.print("Login success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Login failure: " + JSON.stringify(error));
});
```

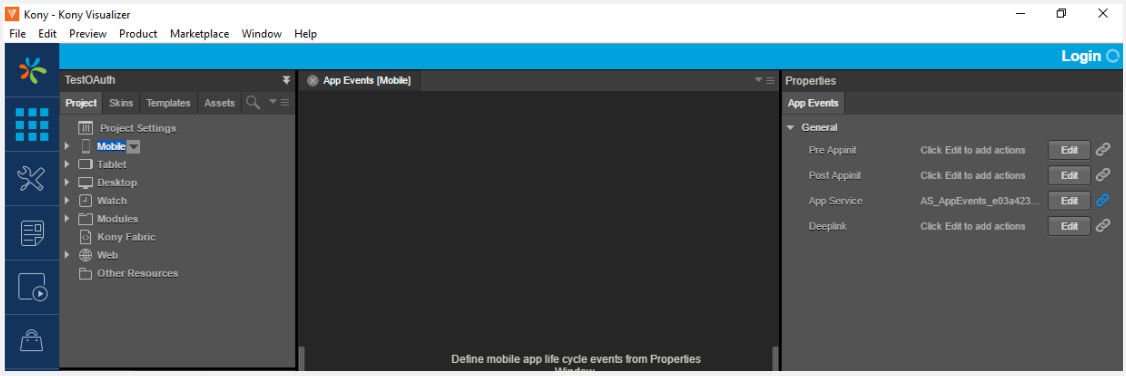
The sample above shows various parameters similar to the parameters of the Login with provider type as OAuth/SAML. The following two optional parameters are added further.

- **UseDeviceBrowser**: This parameter opens the login URL in the native internet browser of the device.
- **success_url**: After the log in is successful, control is redirected to the URL (deep link URL).

Deep link URL is the URL that is registered to the application. After redirection, the client calls the method **handleDeeplinkCallback**. (A global function)

Method signature: `function handleDeeplinkCallback(query params)`

```
// Sample code to call handleDeeplinkCallback after the deeplink
redirection is done.
// This method will be called after the deeplink redirection.
//Need to register a call in the App services tab in App Events.
```




```
function appservicereq(params) {
    handleDeeplinkCallback(params); // Required validations are done
and proceed with rest of login flow.
}
```

For more information on deep links, click here.

> ***Note:***
>
> - The client is the **kony.sdk();** object.

The earlier code sample provides the following information:

- The **isSSOEnabled** flag set to true, indicating that Single Sign-On (SSO) is enabled. This parameter must be passed every time the user logs in. For more information about SSO, refer to Single Sign-On.
- The **include_profile** parameter set to **true**. This parameter specifies whether to encode the user profile as part of the claims token.
- The **continueOnRefreshError** flag set to **false**. The default value is **true**. This flag throws error 1017 with error message "Transient Login failed, Previous Identity Token expired in backend". This error is thrown when the **continueOnRefreshError** is set to **false** and the previous identity provider session has expired.

  From version 7.2.5 onward Kony Fabric apps allow users to use services protected by different Identity services in the same application session as long as users have authenticated to the Identity service. This allows a user to login to Google and Facebook, for example, via identity services and use integration services dependent on those identity services in the same application session, without having to logout of either.
- The **persistLoginResponse** flag is set to true. The default value is false. This option allows users to login to an app once and reuse the response across multiple app sessions. Typically the login details are lost when the app is closed. The **persistLoginResponse** option stores the claims token in the data store of the client device. This allows the app developer to use the token to invoke the integration or object services without prompting the user to authenticate again. If the token has expired, the login will fail.
- When SSO is enabled, logging out with SLO = true does not log out the OAuth provider.
- **The Deeplink API helps the device browser to authenticate the OAuth This can be used for any OAuth2.0 providers. If an OAuth provider does not allow embedded browsers, such as Google OAuth, you must use the Deeplink API, which launches the device browser instead of the embedded browser.**

  To use this option, the Kony Fabric server must be configured so that the Identity session is

enabled for the entire time the persisted response is needed. For more information about configuration settings see How to Configure App Session Settings.

For the Android platform, you must also enable the "Bundle OpenSSL Library" option in Project Settings. For information about this setting see Additional Settings.

- To retrieve the claims token from the data store, use the usePersistedLogin() API.

From version 7.2.02 onward Kony Fabric SDK provides an option to customize the OAuth login form using a user-defined form with a browser widget inside it. The browserWidget parameter accepts a kony.ui.Browser instance. The SDK will run oAuth related logic on the provided browserWidget instance. This parameter enables a developer to provide a customized oAuth UI. If the provided value is not defined or if it is not an instance of the kony.ui.browserWidget, the SDK will create its own browser window with default configuration/customization for oAuth Login.

39.3.6.4 Get Backend Token

```
// Sample code to get backend token for provider
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var options = {
    "requestParams": {
        "refresh": "true"
    }
};
var forceFromServer = false;
identitySvc.getBackendToken(forceFromServer, options, function
(response) {
    kony.print("Backend token is: " + JSON.stringify(response));
}, function(error) {
    kony.print("Failed to get backend token: " + JSON.stringify
```

```
(error));
});
```

> ***Note:***
>
> - If forceFromServer is true, then the SDK fetches the token from the server. If forceFromServer is false, then the SDK gives you the token present in localStorage. Please note that only few backend providers such as Salesforce support refresh. If a backend provider does not support refresh, passing `forceRefreshFromServer=true` would result in empty response from this api.
> - The authClient is the IdentityService object.

39.3.6.5 User Profile

```
//Sample code to get user profile details
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var forceFromServer = false;
identitySvc.getProfile(forceFromServer, function(response) {
    kony.print("User profile is: " + JSON.stringify(response));
}, function(error) {
    kony.print("Failed to fetch profile: " + JSON.stringify(error));
});
```

> ***Note:***
>
> - If forceFromServer is true, then the SDK fetches the token from the server. If forceFromServer is false, then the SDK gives you the token present in localStorage.
> - The authClient is the IdentityService object.

39.3.6.6 Get Provider Name

```
//Sample code to get provider name
var providerName = authClient.getProviderName();
```

> ***Note:*** The authClient is the IdentityService object.

39.3.6.7 Get Provider Type

```
// Sample code to get provider name
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var providerName = identitySvc.getProviderName();
```

> ***Note:*** The authClient is the IdentityService object.

39.3.6.8 Use Persisted Login

When the login option (persistLoginResponse) is set as true, the auth response is stored in the datastore of the client device. You can invoke the **usePersistedLogin** API to check if the login response was stored.

If the API returns true, you can use the services authorized by that login without having to sign in again.

If the API returns false, you need to get authorized again. The persisted response is cleared only when you sign out.

Return Type: Boolean

```
// Sample code to retrieve the claims token from the data store.
var serviceName = "identity_service_name";
// Get an instance of SDK
```

```
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var isLoginPersisted = identitySvc.usePersistedLogin();
```

39.3.6.9 Logout

```
// Sample code to logout from auth service
var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var options = {};
options["slo"] = true;
options["browserWidget"] = myForm.browserWidget;
identitySvc.logout(function(response) {
    kony.print("Logout success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Logout failure: " + JSON.stringify(error));
}, options);
```

Note:

- The authClient is the IdentityService object for the loggedIn provider.
- To log out from all applications, the user must log out of every Identity service that they are logged on to.
- The code sample shows the parameter **slo** set to **true**, indicating that all apps will be logged out when SSO is enabled.
- If **slo** is set to **false**, or if the app does not send **slo**, the user is logged out of the app. The user is not logged out of the other apps that are logged in using SSO.
- Any apps that use SSO based services must log in by entering credentials and re-initiating SSO.
  For more information, refer to Single Sign-On.

39.3.7 Invoking an Integration Service

This API invokes an integration service that is configured in the Kony Fabric portal.

```
// Sample code to fetch the integration service details
var serviceName = "integration_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var integrationSvc = client.getIntegrationService(serviceName);
var operationName = "operation_name";
var params = {
    "custom-input-key1": "custom-input-value1"
};
var headers = {
    "custom-header-key1": "custom-header-value1"
}; //If there are no headers,pass null
// options is an optional parameter that helps in configuring the
network layer.
// To configure for a thin layer, use xmlHttpRequestOptions instead of
httpRequestOptions.
// Values for timeoutIntervalForRequest and timeoutIntervalForResource
are in seconds.
// If the request is failing because the network is unreliable, you
can provide the following values:
// timeoutIntervalForRequest: This is the time taken by the client to
receive headers. This is in seconds.
// timeoutIntervalForResource: The timeout for each chunk download.
This is in seconds.
var options = {
    "httpRequestOptions": {
        "timeoutIntervalForRequest": 60,
        "timeoutIntervalForResource": 600
    }
```

```
};
integrationSvc.invokeOperation(operationName, headers, params,
function(response) {
    kony.print("Integration Service Response is: " + JSON.stringify
(response));
}, function(error) {
    kony.print("Integration Service Failure:" + JSON.stringify
(error));
}, options);
```

> ***Note:*** The client is the kony.sdk(); object.

httpRequestOptions	Datatype	Comments
timeoutIntervalForRequest	int	This is a time out value for the HTTP connection. This can also be referred as connection time out value in seconds.
timeoutIntervalForResource	int	This is used to give a maximum time in seconds for which the network resource should be kept alive on iOS device. This is only applicable for *background network* calls and default value is 1 week (7 days) unless specified in options.
enableBackgroundTransfer	boolean (true/false)	Enables HTTP request calls in background in iOS. ***Note:*** This may lead to duplicate transactions in the system, should only be used for GET calls. iOS internally retries the request to keep the connection alive till it reaches timeoutIntervalForResource value, which may create duplicate transactions in back end.

xmlHttpRequestOptions	Datatype	Comments
enableWithCredentials	true/false	To allows CORS requests in SPA

39.3.7.1 Error Codes for Failure Callbacks for Integration Services

Error Code	Error Message
100	UnhandledMFcode
101	Invalid User Credentials.
102	Invalid App Credentials.
103	Invalid User/App Credentials.
104	Session/Token got invalidated in the backend. Please login.
105	Invalid provider in appServices.
106	Claims Token is Unavailable
1000	An unknown error has occurred
1011	An error occurred while making the request. Please check device connectivity, server url and request parameters
1013	Invalid JSON response was returned
1014	Request to server has timed out

39.3.8 Invoking an Integration Service with Response Passthrough

The Fabric SDK expects a JSON response from Fabric services, by default. To indicate that responses from the target integration service are a passthrough, provide the **passthrough** option to the Integration Service Invocation API. By doing so, the SDK does not convert the response to JSON format.

You can configure an integration service response as a passthrough to cater to specific needs such as downloading binary content. To do so, enable the **Response Passthrough** flag in the Fabric application.

> ***Note:*** For the combination of Visualizer V9 and Fabric V9, the passthrough flag has been deprecated. In this case, the SDKs will determine whether an integration service response is a passthrough or not.

Syntax

```
integrationClient.invokeOperation(operationName, headers, params,
successCallback, failureCallback, options)
```

Parameters

Name	Type
operationName	string
headers	Dictionary
params	Dictionary
successCallback	function
failureCallback	function

Name	Type
options	Dictionary

Sample Code

```
// Sample code to fetch the integration service details with
passthrough
var serviceName = "integration_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var integrationSvc = client.getIntegrationService(serviceName);
var operationName = "operation_name";
var params = {
    "custom-input-key1": "custom-input-value1"
};
var headers = {
    "custom-header-key1": "custom-header-value1"
}; //If there are no headers, pass null
// options is an optional parameter that helps in configuring the
network layer.
integrationSvc.invokeOperation(operationName, headers, params,
function(response) {
    kony.print("Integration Service Response is: " + JSON.stringify
(response));
}, function(error) {
    kony.print("Integration Service Failure: " + JSON.stringify
(error));
}, {
    "passthrough": true
});
```

39.3.8.1 Get Endpoint URL

```
// Sample code to get the endpoint URL
var serviceName = "integration_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var integrationSvc = client.getIntegrationService(serviceName);
var url = integrationSvc.getUrl();
```

> ***Note:*** The integrationClient is the IntegrationService object.

39.3.8.2 Error Codes for Failure Callbacks for Integration Services

Error Code	Error Message
100	UnhandledMFcode
101	Invalid User Credentials.
102	Invalid App Credentials.
103	Invalid User/App Credentials.
104	Session/Token got invalidated in the backend. Please login.
105	Invalid provider in appServices.
106	Claims Token is Unavailable
1000	An unknown error has occurred

Error Code	Error Message
1011	An error occurred while making the request. Please check device connectivity, server url and request parameters
1013	Invalid JSON response was returned
1014	Request to server has timed out

39.3.9 Invoking a Configuration Service

Admin Console provides an interface to define a set of key value pairs at the server and the client level. You can find it in Kony Fabric Console > Admin Console > Settings > Configurable Parameters -> Client App Properties.

You can configure the Client specific properties any time in the server, so that client application can pull and consume the updated properties at runtime.

```
// Sample code to fetch the client app properties from server.
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var configurationSvc = client.getConfigurationService();
configurationSvc.getAllClientAppProperties(function(response) {
    kony.print("client key value pairs retrieved: " + JSON.stringify
(response));
}, function(error) {
    kony.print(" Failed to retrieve client key value pairs: " +
JSON.stringify(error));
});
```

39.3.10 Invoking a Logic Service

The getLogicService API creates an instance of logic service that is configured in the Kony Fabric portal. `logicClient = KNYMobileFabric.getLogicService(serviceName)`

The invokeOperation API invokes the backend operation using the object of logic service. The invoke operation function is as shown below:

```
// Sample code to fetch the logic service details
var serviceName = "logic_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var logicSvc = client.getLogicService(serviceName);
kony.print("Response is :" + logicSvc.getLogicServiceUrl());
var path = "path_defined_on_KonyFabric"; // TODO - what path??
var HttpMethodType = "POST"; // Other supported types: "PUT", "GET",
"DELETE"
var params = {
    "custom-input-key1": "custom-input-value1"
};
var headers = {
    "custom-header-key1": "custom-header-value1"
};
logicSvc.invokeOperation(serviceName, path, HttpMethodType, headers,
params, function(response) {
    kony.print("Successfully fetched logic service: " + JSON.stringify
(response));
}, function(error) {
    kony.print("error occurred in fetching logic service: " +
JSON.stringify(error));
});
```

39.3.11 Invoking a Messaging Service

A developer should register with Google Cloud Messaging (GCM) for Android services to get the deviceToken that is used to register with Kony Fabric Messaging. Also a developer should fetch the **deviceId** and **userfriendlyId** to create an instance of messaging service.

The following are the methods you can use for a messaging service.

- Register
- Register With Auth Token
- Unregister
- Unregister With Auth Token
- Update GeoLocation
- Update GeoLocation With Auth Token
- Register geoBoundaries
- Fetch All Messages
- Mark Message as Read
- Fetch Message Content from Kony Fabric Messaging
- Subscribe Audience
- Get Audience Details by Subscription ID
- Delete Subscribed Audience
- Update List of Beacons for a Device
- Get Rich Push Content

39.3.11.1 Register

Register API registers to the engagement server.

Syntax

```
register = function(osType, deviceId, regID, UFID,successCallback,
failureCallback, options)
```

Parameters

Name	Type	Description	Required
osType	String	Type of operating system	Yes
deviceID	String	Device ID of the device	Yes
regID	String	Registration ID of the user	Yes
UFID	String	User friendly ID configured in messaging service console	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	JSON	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
```

```
var deviceId = kony.os.deviceInfo().deviceid;
var UFID = "user_friendly_id";
var osType = "iphone"; /*"androidgcm" for android, "iphone" for
iphone, "ipad" for ipad, "ipod" for ipod*/
var options = {
    "authToken": "authorization_token"
}; //To get regID, use kony.push.register() method
var regID = kony.push.register();
messagingSvc.register(osType, deviceId, regID, UFID, function
(response) {
    kony.print("register with auth token success: " + JSON.stringify
(response));
}, function(error) {
    kony.print("register with auth token failed: " + JSON.stringify
(error));
}, options);
```

39.3.11.2 Register with Auth Token

The **registerWithAuthToken** API registers with an Auth Token for the messaging service.

Syntax

```
registerWithAuthToken = function(osType, deviceId, regID, UFID,
authToken, successCallback, failureCallback)
```

Parameters

Name	Type	Description	Required
osType	String	Type of operating system	Yes
deviceId	String	Device ID of the device	Yes

Name	Type	Description	Required
regID	String	Registration ID of the user.	Yes
UFID	String	User friendly id or reconciliation key which is configured in engagement console.	Yes
authToken	String	Authorization token configured in engagement console	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes

Example

```
// Sample code to register with an Auth Token
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
messagingSvc.registerWithAuthToken("osType", "deviceId", "regID",
"UFID", "authToken", function(response) {
    kony.print("Subscription Success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Subscription Failure: " + JSON.stringify(error));
});
```

Important: You must register to a messaging service at least once to use the following APIs.

39.3.11.3 Unregister

Unregister API unregisters with the engagement server.

Syntax

```
unregister = function(successCallback, failureCallback, options)
```

Parameters

Name	Type	Description	Required
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	JSON	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var options = {
    "authToken": "xyz"
};
messagingSvc.unregister(function(response) {
    kony.print("Unregistration Success: " + JSON.stringify(response));
}, function(error) {
    kony.print("Unregistration Failure: " + JSON.stringify(error));
}, options);
```

39.3.11.4 Unregister with Auth Token

The **unregisterWithAuthToken** API unregisters from the messaging service with the Auth Token.

Syntax

```
unregisterWithAuthToken = function(authToken, successCallback,
failureCallback)
```

Parameters

Name	Type	Descroption	Required
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	JSON	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var authToken = "Authorization token configured in messaging service
console";
messagingSvc.unregisterWithAuthToken(authToken, function(response) {
    kony.print("Unregistration Success: " + JSON.stringify(response));
}, function(error) {
```

```
    kony.print("Unregistration Failure: " + JSON.stringify(error));
});
```

39.3.11.5 Register geoBoundaries

The **registerGeoBoundaries** API registers geoboundaries to be monitored by the Engagement Service. The parameters of this API specify the radius and number of geoboundaries to monitor.

Use Case

Suppose you have a product that is being sold in several retail locations throughout the area. You have a special offer and you want to send an alert to customers when they are near one of the retail locations. The **registerGeoBoundaries** API is called during the app initialization. The API gets the geoboundaries based on the user's current location and stores them in the device OS. The OS will prompt the SDK when the user reaches a geoboundary and the application will check what action is to happen. The possible actions are:

- A local push notification from the client application is displayed.
- The engagement server sends a notification to display.
- Some custom logic is executed by a callback function specified by the customLogicCallback option.

refreshBoundary

Suppose that the radius parameter was specified as 5 miles when the user launched the app from his home. As the user moves away from his home, there is a point at which the set of geoboundaries that is being monitored should be refreshed. This is referred to as the refreshBoundary. In this example, when the user has moved 2.5 miles, the SDK will prompt the server for a new set of location data.

Syntax

```
registerGeoBoundaries(options, successCallback, failureCallback);
```

Parameters

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>options</td><td>jsonObject</td><td>The jsonObject has the following geoBoundary options
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>radius</td><td>number</td><td>Radius of the refresh boundary, in miles. This value defines the radius that must be monitored. By default, this value is taken from the Kony Engagement Service. The user can override it.</td></tr>
<tr><td>pageSize</td><td>number</td><td>Number of geoBoundaries that can be monitored through the SDK. Default is 19.

Platform limitations

• Android - Can monitor 99 geoBoundaries for a given instance.

• Windows and iOS - Can monitor 19 geoBoundaries for a</td></tr>
</table></td></tr>
</table>

Name	Type	Description
successCallback	function	Callback method on success
failureCallback	function	Callback method on failure

Additional Settings

There are additional settings that must be enabled for this API to work correctly.

> ***Note:*** Exceptions are thrown if location is switched off on the device, or user does not allow permission to retrieve location, or the SDK is not able to retrieve the current location on the device. For more information on SDKs for Messaging Service docs, refer to Kony Visualizer API Developers' Guide > Notifications > Push Notifications

Android

To enable the **registerGeoBoundaries** API for the Android platform perform the following steps in Kony Visualizer.

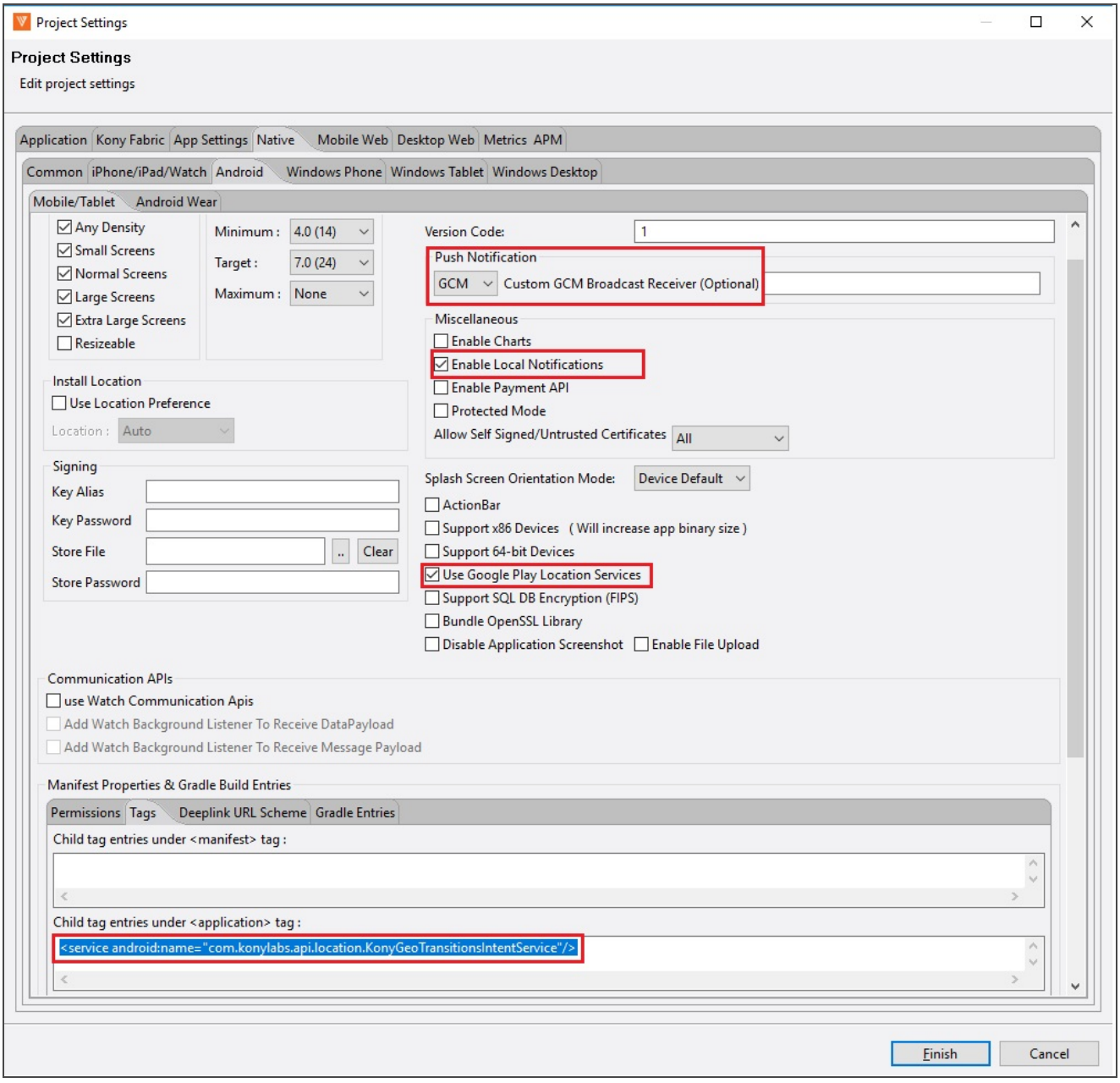
1. On the File menu, click **Settings** to open the Project Settings dialog box.
2. Click the **Native** tab.
3. Click the **Android** sub-tab.
4. In **Push Notifications** section, select either **GCM** or **FCM** for the engagement APIs to work.
5. Enable the following items:

   Enable Local Notifications

   Use Google Play Location Services

6. In the **Manifest Properties** section, select Tags. Add the following to the **Child tag entries <application>** tag:
   <service android:name="com.konylabs.api.location.KonyGeoTransitionsIntentService"/>

The page will look like the following example.



iOS

To enable the **registerGeoBoundaries** API for the iOS platform, add the following keys to the info.plist file.

- "NSLocationAlwaysUsageDescription":"<ThisMessageWillBeDisplayedToUser>"
- "UIBackgroundModes":["location"]

For information about how to access and edit the info.plist file, refer to Build an App for iOS.

> *Important:*
> - Include the `NSLocationWhenInUseUsageDescription` and `NSLocationAlwaysAndWhenInUseUsageDescription` keys in your app's **`Info.plist`** file.
> - If the deployment target is iOS10 or below, the `NSLocationAlwaysUsageDescription` key is required. If those keys are not present, authorization requests fail immediately.

To obtain the geolocation callbacks, run the following code in the **preappinit** of your application.

```
var msgObj = kony.sdk.getCurrentInstance().getMessagingService();
var cback = msgObj.manageGeoBoundariesCallback;
kony.location.setGeofencesCallback(cback);
```

> *Important:* Before building the application, link your Fabric application to your Visualizer project.

Windows

To enable the **registerGeoBoundaries** API for the Windows platform perform the following steps in Kony Visualizer.

1. On the **File** menu, click **Settings** to open the Project Settings dialog box.
2. Click the **Native** tab.
3. Click the **Windows Phone** tab.

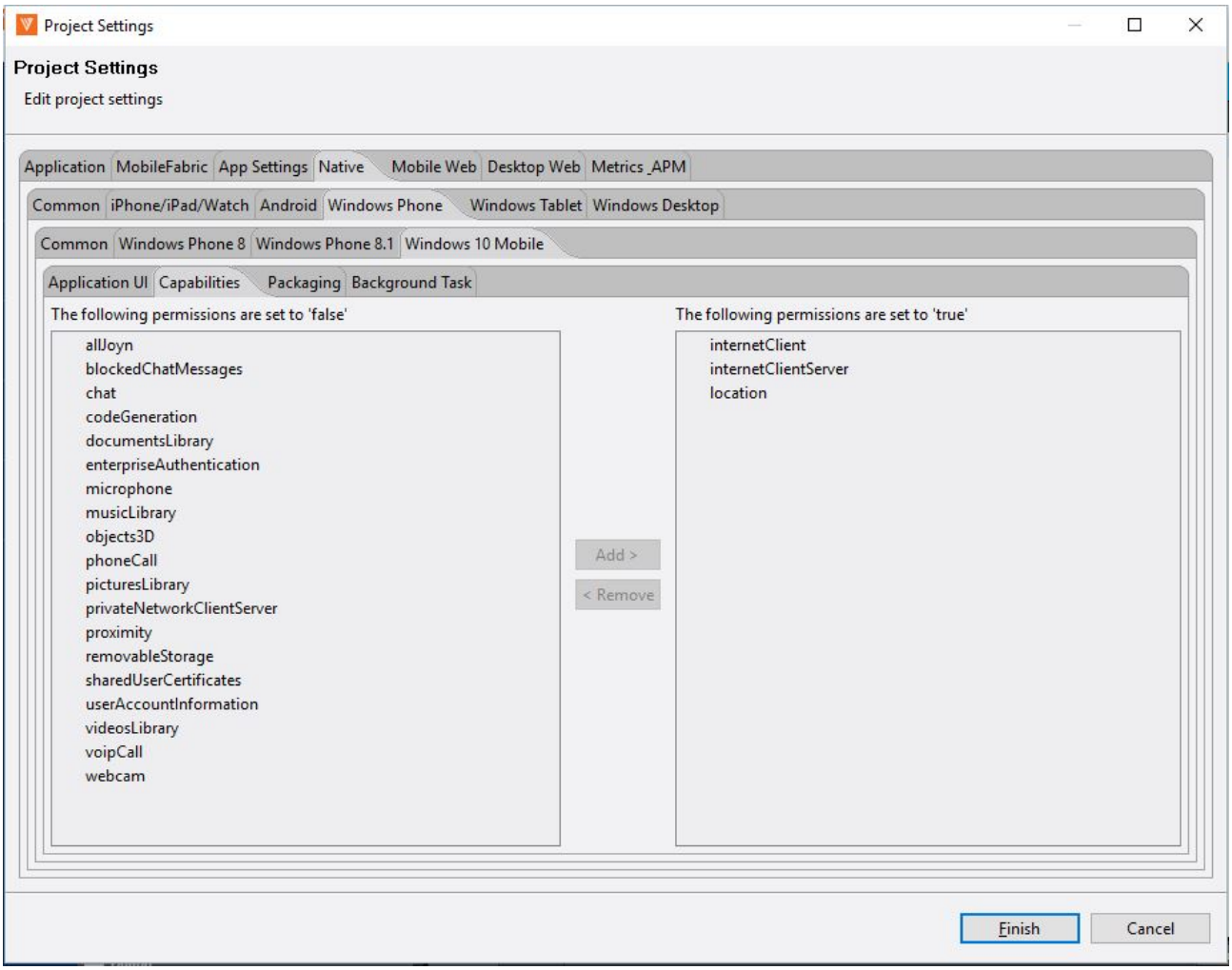
4. Click the **Windows 10 Mobile** tab.
5. Click the **Capabilities** tab.
6. Move the following permissions to "true":

    internetClient

    internetClientService

    location

The Capabilities page will look like the following example.



Example

The following example shows how to register geoboundaries.

```
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
messagingSvc.register("osType", "deviceId", "regID", "UFID", function
(response) {
        kony.print("Register Successful");
        var options = {};
        options.radius = "radius"; //radius in miles
        options.pageSize = "pageSize";
        options.authToken = "Authorization_Token";
        options.tags = "Array of tags"; //TODO
        options.customLogicCallback = function(data) {
            // data is geoBoundary Data received from Kony Messaging
Service
            //custom logic implementation
        };

        function successCallback(response) {
            //Registered Successfully.
        }

        function failureCallback(error) {
            //Registration failure.
        }
        messagingSvc.registerGeoBoundaries(options, successCallback,
failureCallback); //if ksid is already available, register is not
required.
        // If either location is switched off or the SDK is not able
to retrieve current location, exceptions are thrown.
```

```
    },
    function(error) {
        kony.print("Register Failure" + JSON.stringify(error));
    });
```

39.3.11.6 Update GeoLocation

updateGeoLocation API updates the geoLocation for the messaging service.

Syntax

```
updateGeoLocation= function (latitude, longitude, locationName,
successCallback, failureCallback, options)
```

Parameters

Name	Type	Description	Required
latitude	String	Latitude	Yes
longitude	String	Longitude	Yes
locationName	String	Location name	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	String	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var latitude = "latitude_value";
var longitude = "longitude_value";
var locationName = "location_name";
var options = {
    "authToken": "authorization_token"
};
messagingSvc.updateGeoLocation(latitude, longitude, locationName,
function(response) {
    kony.print("Geolocation update successful: " + JSON.stringify
(response));
}, function(error) {
    kony.print("Geolocation update failure: " + JSON.stringify
(error));
}, options);
```

39.3.11.7 Update GeoLocation with Auth Token

The **updateGeoLocationWithAuthToken** API updates the geoLocation with the Auth Token for the messaging service.

Syntax

```
updateGeoLocationWithAuthToken= function(latitude, longitude,
locationName, authToken, successCallback, failureCallback)
```

Parameters

Name	Type	Description	Required
latitude	String	Latitude	Yes
longitude	String	Longitude	Yes
locationName	String	Location Name	Yes
authToken	String	Authorization token that is configured in messaging service console	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes

Example

```
//Sample code to update the geoLocation with the Auth Token for the
messaging service.
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var latitude = "latitude_value";
var longitude = "longitude_value";
var locationName = "location_name";
var authToken = "authorization_token";
messagingSvc.updateGeoLocationWithAuthToken(latitude, longitude,
locationName, authToken, function(response) {
    kony.print("Geolocation update successful : " + JSON.stringify
(response));
}, function(error) {
    kony.print("Geolocation update failure: " + JSON.stringify
```

```
(error));
});
```

39.3.11.8 Fetch All Messages

The **fetchAllMessages** API fetches all messages with/without the Auth Token.

Syntax

```
 fetchAllMessages = function(startIndex, pageSize, successCallback,
failureCallback, options)
```

Parameters

Name	Type	Descroption	Required
startIndex	Number	Starting index of a page	Yes
pageSize	Number	Page size	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	JSON	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var startIndex = 0;
var pageSize = 1000;
messagingSvc.fetchAllMessages(startIndex, pageSize, function(response)
{
    kony.print("Fetched all messages: " + JSON.stringify(response));
}, function(error) {
    kony.print("Failed to fetch messages: " + JSON.stringify(error));
}, {
    "authToken": "authorization_token"
});
```

39.3.11.9 Mark Message as Read

The **markMessageRead** API marks a message as read with/without the Auth Token.

Syntax

```
markMessageRead = function(fetchId, successCallback, failureCallback,
options)
```

Parameters

Name	Type	Description	Required
fetchID	String	A unique ID assigned to a message	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	JSON	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var fetchID = "message ID to mark as read";
messagingSvc.markMessageRead(fetchID, function(response) {
    kony.print("Message marked as read successfully: " +
JSON.stringify(response));
}, function(error) {
    kony.print("Failed to mark message as read: " + JSON.stringify
(error));
}, {
    "authToken": "authorization_token"
});
```

39.3.11.10 Fetch Message Content from Kony Fabric Messaging

The **fetchMessageContent** API fetches a message with/without the Auth Token.

Syntax

```
 fetchMessageContent = function(fetchId, successCallback,
failureCallback, options)
```

Parameters

Name	Type	Description	Required
fetchID	String	A unique ID assigned to a message	Yes
successCallback	Function	Method invoked on success	Yes
failureCallback	Function	Method invoked on failure	Yes
options	JSON	Map for optional parameters	Optional

Optional Keys

Key	Type	Description	Required
authToken	String	Authorization token configured in messaging service console.	Optional

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var fetchID = "message ID to be fetched";
messagingSvc.fetchMessageContent(fetchID, function(response) {
    kony.print("Message content is:" + JSON.stringify(response));
}, function(error) {
    kony.print("Failed to fetch message content: " + JSON.stringify
(error));
}, {
    "authToken": "authorization_token"
});
```

39.3.11.11 Subscribe Audience

The **Subscribe Audience (Create or Update)** API creates a new audience. This API also updates the parameter details of an audience member.

Syntax

```
subscribeAudience = function (firstName, lastName, emailId,
mobileNumber, country, state, successCallback, failureCallback,
options)
```

Parameters

Input Parameter	Type	Description	Required
firstName	String	First name of the user	Yes
lastName	String	Last name of the user	Yes
mobileNumber	Number	Mobile number of the user. If **mobileNumber** is specified as the reconciliationKey, then it is a required value and must be specified for all users. An empty value will cause an error.	Yes
email	String	Email ID of the user	Yes
state	String	If the selected country is USA, the state option is mandatory	Yes
country	String	Country to which the user belongs	Yes
options	JSON	Map for optional parameters and any dynamic properties	Optional

Options keys

Key	Type	Description	Required
active	Boolean	Defines whether the subscription is active or inactive	Optional
emailSubscription	Boolean	Defines whether a user is subscribed to send and receive emails or not	Optional
smsSubscription	Boolean	Defines whether a user is subscribed to send and receive SMS or not	Optional

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var options = {};
options["smsSubscription"] = true;
options["emailSubscription"] = true;
options["active"] = true;
messagingSvc.subscribeAudience("FirstName", "LastName", "EmailID",
"MobileNumber", "Country", "State", function(response) {
    kony.print("Subscribe audience Success: " + JSON.stringify
(response));
}, function(error) {
    kony.print("Subscribe audience Failed: " + JSON.stringify(error));
}, options);
```

> ***Note:*** You can add dynamic properties to the API using options.

Response Status

Code	Description
Status 200	Details added successfully
Status 400	One of the following error messages will be displayed: • Invalid KSID • First Name is required • Last Name is required • Mobile number is required • Audience already exists with the given mobile number • Email ID is required • Country is required
Status 500	Server failed to process request

39.3.11.12 Get subscribed Audience details

The **Get Subscribed Audience Details** API returns the details of the Audience.

Syntax

```
getSubscribedAudienceDetails  = function(successCallback,
failureCallback)
```

Parameters

Output Parameter	Type	Description
id	String	Unique ID assigned to the audience member
firstName	String	First name of the audience member
lastName	String	Last name of the audience member
mobileNumber	String	Mobile number of the audience member
email	String	Email ID of the audience member
active	Boolean	Defines if the audience member is active or inactive
state	String	If the audience member is not a USA national, the response displays a blank string
country	String	Country to which the user belongs to
createdDateStr	String	The date and time when the user was initially created
smsSubscription	Boolean	Defines if the SMS subscription is true or false
emailSubscription	Boolean	Defines if the email subscription is true or false
lastModifiedDateStr	String	The date on which a user was last modified
lastActiveDateStr	String	The date when the user is last active

Note: Apart from the above parameters, the user-defined parameters are also fetched.

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
messagingSvc.getSubscribedAudienceDetails(function(response) {
    kony.print("Get subscribed audience details Success: " +
JSON.stringify(response));
}, function(error) {
    kony.print("Failed to get subscribed  audience details: " +
JSON.stringify(error));
});
```

Response Status

Code	Description
Status 200	Array of Audience details
Status 400	No audience member found mapping to the given KSID
Status 500	Failed to process the request

39.3.11.13 Unsubscribe Audience

The **Unsubscribe Audience or Delete Audience** API deletes an Audience from Engagement server.

Syntax

```
unSubscribeAudience = function(successCallback, failureCallback)
```

Example

```
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
```

```
var messagingSvc = client.getMessagingService();
messagingSvc.unSubscribeAudience(function(response) {
    kony.print("UnSubscribe audience Success: " + JSON.stringify
(response));
}, function(error) {
    kony.print("UnSubscribe audience Failed: " + JSON.stringify
(error));
});
```

Response Status

Code	Description
Status 200	Audience member deleted successfully
Status 400	No Audience Member found mapping to the given KSID
Status 500	Server failed to process request

39.3.11.14 Update List of Beacons for a Device

The **Update List of Beacons for a Device** API updates the list of beacons for a device.

Syntax

```
updateListOfBeacons = function(uuId, major, minor, successCallback,
failureCallback, options)
```

Parameters

Input Parameter	Level-Two	Type	Description	Required
beacon		JSON	An array of beacon objects ***Note:*** You can only create one beacon at a time.	Yes
	uuid	string	Universally Unique Identifier Number (UUID) assigned to the Beacon. UUID contains 32 hexadecimal digits, split into 5 groups, and separated by dashes, for example, f7826da6-4fa2-4e98-8024-bc5b71e0893e By default, beacon format consists of three values: UUID, Major, Minor. Beacons broadcast their IDs, which can be recognized by mobile apps to trigger specific actions.	Yes
	major	string	Major ID is a major identifier of a Bluetooth beacon.	Yes
	minor	string	Minor ID is a minor identifier of a Bluetooth beacon.	Yes
options		JSON	Map for optional parameters	Optional

Options keys

Key	Type	Description	Required
ufid	string	The User Friendly Identifier or UFID is used when you subscribe to Kony Fabric Engagement Services. Based on your requirement, you can provide an UFID. It is alphanumeric, for example xxx@kony.com or 2890XZCY. It can be used to map devices to the user using the value as a reconciliation key	optional
appid	alphanumeric	Unique ID assigned to an app	optional

Example

```
// Sample code to update list of Beacons
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
var options = {};
options["ufid"] = "<user_friendly_identifier>"; //configured in MF
server
options["appid"] = "<Unique ID assigned to an app>"
messagingSvc.updateListOfBeacons("uuid", "major", "minor", function
(response) {
    kony.print("Updated beacons list successfully: " + JSON.stringify
(response));
}, function(error) {
    kony.print("Updated beacons list failed: " + JSON.stringify
(error));
}, options);
```

Response Status

Code	Description
Status 200	Beacons updated successfully
Status 400	Invalid request format
Status 500	Server failed to process request

39.3.11.15 Get Rich Push Content

The **Get Rich Content** API fetches rich content from the Kony Fabric Engagement server.

The Get Rich Content API requires the **pushId** to retrieve the data. The pushId is fetched from the response of the push notification.

Syntax

```
getRichPushContent = function(pushId, successCallback,
failureCallback){
```

Parameters

Input Parameter	Type	Description	Required
pushID	number	Unique ID that identifies the push message.	Yes

Example

```
//sample code for Get Rich Push Content.
//Get an instance of SDK.
var client = kony.sdk.getCurrentInstance();
var messagingSvc = client.getMessagingService();
```

```
var pushId = "pushid";
messagingSvc.getRichPushContent(pushId, function(response) {
    kony.print("get rich push content success: " + JSON.stringify
(response));
    var htmlString = response.rawResponse;
    var dataConfig = {
        "mimeType": "text/html"
    };
    Form1.browser.loadData(htmlString, dataConfig);
}, function(error) {
    kony.print("get rich push content failed: " + JSON.stringify
(error));
});
```

Response Status

Code	Description
Status 200	Rich content
Status 400	One of the following error messages will be displayed: • Invalid push ID / provided • Rich message is not associated with this push
Status 500	Server failed to process request

39.3.12 Invoking a Metrics Service Object

When the JS SDK is initialized, it automatically collects various standard metrics from a client and the standard metrics will be accessible using the Standard Reports within Kony Fabric Console.

The JS SDK also provides the ability for a developer to send additional custom metrics from a client app to Kony Fabric back-end to capture additional information. These custom data sets will be accessible using the Custom Reporting feature within Kony Fabric Console where a business analyst can design and share reports using a combination of standard and custom metrics.

Additionally, the JS SDK provides an Events API that allows an app to track user actions within the app to gain insight into the user journey. The developer can send various standard events such as form entry, touch events, service requests, gestures, and errors. The developer can also send custom events to capture any application specific scenarios or transactions. These events can be analyzed within Kony Fabric Console by using the Standard Reports or user defined Custom Reports. For more details, refer to Custom Metrics and Reports Guide.

This section lists all `MetricsService` object APIs.

> ***Note:*** On SDK initialization, the Metrics Service is available with the variable name KNYMetricsService.

39.3.12.1 Configuring Application Events Reporting

The MetricsService class sets the configuration for APM event reporting.

* @param reportingMode{string }specifies the event reporting mode which can be currently only set to "Buffer"

* @param bufferAutoFlushCount{number }In case the reportingMode is set to Buffer, this property specifies the number of events to be buffered before flushing.

* @param maxBufferCount{number }In case the reportingMode is set to Buffer, this property specifies the maximum number of events that can be buffered. Events exceeding the maxBufferCount will be ignored.

* @Availability Applicable on All native Platforms (iOS, Android)

```
 //Sample code to set the configuration for application events.
KNYMetricsService.setEventConfig("Buffer", 50, 200);
```

setBatchSize

The **setBatchSize** API allows a developer to specify the batch size to be set to flush events. Default batch size is 50.

Parameters:

- @param batchSize {number}

```
//Sample code to set batch size to flush events
KNYMetricsService.setBatchSize(20);
```

setUserID

The **setUserID** API sets the user ID for the data gathered from an application. The user ID allows the data to be tracked on a user basis for broad analysis like how many different users used the application. It also helps to track activities of a specific user, which can help in seeing what activities were done before a crash, or what events led to a transaction not passing through. The user ID allows the same user to be tracked across different devices as well.

- @param setUserId {string }User ID to be passed

```
setUserID = function( /**string */ setUserId, ) {}
```

```
//Sample code to set up the user ID of application user
kony.setUserID("myUserID");
```

> ***Note:*** The UserID related to metrics. The UserID length cannot be more than 100 characters.

> ***Note:*** **KNYMetricsService.setUserId** has been removed from apps built with Kony Visualizer Enterprise.

sendEvent

The **sendEvent** API allows a developer to send event details from an application to a server for analytics and reporting purposes. The event data is added to a buffer and sent to the server as per configuration values set by the developer using setEventConfig API.

> ***Note:*** From Kony V8 SP3 and latest plug-in versions of 7.2, 7.3, V8.2 and above, the sendEvent API will not capture the network calls to image URLs as part of `serviceRequest/serviceResponse` events. This is applicable for both manual invocation from an application code and for events captured automatically from the Project Settings.

- @param eventType {string }specifies the event type. Can be one of the following constants. FormEntry, FormExit, Touch, ServiceRequest, ServiceResponse, Gesture, Orientation, Error, Exception, Crash, Custom
- @param eventSubType{string }specifies the sub type of event.
- @param formID{string }widget ID of the form where event happened.
- @param widgetID {string }widgetID of the widget on which the event happened.
- @param flowTag {string }flowTag to added for this event
- @param metaInfo {JSObject [Key value pairs]}event specific meta data
- @Availability Applicable on All native Platforms (iOS, Android)

```
sendEvent = function( /**string */ eventType, /**string */
eventSubType, /**string */ formID, /**string */ widgetID, /**string */
flowTag, /**JSObject [Key value pairs]*/ metaInfo) {}
```

```
//Sample code to send reports
KNYMetricsService.sendEvent("FormEntry", "frmHome", "frmHome",
"widgetID", "flowTag", {
```

```
    "key1": "value1"
});
```

> ***Note:*** This API is required to be used only if the application developer chooses to send their own custom events. All event types chosen for automatic event tracking from the **Metrics APM** tab in application properties or set using the setEventTracking API will automatically be tracked.

setFlowTag

The **setFlowTag** API sets an event flow tag to be associated with all new events that are reported by using the sendEvent API. The flow tag is used to ease searching event data in terms of application flows like loginflow, searchflow. The setFlowTag also helps in sorting and filtering data while building custom reports or running standard reports for the application events.

- @param flowtag {string }Flow tag name
- @Availability Applicable on All native Platforms (iOS, Android)

```
setFlowTag = function( /**string */ flowtag, ) {}
```

```
//Sample code for the setFlowTag API
KNYMetricsService.setFlowTag("MyFlowTag");
```

setEventTracking

The **setEventTracking** API sets the event types to be tracked.

- @param EventTypes {JSObject [Array]}An array of string constants which are valid event types This method must be called during the lifetime of the application to enable event tracking, otherwise the default behavior is not to track any events. An empty array or a null object as a parameter to this method results in not to track any of the events.
- @Availability Applicable on All native Platforms (iOS, Android)

- For example, while entering critical flow, the following is a sample code:

```
KNYMetricsService.setEventTracking(["FormEntry", "Error",
"Crash", "FormExit", "ServiceResponse"]);
```

- For example, while exiting critical flow, the following is a sample code:

```
KNYMetricsService.setEventTracking(["FormEntry", "Error",
"Crash"]);
```

> ***Note:*** The events set by using the setEventTracking API override any setting set from the application Project Settings at build time. So, you must set critical events like Error and Crash while using the API.

> ***Note:*** Supported values for setEventTracking are ["FormEntry","FormExit","Touch","ServiceRequest","ServiceResponse","Gesture","Orientation","Error","Crash"]

getEventTracking

The **getEventTracking** API gets the list of all event types that are being tracked currently.

- @Availability Applicable on All native Platforms (iOS, Android)

```
getEventTracking = function() {}
```

```
//Sample code for the getEventTracking API

var events = KNYMetricsService.getEventTracking();
```

clearFlowTag

The **clearFlowTag** API clears the currently set event flow tag.

- @Availability Applicable on All native Platforms (iOS, Android)

```
clearFlowTag = function() {}
```

```
//Sample code for the clearFlowTag API
KNYMetricsService.clearFlowTag();
```

getFlowTag

The **getFlowTag** API gets the currently set event flow tag.

- @Availability Applicable on All native Platforms (iOS, Android)

```
getFlowTag = function() {}
```

```
//Sample code for the getFlowTag API

var flowtag = KNYMetricsService.getFlowTag();
```

reportError

The **reportError** API enables an app to report an error event to metrics server.

- @param errorCode{string } errorCode can be nil if not applicable.
- @param errorType {string }errorType can be nil if not applicable.
- @param errorMessage {string } errorMessage can be nil if not applicable.
- @param errorDetails {string }errorDetails is a json string that can have key value pairs for the following keys errfile, errmethod, errline, errstacktrace, errcustommsg, errcrashreport, formID, widgetID, and flowTag.
- @Availability Applicable on All native Platforms (iOS, Android)

```
reportError = function( /**string */ errorCode, /**string */
errorType, /**string */ errorMessage, /**string */ errorDetails, ) {}
```

```
//Sample code for the reportError API
KNYMetricsService.reportError("1234", "SpecificError", "custom error
message", "{errfile:file.js}");
```

> ***Note:*** This API is required to be used only if the application developer chooses to send their own error events. If Error event type is chosen for supported platforms via application properties or setEventTracking API, error tracking will automatically be done.

reportHandledException

The **reportHandledException** API enables apps to report a handled exception event. Application developers can use this API to report handled exceptions in the application code.

- @param exceptionCode {string }exceptionCode can be nil if not applicable.
- @param exceptionType {string }string type of exception, such as Eval Error or syntax error. The exceptionType can be nil if not applicable.
- @param exceptionMessage {string }exceptionMessage can be nil if not applicable.
- @param exceptionDetails {string }exceptionDetails is a JSON string that can have key value pairs for the following keys exceptionfile, exceptionmethod, exceptionline, exceptionstacktrace, formID, widgetID, and flowTag.
- @Availability Applicable on All native Platforms (iOS, Android)

```
reportHandledException = function( /**string */ exceptionCode,
/**string */ exceptionType, /**string */ exceptionMessage, /**string
*/ exceptionDetails, ) {}
```

```
//Sample code to send exception to metrics server
KNYMetricsService.reportHandledException("1234", "SpecificException",
"custom exception message", "{errfile:file.js}");
```

flushEvents

The **flushEvents** API allows a developer to force events to be sent to the server. The entire current event buffer is loaded and sent to the server for processing.The flushEvents API is used as an override to send event data to a server before the value configured in seteventconfig for autoflushcount is reached.

- @Availability Applicable on All native Platforms (iOS, Android)

```
flushEvents = function() {}
```

```
//Sample code for the flushEvents API
KNYMetricsService.flushEvents();
```

sendCustomMetrics

The **sendCustomMetrics** API allows the developer to send custom metrics from the application.

The custom metrics keys should already be registered in Kony Fabric Console for the application before data is sent from the application.

- @param groupId{JSObject [Array]}

  formID length cannot be more than 250 characters.

- @param data {data}data to be send
- @Availability Applicable on All native Platforms (iOS, Android)

```
sendCustomMetrics = function( /**JSObject [Array]*/ groupId, /**data*/
data, ) {}
```

```
//Sample code for the sendCustomMetrics API
KNYMetricsService.sendCustomMetrics("formID", {
    "metric": "metricdata"
});
```

For more details about custom metrics and reports, refer to Custom Metrics and Reports Guide.

39.3.12.2 Global Error Handler

The **uncaughtexceptionhandler** APIs are available only for iOS and Android native apps built from Kony Visualizer.

setUncaughtExceptionHandler

This API allows a developer to register a callback function to be invoked for uncaught JS exception.

```
kony.lang.setUncaughtExceptionHandler("< Callback function for
ErrorHandling >");
```

```
//Sample code for the setUncaughtExceptionHandler API
function myErrorHandler() {}
kony.lang.setUncaughtExceptionHandler(myErrorHandler);
```

getUncaughtExceptionHandler

This API allows a developer to get the function that is currently registered for the uncaught JS exceptions.

```
//Sample code for the getUncaughtExceptionHandler API
kony.lang.getUncaughtExceptionHandler()
```

39.3.12.3 Event Details

For all event details, timestamp of event and session identifier values are automatically filled by MBaaS Client SDK, as part of the reportEvent, reportError and reportHandledException API calls. In case of automatically captured events, flowTag is also automatically filled with the currently set flowTag. The following are event specific details to be used while interfacing with MBaaS SDK while manually invoking sendEvent API to send event data.

FormEntry

- API to be used - sendEvent
- evtType - FormEntry
- FormID - the value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - null

FormExit

- API to be used - sendEvent
- evtType - FormExit
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - Dictionary (hash table) that contains the following key value pairs:
    - formdur - Duration spent in form in milliseconds. Optional.

Touch

- API to be used - sendEvent
- evtType - Touch
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - value of the ID property of the widget on which the touch happened
- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget)
- metadata - null

ServiceRequest

- API to be used - sendEvent
- evtType - ServiceRequest (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType
  Service ID - in case of service invoking Kony middleware
  URL - in case of other requests
- metadata - null

ServiceResponse

- API to be used - sendEvent
- evtType - ServiceResponse (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middleware
  URL - in case of other requests
- metadata
  JSON object (hash table) containing following key value pairs:
  - opstatus - Optional
    returned by Kony servers
  - httpcode - HTTP status code
  - resptime - time taken to get the response.

Gesture

- API to be used - sendEvent
- evtType - Gesture (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget where the gesture happened
- WidgetID - value of the ID property of the widget on which the gesture happened

- evtSubType [String]
  `GESTURETYPE_NUMBEROFINPUTS_DIRECTION`
  For example, two finger left swipe - SWIPE_2_LEFT
- metadata - null

Orientation

- API to be used - sendEvent
- evtType - Orientation (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String] - any one of the below constants is used
  - PORTRAIT_TO_LANDSCAPE
  - LANDSCAPE_TO_PORTRAIT
- metadata - null

Error

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ErrorCode - Optional
- metadata
  JSON object (hash table) containing following key value pairs:

- errcode - Optional
- errmsg - Optional
- errfile - Optional
- errmethod - Optional
- errstacktrace - Optional
- errcustommsg - Optional

HandledException

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ExceptionCode - Optional
- metadata
  JSON object (hash table) containing following key value pairs:
  - exceptioncode - Optional
  - exceptionev - Optional
  - exceptionmsg - Optional
  - exceptionfile - Optional
  - exceptionmethod - Optional
  - exceptionstacktrace - Optional
  - exceptioncustommsg - Optional

Crash

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String]
- metadata
  JSON object (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errline - Optional
  - errstacktrace - Optional
  - errcrashreport - Optional

Custom

- API to be used - sendEvent
- evtType - Custom
- FormID - any supplied form ID
- WidgetID - any supplied widget ID
- evtSubType - any supplied event subtype
- metadata - string or a dictionary

39.3.13 Invoking an Object Service

Kony supplies you with programmatic access to backend data, both online and offline. To gain access, do the following:

1. Acquire a current instance of your object service.
2. Use the object service instance together with data transfer objects to communicate as needed with your backend, either online or offline with a local cache.

To know more about how to acquire a current instance of your object service, refer to getObjectService Method documentation.

To know more about the methods that act on the Kony Fabric endpoint directly, refer to OnlineObjectServiceClass documentation.

To know more about the methods that act on the local sync database, refer to OfflineObjectServiceClass documentation.

To know more about the usage of the data transfer objects, refer to Data Transfer Objects documentation.

39.3.13.1 getObjectService Method

The **getObjectService Method** gets the current instance of the object service. The getObjectService method is invoked on the sdk instance; init must already have been successfully run before invoking this method.

Syntax

```
getObjectService(serviceName,{serviceType});
```

Parameters

Parameter	Type	Description
serviceName	string	Name of the object service
serviceType	JSON object	One of the following: • "access" : "online" *[default]* • "access" : "offline" • "access" : "registeredObjectServiceName" *[This is the name of the object service defined in the application]*

Response

Returns the object service instance based on the value specified in serviceType. The default value is "online", which returns an "OnlineObjectService Class" below instance. A value of "offline" returns an "OfflineObjectService Class" on page 1399 instance.

39.3.13.2 OnlineObjectService Class

Provides methods that perform operations acting on the Kony Fabric endpoint, including basic CRUD, metadata, and binary-related functions. An instance of OnlineObjectService is returned by the getObjectService Method when the second parameter specifies {"access":"online"}.

Methods

The following methods are used by the OnlineObjectService class and its instantiations.

create Method

Creates an object in the Kony Fabric endpoint.

39.3.13.3 Syntax

```
create(options, successCallback, failureCallback);
```

39.3.13.4 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject" which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds, with the primary key of the created object
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.5 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});
var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("field1", "value1");
var options = {
    "dataObject": dataObject
};
objSvc.create(options,
    function(response) {
        kony.print("Record created: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in record creation: " + JSON.stringify
(error));
    } );
```

> ***Note:*** When using object services for SAP, the general norm is to have character field values stored in upper case. However, if you need to pass in mixed/lower case values for an SAP field, ensure that this field is designated as mixed case in the SAP Add-in LDB workbench.

update Method

Updates an object in the Kony Fabric endpoint.

39.3.13.6 Syntax

```
update(options, successCallback, failureCallback);
```

39.3.13.7 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.8 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("field1", "value1");
dataObject.addField("primaryKeyField", "value");
```

```
var options = {
    "dataObject": dataObject
};

objSvc.update(options,
    function(response) {
        kony.print("Record updated: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in record update: " + JSON.stringify
(error));
    });
```

delete Method

Deletes an object in the Kony Fabric endpoint.

39.3.13.9 Syntax

```
deleteRecord(options, successCallback, failureCallback);
```

39.3.13.10 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds, with the number of records deleted
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.11 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("field1", "value1");
dataObject.addField("primaryKeyField", "value");

var options = {
    "dataObject": dataObject
};
objSvc.deleteRecord(options,
    function(response) {
        kony.print("Record deleted: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in record deletion: " + JSON.stringify
(error));
    });
```

customVerb Method

Performs a custom operation on an object in the Kony Fabric endpoint.

39.3.13.12 Syntax

```
customVerb(verbName, options, successCallback, failureCallback);
```

39.3.13.13 Parameters

Parameter	Description
verbName	Name of the custom verb defined in the Kony Fabric console

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.14 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("field1", "value1");
dataObject.addField("primaryKeyField", "value");

var options = {
    "dataObject": dataObject
};

objSvc.customVerb("verbName", options,
    function(response) {
        kony.print("Custom operation performed: " + JSON.stringify
(response));
    },
    function(error) {
        kony.print("Error in custom operation:" + JSON.stringify
(error));
    });
```

fetch Method

Fetches an object from the server.

39.3.13.15 Syntax

```
fetch(options, successCallback, failureCallback);
```

39.3.13.16 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410. This instance must have the property *selectQueryObject*, which is an instance of kony.sdk.dto.SelectQuery, in order to fetch records based on the given criteria.
successCallback	Function invoked when the method succeeds, with the number of records fetched
failureCallback	Function invoked when fetch fails, with cause of failure

39.3.13.17 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
var odataUrl = "$filter=fieldname eq value";
dataObject.odataUrl = odataUrl;


var options = {
```

```
    "dataObject": dataObject
};

objSvc.fetch(options,
    function(response) {
        kony.print("record: " + response["records"]);
    },
    function(error) {
        kony.print("Failed to fetch: " + JSON.stringify(error));
    });
```

getMetadataOfAllObjects Method

Gets the metadata associated with the objects defined in the service from the server.

39.3.13.18 Syntax

```
getMetadataOfAllObjects(options, successCallback, failureCallback);
```

39.3.13.19 Parameters

Parameter	Description
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.20 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
         "access": "online"
     });
```

```
objSvc.getMetadataOfAllObjects({},
function(response) {
  kony.print("Metadata: " + JSON.stringify(response));
},
function(error) {
  kony.print("Error in metadata: " + JSON.stringify(error));
});
```

getMetadataOfObject Method

Gets the metadata associated with an object defined in the service from the server.

39.3.13.21 Syntax

```
getMetadataOfObject(objectName, options, successCallback,
failureCallback);
```

39.3.13.22 Parameters

Parameter	Description
objectName	The name of the desired object as defined in the service
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds, with the number of records gotten
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.23 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});
```

```
objSvc.getMetadataOfObject("objectName", {},
    function(response) {
        kony.print("Metadata: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in metadata: " + JSON.stringify(error));
    });
```

Parameter	Description
options (Refer the Options below)	JSON object with the mandatory parameter **dataObject**, which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410,
fileDownloadStartedCallback	Callback to be invoked after file download start, this callback is optional.
chunkDownloadCompletedCallback	callback invoked after each successful chunk download, this is invoked only when streaming is true. This callback is mandatory if streaming is **true**.
fileDownloadCompletedCallback	Callback invoked after successful file download, with the file path. This is manditory if streaming is **false**.
downloadFailureCallback	Callback invoked in case of download failure.

Key	Value	**Mandatory**	**Default Value**
dataObject	Instance of kony.sdk.dto.DataObject	Yes	-
streaming	Boolean flag to determine whether the data needed to be given to chunks or a file after download.	No	false

queryParams	provision for user to specify additional query params that need to be sent in the download call.	No	null
headers	Provision for sending custom headers	No	null

Parameter	Description
options (Refer the Options below)	JSON object with the mandatory parameter **dataObject**, which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410,
binaryAttributeName	Binary field name in the object.
fileDownloadStartedCallback	Callback to be invoked after file download start, this callback is optional.
chunkDownloadCompletedCallback	callback invoked after each successful chunk download, this is invoked only when streaming is true. This callback is mandatory if streaming is **true**.
fileDownloadCompletedCallback	Callback invoked after successful file download, with the file path. This is manditory if streaming is **false**.
downloadFailureCallback	Callback invoked in case of download failure.

Key	Value	**Mandatory**	**Default Value**
dataObject	Instance of kony.sdk.dto.DataObject	Yes	-
streaming	Boolean flag to determine whether the data needed to be given to chunks or a file after download.	No	false

queryParams	provision for user to specify additional query params that need to be sent in the download call.	No	null
headers	Provision for sending custom headers	No	null

getBinaryContent Method

Gets binary content on the server.

39.3.13.24 Syntax

```
getBinaryContent(options, successCallback, failureCallback);
```

39.3.13.25 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410, and "binaryAttrName", which is the binary field name in the object
successCallback	Function invoked when the operation succeeds, with the number of records gotten
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.26 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
//primary key details to get media object
```

```
dataObject.addField("primary_key", "value");

objSvc.getBinaryContent({
        "dataObject": dataObject,
        "binaryAttrName": "data"
    },
    function(response) {
        kony.print("binary content is : " + JSON.stringify(response));
        frmBinary.downloadImg.isVisible = true;
        frmBinary.downloadImg.base64 = response;
    },
    function(error) {
        kony.print("failed to get binary data : " + JSON.stringify
(error));
    });
```

createBinaryContent Method

Creates binary content on the server.

39.3.13.27 Syntax

```
createBinaryContent(options, successCallback, failureCallback);
```

39.3.13.28 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410, and "binaryAttrName", which is the binary field name in the object
successCallback	Function invoked when the operation succeeds, with the number of records created

Parameter	Description
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.29 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("name", "imgName");
dataObject.addField("data", "binaryText");

objSvc.createBinaryContent({
        "dataObject": dataObject,
        "binaryAttrName": "data"
    },
    function(response) {
        kony.print("Binary content created: " + JSON.stringify
(response));
    },
    function(error) {
        kony.print("Failed: " + JSON.stringify(error));
    });
```

updateBinaryContent Method

Updates binary content on the server.

39.3.13.30 Syntax

```
updateBinaryContent(options, successCallback, failureCallback);
```

39.3.13.31 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410, and "binaryAttrName", which is the binary field name in the object
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.32 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "online"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("name", "imgName");
dataObject.addField("data", "binaryText");

objSvc.updateBinaryContent({
        "dataObject": dataObject,
        "binaryAttrName": "data"
    },
    function(response) {
        kony.print("Binary content updated: " + JSON.stringify
(response));
    },
    function(error) {
```

```
        kony.print("Failed: " + JSON.stringify(error));
    });
```

39.3.13.33 OfflineObjectService Class

Provides methods that perform operations acting on the sync database, including basic CRUD, metadata, and binary-related functions. An instance of OfflineObjectService is returned by the getObjectService Method when the second parameter specifies {"access":"offline"}.

Methods

The following methods are used by the OfflineObjectService class and its instantiations.

create Method

Creates an object offline in the sync database.

39.3.13.34 Syntax

```
create(options, successCallback, failureCallback);
```

39.3.13.35 Parameters

Parameter	Description
options	JSON object with the following parameters: "dataObject" - a mandatory parameter which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410 "httpRequestOptions" an optional parameter used for configuring thin rich network calls. -or- ""xmlHttpRequestOptions" - an optional parameter used for configuring thin thin network calls. The possible values for these options are: "timeoutIntervalForRequest" - specifies the timeout interval. "timeoutIntervalForResource" - specifies the timeout interval.
successCallback	Function invoked when the operation succeeds, with the primary key of the created object
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.36 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "offline"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("field1", "value1");
```

```
var options = {
    "dataObject": dataObject,
    "httpRequestOptions": {
        "timeoutIntervalForRequest": 60,
        "timeoutIntervalForResource": 600
    }
};

objSvc.create(options,
    function(response) {
        kony.print("Record created: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in record creation: " + JSON.stringify
(error));
    });
```

> ***Note:*** When using object services for SAP, the general norm is to have character field values stored in upper case. However, if you need to pass in mixed/lower case values for an SAP field, ensure that this field is designated as mixed case in the SAP Add-in LDB workbench.

update Method

Updates an object offline in the sync database (local store).

39.3.13.37 Syntax

```
update(options, successCallback, failureCallback);
```

39.3.13.38 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.39 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "offline"
});


var dataObject = new kony.sdk.dto.DataObject("ObjectName")
dataObject.addField("field1", "value1");
dataObject.addField("primaryKeyField", "value");

var options = {
    "dataObject": dataObject
};

objSvc.update(options,
    function(response) {
        kony.print("Record updated: " + JSON.stringify(response));
    },
    function(error) {
```

```
        kony.print("Error in record update: " + JSON.stringify
(error));
    });
```

delete Method

Deletes an object offline in the sync database.

39.3.13.40 Syntax

```
deleteRecord(options, successCallback, failureCallback);
```

39.3.13.41 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds, with the number of records deleted
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.42 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "offline"
});

var dataObject = new kony.sdk.dto.DataObject("ObjectName");
dataObject.addField("field1", "value1");
dataObject.addField("primaryKeyField", "value");

var options = {
```

```
    "dataObject": dataObject
};

objSvc.deleteRecord(options,
    function(response) {
        kony.print("Record deleted: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in record deletion: " + JSON.stringify
(error));
    });
```

getMetadataOfAllObjects Method

Gets the metadata associated with the objects defined in the service from the local store.

39.3.13.43 Syntax

```
getMetadataOfAllObjects(options, successCallback, failureCallback);
```

39.3.13.44 Parameters

Parameter	Description
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.45 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "offline"
```

```
});

objSvc.getMetadataOfAllObjects({},
    function(response) {
        kony.print("Metadata: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in metadata: " + JSON.stringify(error));
    });
```

getMetadataOfObject Method

Gets the metadata associated with an object defined in the service from the local store.

39.3.13.46 Syntax

```
getMetadataOfObject(objectName, options, successCallback,
failureCallback);
```

39.3.13.47 Parameters

Parameter	Description
objectName	The name of the desired object as defined in the service
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds, with the number of records returned
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.48 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
```

```
 });

objSvc.getMetadataOfObject("objectName", {},
function(response) {
  kony.print("Metadata: " + JSON.stringify(response));
},
function(error) {
  kony.print("error in metadata: " + JSON.stringify(error));
});
```

executeSelectQuery Method

Executes a Select query on the sync database.

39.3.13.49 Syntax

```
executeSelectQuery(queryString, successCallback, failureCallback);
```

39.3.13.50 Parameters

Parameter	Description
queryString	SQL Select query string
successCallback	Function invoked when the operation succeeds, with the number of records operated on
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.51 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "offline"
});
```

```
var queryString = "select * from table";
objSvc.executeSelectQuery(queryString,
    function(response) {
        kony.print("Metadata: " + JSON.stringify(response));
    },
    function(error) {
        kony.print("Error in metadata::" + JSON.stringify(error));
    });
```

getBinaryContent Method

Gets binary content from the sync database.

39.3.13.52 Syntax

```
getBinaryContent(options, successCallback, failureCallback);
```

39.3.13.53 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1410
successCallback	Function invoked when the operation succeeds, with the number of records gotten
failureCallback	Function invoked when the operation fails, with cause of failure

39.3.13.54 Example

```
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
    "access": "offline"
});
```

```
var dataObject = new kony.sdk.dto.DataObject("ObjectName");
//primary key details to get object
dataObject.addField("primarykeyfield1", "value1");

var options = {};
options["dataObject"] = dataObject;
//binary column name to get the binary data
options["binaryAttrName"] = "data";
options["responsetype"] = "base64string/filepath";

//filepath is default if it is not set by developer
objSvc.getBinaryContent(options, successcallback, errorcallback);

function successcallback(response) {
    //response will contain base64string or filepath
    //  based on responsetype
    //value provided by the developer.
    var content = response["records"][0]["data"];
    kony.print("Binary Content: " + JSON.stringify(content));
}

function errorcallback(error) {
    kony.print("Error in getBinary: " + JSON.stringify(error));
}
```

39.3.13.55 Data Transfer Objects

A data transfer object (DTO) represents a database element in a programmatic context. Each DTO is used in various Object Services API methods. However, DTOs are used only for offline Object Services, not for online Object Services. The DTOs comprise several classes:

- kony.sdk.dto.Column Class
- kony.sdk.dto.DataObject Class

kony.sdk.dto.Column Class

This class represents a column in a database table, which is in turn represented by a kony.sdk.dto.Table object. It is used when creating a query.

> ***Note:*** This class is used only for offline Object Services, not for online Object Services.

Constructors

The kony.sdk.dto.Column class has one constructor.

39.3.14 kony.sdk.dto.Column(table, columnName) Constructor

39.3.14.1 Signature

```
kony.sdk.dto.Column(table, columnName)
```

39.3.14.2 Parameters

Parameter name	Type	Description
table	kony.sdk.dto.Table	The table containing the column
columnName	string	The name of the column

Example

```
var dataObject = new kony.sdk.dto.DataObject("ObjectName");
var tblDto = new kony.sdk.dto.Table("ObjectName", "ObjectName",
false);
var selQuery = new kony.sdk.dto.SelectQuery("serviceName", tblDto);
var colObj = new kony.sdk.dto.Column(tblDto, "field1");
var colObj2 = new kony.sdk.dto.Column(tblDto, "field2");
var colObj3 = new kony.sdk.dto.Column(tblDto, "field3");
selQuery.addColumn(colObj);
selQuery.addColumn(colObj2);
```

```
selQuery.addColumn(colObj3);
dataObject.setSelectQueryObject(selQuery);
```

kony.sdk.dto.DataObject Class

This class represents a data object in the Object Service. An instantiation of this class is required as a parameter in many methods of the OnlineObjectService Class and the OfflineObjectService Class.

Constructors

The kony.sdk.dto.DataObject class has one constructor.

39.3.15 kony.sdk.dto.DataObject(string objectName)

Parameter	Type	Description
objectName	string	Name of object defined in the object service

Methods

The kony.sdk.dto.DataObject class includes the following methods.

- addChildDataObject(child)
- addField(fieldName, value)
- setOdataUrl(URL)
- setRecord(record)
- setSelectQueryObject(queryObject)

39.3.16 addChildDataObject(child) Method

Adds another DataObject as a child.

39.3.16.1 Signature

```
addChildDataObject(child)
```

39.3.16.2 Parameters

Parameter	Type	Description
child	kony.sdk.dto.DataObject	The DataObject to be added as a child

39.3.17 addField(fieldName, value) Method

Adds a field in the data object.

39.3.17.1 Signature

```
addField(fieldName, value)
```

39.3.17.2 Parameters

Parameter	Type	Description
fieldName	string	The name of the field being added.
value	string	The value of the added field

39.3.18 setOdataUrl(URL) Method

Specifies the Odata URL.

39.3.18.1 Signature

```
setOdataUrl(URL)
```

39.3.18.2 Parameters

Parameter	Type	Description
URL	string	The Odata URL to set.

39.3.19 setRecord(record) Method

Specifies a record in the data object.

39.3.19.1 Signature

```
setRecord(record)
```

39.3.19.2 Parameters

Parameter	Type	Description
record	JSON object	The record to be set.

39.3.20 setSelectQueryObject(queryObject) Method

Sets the specific query object to be used in the query.

39.3.20.1 Signature

```
setSelectQueryObject(queryObject)
```

39.3.20.2 Parameters

Parameter	Type	Description
queryObject	kony.sdk.dto.SelectQuery	The object that contains the query.

Example

```
// **Simple Object**
var dataObject = new kony.sdk.dto.DataObject("ObjectName");

var record = {};
record["field1"] = "value1";
record["field2"] = "value2";
```

```
record["field3"] = "value3"; //sets the record to the dataObject
dataObject.setRecord(record);

// **Complex Object*
var parentDataObject = new kony.sdk.dto.DataObject("parentObject");
var record = {};
record["field1"] = "value1";
record["field2"] = "value2";
record["field3"] = "value3";
//sets the record to the parent.
parentDataObject.setRecord(record);

var childDataObject = new kony.sdk.dto.DataObject("childObject");
var record = {};
record["field4"] = "value3";
record["field5"] = "value5";
record["field6"] = "value6"; //sets the record to the child
childDataObject.setRecord(record);

parentDataObject.addChildDataObject(childDataObject); // **Adding
OdataUrl**

var dataObject = new kony.sdk.dto.DataObject("objectName");
dataObject.setOdataUrl("$filter=fieldname eq value");
```

39.3.21 Cache Service Response for Integration and Object Service

The Cache service response feature provides a non -persistent cache for storing response from integration and object service operations. The stored response can be retrieved by cacheID. Consequently, you can avoid making additional network calls for read-only data.

<table>
<tr><td>Usage</td><td>Optional parameters <code>useCache</code>, <code>cacheID</code> and <code>expiryTime</code> are used to save and retrieve the responses from the cache for both Integration Service and Object Service.

Note: Cache size is 100 by default .

<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>useCache</td><td>Boolean</td><td>The parameter to enable the cache service. If the parameter is enabled, SDK generates a unique key for the operation and cache the result with it. The same key is sent in response JSON with key cacheID. Existing entries with the same key will be overridden.</td></tr>
<tr><td>cacheID</td><td>String</td><td>Key to set or retrieve the mapping from the cache. If the key is not found, the requested operation is performed and the response is mapped to a key in the cache.</td></tr>
<tr><td>expiryTime</td><td>Non zero positive Integer</td><td>Expiry time in seconds for the key in cache. If expired, responses are removed from the cache. The parameter can be set while adding a new response to cache, and it cannot be updated further.</td></tr>
</table>
</td></tr>
<tr><td>Features Supported</td><td>Integration Service, Object Service fetch</td></tr>
</table>

<table>
<tr><td>Platforms Supported</td><td>IDE</td></tr>
<tr><td>Scope</td><td>The scope of the feature is limited to the application level.</td></tr>
<tr><td>API</td><td>

Integration

```
IntegrationClient.invokeOperation(operationName, headers, params, successCallback, failureCallback, options)
```

<table>
<tr><th>Parameter</th><th>Type</th></tr>
<tr><td>Operation Name</td><td>String</td></tr>
<tr><td>headers</td><td>JSON</td></tr>
<tr><td>params</td><td>JSON</td></tr>
<tr><td>successCallback</td><td>Function</td></tr>
<tr><td>failueCallback</td><td>Function</td></tr>
<tr><td>options</td><td>JSON</td></tr>
</table>

Object Service

```
ObjectClient.fetch(options, successCallback, failureCallback)
```

<table>
<tr><th>Parameter</th><th>Type</th></tr>
<tr><td>option</td><td>JSON</td></tr>
<tr><td>successCallback</td><td>function</td></tr>
<tr><td>failureCallback</td><td>function</td></tr>
</table>

</td></tr>
</table>

> ***Note:*** You can remove any key from cache explicitly using `new kony.sdk.ClientCache().remove(cacheID)`.

Sample Code

```
//Integration Service
function cache_integration_svc_response() {
    var options = {};
    var responseCacheKey;
    options["useCache"] = true;

    //Optional
    //options["cacheID"] = "cachedEmployeeDetailsResponse";
    //options["expiryTime"] = 30;  - Assign expiry time for the
current response if desired

    try {
        var serviceName = "integration_service_name";
        // Get an instance of SDK
        var client = kony.sdk.getCurrentInstance();
        var integrationSvc = client.getIntegrationService
(serviceName);
        var operationName = "operation_name";
        integrationSvc.invokeOperation(operationName, null, null,
            function(response) {
                // Save the response cache key
                responseCacheKey = response["cacheID"];
                kony.print("Success: " + JSON.stringify(response));

            },
            function(error) {
                kony.print("Failure: " + JSON.stringify(error));
            },
            options);
```

```
        // Try fetching employee details using cacheID
        options = {};
        options["useCache"] = true;
        options["cacheID"] = responseCacheKey;

        integrationSvc.invokeOperation(operationName, null, null,
            function(response) {
                // Save the response cache key
                responseCacheKey = response["cacheID"];
                kony.print("Success: " + JSON.stringify(response));

            },
            function(error) {
                kony.print("Failure: " + JSON.stringify(error));
            },
            options);

    } catch (exception) {
        alert("Integration Service Connect Failed " +
exception.message);
    }
}

//Object Service
function cache_object_svc_fetch() {
    var responseCacheKey;

    var objSvc = kony.sdk.getCurrentInstance().getObjectService
(serviceName, {
        "access": "online"
    });
```

```
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");

    var options = {
        "dataObject": dataObject
    };

    options["useCache"] = true;

    //Optional
    //options["cacheID"] = "cachedEmployeeDetailsResponse";
    //options["expiryTime"] = 30;  - Assign expiry time for the
current response if desired

    objSvc.fetch(options,
        function(response) {
            responseCacheKey = successRes["cacheID"];
            kony.print("Success: " + JSON.stringify(response));

        },
        function(error) {
            kony.print("Failure: " + JSON.stringify(error));
        });

    // Try fetching employee details using cacheID
    options = {};
    options["useCache"] = true;
    options["cacheID"] = responseCacheKey;

    objSvc.fetch(options,
        function(response) {
            responseCacheKey = successRes["cacheID"];
            kony.print("Success: " + JSON.stringify(response));
```

```
        },
        function(error) {
            kony.print("Failure: " + JSON.stringify(error));
        });
}
```

39.3.22 Kony Logger

39.3.22.1 Introduction

Kony Logger is introduced to persist logs based on six log levels:

- Trace
- Debug
- Info
- Warn
- Error
- Fatal

The logger feature provides three ways of providing logs to output: console, file and network. The logger feature is defined in the `namespace kony.logger`. Each log will be printed as per the below syntax:

```
 [LoggerInstanceName][AppID AppVersion][TimestampTimeZone][LogLevel]
[SessionID][ThreadInformation][DeviceId][DeviceOSInfo][FileName]
[ClassName][MethodName][LineNo]Message
```

39.3.22.2 Scope

The Kony Logger feature is available for Kony Visualizer, Android, iOS, and Windows 10.

39.3.22.3 Properties

`kony.logger` exposes the below properties:

`currentLogLevel`

`currentLogLevel` sets the log level of the logger to any of the six mentioned levels. setting to a certain level will allow the logs of that level and above to be persisted. For instance, if the `currentLogLevel` is set to **Error**, only the error and fatal log statements are printed.

Syntax	**JavaScript**: `kony.logger.currentLogLevel`
Possible Values	The below log levels may be set to the current log level: • `kony.logger.logLevel.ALL` • `kony.logger.logLevel.NONE` • `kony.logger.logLevel.TRACE` • `kony.logger.logLevel.DEBUG` • `kony.logger.logLevel.INFO` • `kony.logger.logLevel.WARN` • `kony.logger.logLevel.ERROR` • `kony.logger.logLevel.FATAL`
Default Value	`kony.logger.logLevel.NONE`
Read or Write	Yes (Read and Write)
JavaScript Example	`kony.logger.currentLogLevel = kony.logger.logLevel.DEBUG;`

39.3.22.4 APIs

Multiple instances of the logger can be created using APIs. The console and the file can be activated as a part of the configuration, if required. The below APIs are introduced to create logger, activate persistors and generate the logs.

createFilePersistor()

The `createFilePersistor()` API is used to create the configuration for the file persistor. This configuration can be used with the `loggerConfig` object (For more information, refer `addPersistor` API).

> ***Note:*** In Visualizer Android, if you want activate file persistor and view the logs in the file, you need to add `WRITE_EXTERNAL_STORAGE` under **Permissions** tab.

Signature	`JavaScript: createFilePersistor`
Parameters	N/A
Return Type	The API return a `fileConfig` object. The below properties have been exposed on the file persistor. • `maxFileSize` • `maxNumberOfLogFiles`

JavaScript Example	`var persistor1 = new kony.logger.createFilePersistor();`
	`persistor1.maxFileSize = 100;`
	`persistor1.maxNumberOfLogFiles = 15;`

Platform Availability	Available on IDE, Android, iOS, and Windows 10

maxFileSize

The `maxFileSize` API is used to define the maximum size of the files created in bytes.

Syntax	`maxFileSize`
Type	Number
Possible Values	Positive Integer
Default Value	10000
Read or Write	Only Write
JavaScript Example	`var persistor1 = new kony.logger.createFilePersistor(); persistor1.maxFileSize = 10000;`

`maxNumberOfLogFiles`

To define the `maxNumberOfLogFiles` the app can have at a maximum during any instant. The `filePersistor` is a rolling file appender i.e. the data of latest configured files is preserved.

Syntax	`maxNumberOfLogFiles`
Type	Number
Possible Values	Non-zero, Positive Integer
Default Value	10
Read or Write	Only Write
JavaScript Example	`var persistor1 = new kony.logger.createFilePersistor(); persistor1.maxNumberOfLogFiles = 20;`

createLoggerConfig()

`CreateLoggerConfig()` is used to create the configuration object for the logger. This object is used in the `createNewLogger()` API. The object has various properties exposed to define our configuration. To apply the configuration, you need to use `setConfig()` API . When you pass the object through `createNewLogger()` API , configuration object is immediately applied .

Signature	**Javascript**: `createLoggerConfig`
Parameters	N/A
Return Type	The API returns a `loggerConfig` Object. All the properties are static and affect all the logger instances. The below properties have been exposed. • `bytesLimit` • `statementsLimit` • `timeFormat` • `timeZone` • `overrideConfig` • `logLevel` • `addPersistor`

JavaScript Example	**Example 1**:
	`var loggerConfig = new kony.logger.createLoggerConfig();`
	
	Example 2:
	`var loggerConfig = new kony.logger.createLoggerConfig(); loggerConfig.bytesLimit = 10000; loggerConfig.statementsLimit = 10; loggerConfig.timeFormat = "dd-MM-yyyy HH.mm.ss.SSS"; loggerConfig.timeZone = "UTC"; loggerConfig.overrideConfig = true; loggerConfig.logLevel= kony.logger.logLevel.INFO.value;`

Platform Availability	IDE, Android, iOS, and Windows 10

`bytesLimit`

The console logs will be printed as soon as the logging methods are invoked. It is performance intensive to persist the logs one by one on the file. The `bytesLimit` can be used to accumulate the logs until the `bytesLimit` is reached and then flushed onto the file.

Syntax	`bytesLimit`
Type	Number
Possible Values	Any positive integer
Default Value	10000
Read or Write	Only Write
JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig(); loggerConfig.bytesLimit = 20000;`

statementsLimit

The console logs will be printed as soon as the logging methods are invoked. It is performance intensive to persist the logs one by one on the file. The `statementsLimit` can be used to accumulate the logs until the specified number of statements are reached and then flushed onto the file.

Syntax	`statementsLimit`
Type	Number
Possible Values	Any non-negative integer
Default Value	20
Read or Write	Only Write
JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig();` `loggerConfig.statementsLimit = 30;`

timeFormat

Each log is printed in a specified format. The format includes the `timestamp`. This property is used to specify the syntax of the `timeFormat`.

Syntax	timeFormat
Type	String
Possible Values	Any valid time format pattern
Default Value	"dd-mm-yyyy HH.mm.ss.SSS"

Read or Write	Only Write
JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig(); loggerConfig.timeFormat = "dd-MM-yyyy HH.mm.ss.SSS";`

timeZone

Each log is printed in a specified format. The format includes the `timestamp`. This property is used to specify the syntax of the `timeZone`.

Syntax	timeZone
Type	String
Possible Values	UTC, LocalTime
Default Value	UTC
Read or Write	Only Write
JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig(); loggerConfig.timeZone = "UTC";`

overrideConfig

All the properties are exposed by `loggerConfig`. Changing the property and using `setConfig()` API will affect all the logger instances created. `overrideConfig` is used to specify if the configuration should override the existing configuration.

Syntax	overrideConfig
Type	Bool

Possible Values	True or False
Default Value	True
Read or Write	Only Write
JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig(); loggerConfig.overrideConfig = false;`

logLevel

The `logLevel` property works similar to `currentLogLevel` property. There are six log levels in the `logLevel` property.

Syntax	`logLevel`

Possible Values	`kony.logger.logLevel.ALL`
	`kony.logger.logLevel.NONE`
	`kony.logger.logLevel.TRACE`
	`kony.logger.logLevel.DEBUG`
	`kony.logger.logLevel.INFO`
	`kony.logger.logLevel.WARN`
	`kony.logger.logLevel.ERROR`
	`kony.logger.logLevel.FATAL`

Default Value	`kony.logger.logLevel.NONE`
Read or Write	Only Write
JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig(); loggerConfig.logLevel = kony.logger.logLevel.INFO.value;`

addPersistor

The property can be used to add file persistor through `loggerConfig` object.

Syntax	addPersistor
Possible Values	Object of type filePersistor
Default Values	N/A
Read or Write	Only Write

JavaScript Example	`var loggerConfig = new kony.logger.createLoggerConfig();`
	`var filePers = new kony.logger.createFilePersistor();`
	`filePers.maxNumberOfLogFiles = 15;`
	`loggerConfig.addPersistor(filePers);`

createNewLogger() API

Signature	`createNewLogger`
Parameters	This API takes three parameters: LoggerName, LoggerConfig **loggerName**: The `loggerName` is a string used to identify the instance of the `loggerObject`. ***Note:*** This is a mandatory value. **loggerConfiguration**: The `loggerConfiguration` is the configuration object. For more information, refer `createLoggerConfig()`.

Return Type	Returns a logger object on which various functions such as `trace()`, `debug()`, `info()`, `warn()`, `error()` and `fatal()` APIs are exposed.
JavaScript Example	`var lConfig = new kony.logger.createLoggerConfig();` `var loggerObj = new kony.logger.createNewLogger("AndroidLoggerDemo", lConfig);`
Platform Availability	IDE, Android, iOS, and Windows 10

`activatePersistors()`

Unless we activate persistors, logs will not be pushed to the corresponding persistors. This is a mandatory step and no persistor is activated by default.

Signature	`activatePersistors`
Parameters	The API takes one parameter: persistor name. This parameter could be either a console or a file.
Return Type	N/A

JavaScript Example	`kony.logger.activatePersistors(kony.logger.consolePersistor)`
	`kony.logger.activatePersistors(kony.logger.filePersistor)`
	`kony.logger.activatePersistors(kony.logger.consolePersistor | kony.logger.filePersistor)`

Platform Availability	IDE, Android, iOS, and windows 10

> ***Note:*** Console Persistor logs will not be available in Visualizer Windows 10 and Native Windows 10.

`deactivatePersistors()`

If you want to make sure the logs are not pushed to a corresponding persistor or any combination of them, use `deactivatePersistors()` API.

Signature	`deactivatePersistors`
Parameters	The API takes one parameter: persistor name. This parameter could be either a console or a file or any combination of these.
Return Type	N/A

JavaScript Example	`kony.logger.deactivatePersistors (kony.logger.consolePersistor)`
	`kony.logger.deactivatePersistors (kony.logger.filePersistor)`
	`kony.logger.deactivatePersistors (kony.logger.consolePersistor | kony.logger.filePersistor)`

Platform Availability	IDE, Android, iOS, and windows 10

`trace()` API

The `trace()` API is used to trace log level messages.

Signature	trace
Parameters	The API takes one parameter - a string input log that must be persisted.

Return Type	N/A
JavaScript Example	`Var lconfig = new kony.logger.createNewLoggerConfig(); loggerobj = new kony.logger.createNewLogger ("AndroidLoggerDemo", lconfig); loggerobj.trace("Message to be added in Trace");`
Platform Availability	IDE, Android, iOS, and windows 10

`debug()` API

The `debug()` API is used to log the debug level messages:

Signature	debug
Parameters	The API takes one parameter - a string input log that must be persisted
Return Type	N/A
JavaScript Example	`Var lconfig = new kony.logger.createNewLoggerConfig(); loggerobj = new kony.logger.createNewLogger ("AndroidLoggerDemo", lconfig); loggerobj.debug("Message to be added in Debug");`
Platform Availability	IDE ,Android , iOS and windows 10

`info()` API

The `info()` API is used to log the info level messages.

Signature	info

Parameters	The API takes one parameter - a string input log that must be persisted.
Return Type	N/A
JavaScript Example	`Var lconfig = new kony.logger.createNewLoggerConfig(); loggerobj = new kony.logger.createNewLogger ("AndroidLoggerDemo", lconfig); loggerobj.info("Message to be added in Info");`
Platform Availability	IDE, Android, iOS, and windows 10

`error()` API

The `error()` API is used to log the error messages:

Signature	error
Parameters	The API takes one parameter - a string input log that must be persisted.
Return Type	N/A
JavaScript Example	`Var lconfig = new kony.logger.createNewLoggerConfig(); loggerobj = new kony.logger.createNewLogger ("AndroidLoggerDemo", lconfig); loggerobj.error("Message to be added in Error");`
Platform Availability	IDE, Android, iOS, and windows 10

warn() API

The `warn()` API is used to log the warning messages:

Signature	warn
Parameters	The API takes one parameter - a string input log that must be persisted
Return Type	N/A
JavaScript Example	`Var lconfig = new kony.logger.createNewLoggerConfig(); loggerobj = new kony.logger.createNewLogger ("AndroidLoggerDemo", lconfig); loggerobj.warn("Message to be added in Warn");`
Platform Availability	IDE, Android, iOS, and windows 10

fatal() API

The `fatal()` API is used to log the fatal messages:

Signature	fatal
Parameters	The API takes one parameter - a string input log that must be persisted
Return Type	N/A
JavaScript Example	`Var lconfig = new kony.logger.createNewLoggerConfig(); loggerobj = new kony.logger.createNewLogger ("AndroidLoggerDemo", lconfig); loggerobj.fatal("Message to be added in fatal");`

Platform Availability	IDE, Android, iOS, and windows 10

`flush()` API

The `flush()` API is used to flush all the statements onto the activated persistors regardless whether the `bytesLimit` or `statementsLimit` are met.

Signature	`flush`
Parameters	N/A
Return Type	N/A
JavaScript example	`kony.logger.flush();`
Platform Availability	IDE, Android, iOS, and windows 10

`setconfig()` API

The `setconfig()` API is a convenience API used to set `LoggerConfig` (mentioned earlier). Since `loggerConfig` is a singleton, If you set the `loggerConfig` once, it is applied across all the logger objects.

Signature	`setConfig`
Parameters	`loggerConfig`
Return Type	N/A
JavaScript Example	`Var logConfig = new kony.logger.createLoggerConfig ();` `kony.logger.setConfig(logConfig);`
Platform Availability	IDE, Android, iOS, and windows 10

setPersistConfig()

The `setPersistConfig()` API is used to change `persistorConfig` or set it to fresh, when it is not set through `loggerConfig`. As the persistors are singletons, they will have impact on all `loggerObjects` when set.

Signature	`setPersistorConfig`
Parameters	persistor
Return Type	N/A
JavaScript Example	`Var filePers = new kony.logger.createFilePersistor(); filePers.maxNumberOfFiles = 1; kony.logger.setPersistorConfig(filePers);`
Platform Availability	IDE, Android, iOS, and Windows 10

39.3.22.5 App Logger

The default logger Instance created can be used to write log statements. If the user wants to create additional logger instances he is free to do so. The logger name used for appLogger is **konyLogger**.

Usage

```
 kony.logger.appLogger.error("Message to be logged at error log
level");
kony.logger.appLogger.debug("Message to be logged at error log
level");
kony.logger.appLogger.trace("Message to be logged at error log
level");
```

39.3.22.6 Usage Guidelines/Restrictions/Examples

```
 //Create Logger Configuration
var lConfig = new kony.logger.createLoggerConfig();
//Create FilePersistor
Var persistor1 = new kony.logger.createFilePersistor();
//Add Persistor to the loggerConfig
lConfig.addPersistor(persistor1);
//Create Logger Object
loggerObj = new kony.logger.createNewLogger("UserLogs", lConfig);
kony.logger.activatePersistors(kony.logger.consolePersistor);
kony.logger.activatePersistors(kony.logger.filePersistor);
kony.logger.currentLogLevel = kony.logger.logLevel.INFO;
//Print Statements
for(var i=0;i<20;i++){
var msg = "statement" + i;
loggerObj.debug("Message in debug level" + msg);
loggerObj.trace("Message in trace level" + msg);
loggerObj.fatal("Message in Fatal level" + msg);
loggerObj.info("Message in Info level" + msg);
loggerObj.warn("Message in warn level" + msg);
loggerObj.error("Message in error level" + msg);
}
```

Note:

- Unlike the file and console persistors, the network persistor is always enabled and can be managed from Kony Fabric Admin Console. Do not create/activate/deactivate the network persistor from the client application code. For more details on configuring Network Persistor, refer to the **Log Level by Client Filters** section in the Standard Logs doc.
- To manage middleware logs, refer to App Services > Logs.
- Running the logger on Android devices that run on Android Lollipop or earlier versions with less than 1GB RAM may cause memory issues. These memory issues are at the Android

OS level and have been declared obsolete by Google. For more information, refer to the Issue Tracker.

39.3.23 Binary APIs

Kony Fabric SDK offer a different set of API to work with binary data on integration and object service. The APIs allow uploading and downloading binary data from different data providers such as Amazon S3, Box etc.

Online binary API allows downloading the files as a whole (typically small image files), in chunks (large pdfs or videos) or in streaming mode (raw bytes).

Note: For SPA/Desktop Web, SDK supports Binary APIs from release version V8 SP3 for Box adapter.

For Android and iOS, the Binary APIs are supported from V8 SP2 for Box adapter.

39.3.23.1 Integration Service APIs

Download API

The `getBinaryData` API allows users to download binary objects such as PDF, Document or any other files.

Syntax

```
getBinaryData(operationName, downloadParams, streaming, headers,
fileDownloadStartedCallback, chunkDownloadCompletedCallback,
fileDownloadCompletedCallback, fileDownloadFailureCallback, options);
```

Parameters

Parameter	Type	Description
operationName	String	Integration service operation that provides the binary details. The parameter is used as part of Integration Service.
downloadParams	JSON	Download related parameters required for downloading the binary.
Streaming	Boolean	Boolean flag to determine if the required data must be sent in chunks or a file after download.
headers		Provision to send custom headers.
fileDownloadCompletedCallback	True/False	The callback is invoked after successful file download with the file path. The parameter is mandatory if the streaming is false.
chunkDownloadCompletedCallback	True/False	The callback is invoked after each successful chunk download. The parameter is invoked when the streaming is true. The callback is mandatory if the streaming is true.
fileDownloadCompletedCallback	True/False	The callback is invoked after successful file download with the file path. The parameter is mandatory if the streaming is false.
fileDownloadFailureCallback		Callback invoked in case of download failure.

Parameter	Type	Description
options		Optional parameters are sent from this parameter. `ChunkSize` can be sent from this parameter. The parameter is used as part of Integration Service.

Options

Options	Type	Description	Mandatory
ChunkSize	String	ChunkSize option allows to download the binary in chunks of given size. If a value is not specified, the default value is used. Default size is 1048576.	No

> ***Note:*** It is mandatory to pass all its parameters defined on the service.

Sample Code

```
function fileDownloadStartedCallback(res) {
    //Callback
}

function chunkDownloadCompletedCallback(res) {
    //Callback
}

function fileDownloadCompletedCallback(res) {
    //Callback
}

function fileDownloadFailureCallback(err) {
    //Callback
```

```
}

var downloadParams = {};
var serviceName = "integration_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var operationName = "operation_name";

// Adapter specific parameters (Mentioned below)
var integrationSvc = client.getIntegrationService(serviceName);
var streaming = false;
integrationSvc.getBinaryData(operationName,
    downloadParams,
    streaming, {},
    //Provision for user defined headers
    fileDownloadStartedCallback,
    chunkDownloadCompletedCallback,
    fileDownloadCompletedCallback,
    fileDownloadFailureCallback);
```

> ***Note:*** For Android and iOS platforms, filePath or base64 chunk data is provided based on the streaming flag. For DesktopWeb Blob object is specified in successCallback.

Upload API

The `uploadBinaryData` API allows users to upload binary objects such as PDF, Document or any other files.

Syntax

```
uploadBinaryData(operationName, uploadParams,
fileUploadStartedCallback, chunkUploadCompletedCallback,
fileUploadCompletedCallback, fileUploadFailureCallback)
```

Parameters

Parameters	Type	Description
operationName	String	Integration Service Operation which provides the binary details. The parameter is used as part of Integration Service.
uploadParams	JSON	upload related parameters required for uploading binary. The parameter is used as part of Integration Service.
fileUploadStartedCallback		The callback is invoked when the upload starts.
chunkUploadCompletedCallback		The callback is invoked when the upload of a chunk is completed. The parameter is applicable for backends that support chunked upload.
fileUploadCompletedCallback		The callback is invoked when the upload completes.
fileUploadFailureCallback		The callback is invoked when the upload fails (for any reason).

> ***Note:*** It is mandatory to specify either FilePath or rawBytes for upload in the `uploadParams` parameter. Both parameters are NOT allowed together.

FilePath must contain a fully qualified path of a valid file present on the device.

rawBytes contains the raw bytes to be uploaded. An obvious usage is capturing an image using camera and uploading the same.

fileObject is accepted for DesktopWeb. fileObject is the output of Browse API and is the instance of File.

Common Parameters

uploadParams	Type	Description	Mandatory
fileName	String	The valid file name of the file to be uploaded with the proper extension.	Yes
FilePath	String	Fully qualified path of a valid file present on the mobile device.	Yes (if rawBytes are not provided)
rawBytes	String	Raw bytes of file to be uploaded	Yes (if FilePath is not provided)
fileObject	File	Output of the Browser API provided by Kony or the browser is a reference to the file in the device.	Yes (for DesktopWeb)

Note: It is mandatory to pass all its parameters defined on the service.

Sample Code

```
function fileUploadStartedCallback(res) {
    //Callback
}

function chunkUploadCompletedCallback(res) {
    //Callback
}

function fileUploadCompletedCallback(res) {
    //Callback
}
```

```
function fileUploadFailureCallback(err) {
    //Callback
}

var uploadParams = {};

var serviceName = "integration_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var operationName = "operation_name";

// Adapter specific parameters (Mentioned below)
var integrationSvc = client.getIntegrationService(serviceName);

integrationSvc.uploadBinaryData(operationName,
    uploadParams,
    fileUploadStartedCallback,
    chunkUploadCompletedCallback,
    fileUploadCompletedCallback,
    fileUploadFailureCallback);
```

39.3.23.2 Object Service APIs

Download API

The **getBinaryData** API allows users to download binary objects such as PDF, Doc or any other files.

Syntax

```
getBinaryData(options,fileDownloadStartedCallback,
chunkDownloadCompletedCallback, fileDownloadCompletedCallback,
fileDownloadFailureCallback)
```

Parameters

Parameter	Type	Description
options	JSON	Accepts a JSON object on which the binary is downloaded.
fileDownloadStartedCallback	JSON	The callback is invoked after the file download starts.
chunkDownloadCompletedCallback	True/False	The callback is invoked after each successful chunk download. The parameter is invoked when the streaming is true. The callback is mandatory if the streaming is true.
fileDownloadCompletedCallback	True/False	The callback is invoked after successful file download with the file path. The parameter is mandatory if the streaming is false.
fileDownloadFailureCallback	True/False	The callback is invoked in case of download failure.

Options

Options	Type	Description	Mandatory	Default
dataObject	String	Instance of `kony.sdk.dto.DataObject`	Yes	
streaming	Boolean	Boolean flag to determine whether the data needed to be given to chunks or a file after download	No	False

Options	Type	Description	Mandatory	Default
queryParams		provision for the user to specify the additional query params to be sent in the download call.	No	Null
headers		Provision for sending custom headers.	No	Null
chunkSize		chunk size to download	No	1048576

Sample Code

```
var downloadParams = {};

// Adapter specific parameters (Mentioned below)
var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName");

var dataObject = new kony.sdk.dto.DataObject("binaryObjectName");
dataObject.setRecord(downloadParams);

objSvc.getBinaryData({
        "dataObject": dataObject
    },
    function(response) {
        kony.print("fileDownloadStartedCallback: " + JSON.stringify
(response));
    },
    function(response) {
        kony.print("chunkDownloadCompleteCallback: " + JSON.stringify
(response));
    },
    function(response) {
        kony.print("fileDownloadCompleteCallback: " + JSON.stringify
(response));
```

```
    },
    function(response) {
        kony.print("fileDownloadFailureCallback: " + JSON.stringify
(response));
    });
```

> ***Note:*** It is mandatory to pass all its parameters defined on the service.

> ***Note:*** For Android and iOS platforms, filePath or base64 chunk data is provided based on the streaming flag. For DesktopWeb Blob object is specified in successCallback.

Upload API

The **uploadBinaryData** API allows users to download binary objects such as PDF, Doc, or any other files.

Syntax

```
uploadBinaryData(options, fileUploadStartedCallback,
chunkUploadCompletedCallback, fileUploadCompletedCallback,
fileUploadFailureCallback)
```

Parameters

Parameter	Type	Description
options	JSON	The JSON object, based on the binary uploaded.
fileUploadStartedCallback		The callback is invoked when the upload starts.
chunkUploadCompletedCallback		The callback is invoked when the upload of a chunk is completed. The parameter is applicable for backends that support chunked upload.

Parameter	Type	Description
fileUploadCompletedCallback		The callback is invoked when the upload completes.
fileUploadFailureCallback		The callback is invoked when the upload fails (for any reason). The JSON object specified under options is a `kony.sdk.dto.DataObject` containing relevant parameters for upload.

Options

Options	Type	Description	Mandatory
dataObject		Instance of `kony.sdk.dto.DataObject`	Yes

Common Parameters

uploadParams	Type	Description	Mandatory
fileName	String	The valid file name of the file to be uploaded with the proper extension.	Yes
FilePath	String	Fully qualified path of a valid file present on the mobile device.	Yes (if rawBytes are not provided)
rawBytes	String	Raw bytes of file to be uploaded	Yes (if FilePath is not provided)
fileObject	File	Output of the Browser API provided by Kony or the browser is a reference to the file in the device.	Yes (for DesktopWeb)

> ***Note:*** It is mandatory to pass all its parameters defined on the service.

Sample Code

```
var uploadParams = {};
var client = kony.sdk.getCurrentInstance();
var objectSvc = konyClient.getObjectService("serviceName");
var dataObject = new kony.sdk.dto.DataObject("binaryObjectName");
dataObject.setRecord(uploadParams);

objectSvc.uploadBinaryData({
        "dataObject": dataObject
    },
    function(response) {
        kony.print("fileUploadStartedCallback: " + JSON.stringify
(response));
    },
    function(response) {
        kony.print("chunkUploadCompleteCallback: " + JSON.stringify
(response));
    },
    function(response) {
        kony.print("fileUploadCompleteCallback: " + JSON.stringify
(response));
    },
    function(response) {
        kony.print("fileUploadFailureCallback: " + JSON.stringify
(response));
    });
```

39.4 JavaScript SDK

These steps show how to download JS SDK files and initialize JS client.

- Prerequisites
- Downloading Kony Plain JS SDK
- Initializing the JS Client SDK
- Invoking an Identity Service
- Invoking an Integration Service
- Invoking a Configuration Service
- Invoking a Logic Service
- Invoking a Metrics Service
- API Reference

  To view the API Reference for Plain JS, click Kony JS docset.

39.4.1 Prerequisites

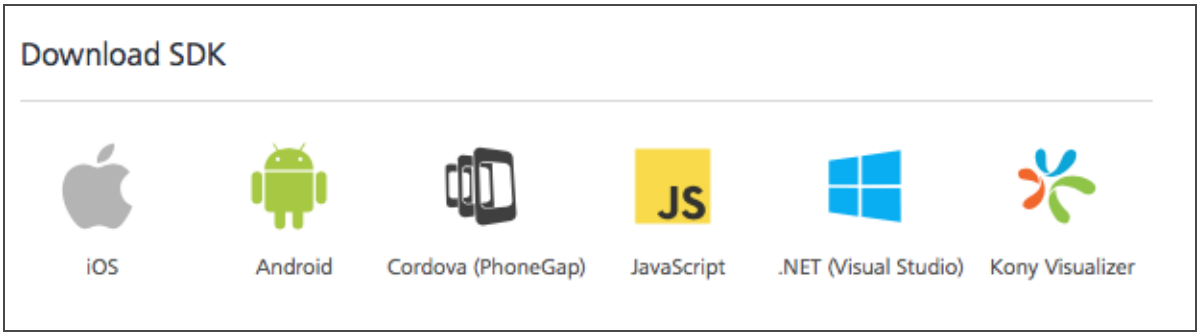
Kony JS supports the following Web Browsers:

- Firefox 31.0
- Google Chrome 36.0
- Internet Explorer 10.1
- Opera 23.0
- JQuery 2.1.1 or higher

39.4.2 Downloading Kony Plain JS SDK Files

To download Plain JS SDK, follow these steps:

1. In the Kony Fabric console, navigate to **Apps** > **SDKs**, and click **JavaScript**. The system prompts you to save the zip file in your local system.



2. Save the `kony-plainJS-sdk.zip` file in your local system.
3. Extract `kony-plainJS-sdk.zip` file that you just downloaded.
   The `kony-plainJS-sdk` folder contains the following files:
   - kony-sdk.js
   - kony-sdk.doc
   - LICENSE.txt
   - version.txt

39.4.3 Initializing the JS Client SDK

Initialize Kony Fabric client with the following code, and start using the services provided in Kony Fabric. The initialization method fetches the configuration from Kony Fabric and saves in the cache. Later, the application uses the cached configuration. It is a synchronous call.

When SDK is initialized, the Kony SDK registers a session and sends its information to the Kony Fabric Server. If the device is offline, or the server is not reachable, the session information persists on the device until it can successfully send the information to the Kony Fabric server.

For more information on application session, refer Standard Report Docs.

Note:

- The sessions created by Kony Fabric's JavaScript SDKs are interactive.
- Based on the browser settings, some browsers store the response of a call in the cache. When another request is made to the same URL, the browser sends the response from the cache instead of directing the call to the server.

  To disable response caching, set the **Pragma header** (as a global request header) to **no-cache** by using the **setGlobalRequestParam()** API.

39.4.3.1 init

```
//Sample code to initialize Kony Fabric Client
var appkey = < your - app - key > ;
var appsecret = < your - app - secret >
var serviceURL = < your - service - url > ;

var client = new kony.sdk();
//set Pragma Header to disable the use of response cache in browsers.
client.setGlobalRequestParam("Pragma", "no-cache",
client.globalRequestParamType.headers);
client.init(appkey, appsecret, serviceURL, function(response) {
    console.log("Init success");
}, function(error) {
    console.log("Init Failure");
});
```

39.4.4 Invoking an Identity Service

You can use the following methods for an identity service.

- Login with provider type as Basic
- Login with provider type as OAuth/SAML
- Login with provider type as OAuth 2.0 with Deep link URL

- Get Backend Token
- User Profile
- Get Provider Name
- Get Provider Type
- Logout

39.4.4.1 Login with provider type as Basic

```
 // Sample code to authenticate to Kony Fabric Client

var authClient = null;
var providerName = < your - provider - name > ;
var username = < uername - for -your - provider > ;
var password = < password - for -your - provider > ;
try {
    authClient = client.getIdentityService(providerName);
} catch (exception) {
    console.log("Exception" + exception.message);
}
authClient.login({
    "userid": username,
    "password": password
}, function(response) {
    console.log("Login success" + JSON.stringify(response));
}, function(error) {
    console.log("Login failure" + JSON.stringify(error));
});
```

> ***Note:*** The client is the kony.sdk(); object.

> ***Important:*** When you select Kony User Repository as the identity type, the system does not allow you to provide an identity name.
>
> To use Kony User Repository as authentication service, ensure that the value for `providerName` is set as `userstore`. If you set it with any other value (for example, Kony User Repository, User Store or Cloud Repository), the system throws an error.

39.4.4.2 Login with provider type as OAuth/SAML

```
 // Sample code to authenticate to Kony Fabric Client
var authClient = null;
var providerName = < your - provider - name > ;
try {
    authClient = client.getIdentityService(providerName);
} catch (exception) {
    console.log("Exception" + exception.message);
}
authClient.login({},
    function(response) {
        console.log("Login success" + JSON.stringify(response));
    }, function(error) {
        console.log("Login failure" + JSON.stringify(error));
    }
);
```

> ***Note:*** The client is the kony.sdk(); object.

39.4.4.3 Login with provider type as OAuth 2.0 with Deep link URL

```
// Sample code to authenticate to Kony Fabric Client

var authClient = null;
var providerName = <your-provider-name>;
var username = <username-for-your-provider>;
```

```
var password = <password-for-your-provider>;
var options = {};
options["noPopup"] = true; // This parameter in options will open the
login url in the same window. It will not open any pop up.
options["success_url"] = http: //mynativeapplication; // This is a
deeplink url, where the control will be redirected after login.

try {
    //The client is the kony.sdk();
    authClient = client.getIdentityService(providerName);
} catch (exception) {
    console.log("Exception" + exception.message);
}
authClient.login(options,
    function(response) {
        console.log("Login success" + JSON.stringify(response));
    }, function(error) {
        console.log("Login failure" + JSON.stringify(error));
    }
);
```

The sample above shows various parameters similar to the parameters of the Login with provider type as OAuth/SAML. The following two optional parameters are added further.

- **noPopup**: This parameter opens the login URL in the same window. It will not open any pop up.
- **success_url**: After the log in is successful, control is redirected to the URL (deep link URL).

Deep link URL is the URL that is registered to the application. After redirection, the client calls the method **handleDeeplinkCallback**. (A global function)

Method signature: `function handleDeeplinkCallback(params, successcallback, failurecallback)`

```
 // windown.onload gets called upon page reload, client can choose
which method should be called after the deeplink redirection. In this
sample onload gets called after redirection.

window.onload = function() {
    // Getting the current url to retrieve the query params.
    var url = window.location.href;
    var hashes = url.split("?")[1];
    var queryParams = {};

    if (hashes) {
        var hash = hashes.split('&');

        for (var i = 0; i < hash.length; i++) {
            var params = hash[i].split("=");
            queryParams[params[0]] = params[1];
        }

        handleDeeplinkCallback(queryParams, function success(resp) {
            alert("Client login success");
        }, function failure(resp) {
            alert("Client login failure");
        });
    }
};
```

For more information on deep links, click here.

39.4.4.4 Get Backend Token

```
// Sample code to get backend token for provider
var userid = < username_for_logged_in_provider > ;
var password = < password_for_logged_in_provider > ;
var forceFromServer = true / false;
```

```
authClient.getBackendToken(forceFromServer, {
    "userid": userid,
    "password": password
}, function(response) {
    console.log("Backend token is  :" + JSON.stringify(response));
}, function(error) {
    console.log("Failed to get backend token : " + JSON.stringify
(error));
});
```

Note: If forceFromServer is true, then the SDK fetches the token from the server. If forceFromServer is false, then the SDK gives you the token present in localStorage. Please note that only few backend providers such as Salesforce support refresh. If a backend provider does not support refresh, passing `forceRefreshFromServer=true` would result in empty response from this API.

Note: The authClient is the IdentityService object.

39.4.4.5 User Profile

```
 // Sample code to get user profile details
var forceFromServer = true / false;
authClient.getProfile(forceFromServer,
    function(response) {
        console.log("User profile is  :" + JSON.stringify(response));
    }, function(error) {
        console.log("Failed to fetch profile : " + JSON.stringify
(error));
    }
);
```

> ***Note:*** If forceFromServer is true, then the SDK fetches the token from the server. If forceFromServer is false, then the SDK gives you the token present in localStorage.

> ***Note:*** The authClient is the IdentityService object.

39.4.4.6 Get Provider Name

```
 // Sample code to get provider name
Var providerName = authClient.getProviderName();
```

> ***Note:*** The authClient is the IdentityService object.

39.4.4.7 Get Provider Type

```
 // Sample code to get provider type
Var providerType = authClient.getProviderType();
```

> ***Note:*** The authClient is the IdentityService object.

39.4.4.8 Logout

```


//Samplcode to logout from auth service

 var serviceName = "identity_service_name";
// Get an instance of SDK
var client = kony.sdk.getCurrentInstance();
var identitySvc = client.getIdentityService(serviceName);
var options = {};
options["slo"] = true;
identitySvc.logout(function(response)
```

```
{ kony.print("Logout success: " + JSON.stringify(response)); }
, function(error)

{ kony.print("Logout failure: " + JSON.stringify(error)); }
, options);
```

> ***Note:*** 1. Single Logout ("slo") property enables you to log out from all the identity providers. If you do not provide this value as options, only the specified identity provider will be logged out in multi-login scenario.
>
> 2. If you are upgrading to V8 ServicePack4 Fixpack 101 or later, middleware has to be upgraded to V8.4.3.16 or later version to avoid failure in logout.

39.4.5 Invoking an Integration Service

39.4.5.1 Invoke a Service

This API invokes an integration service that is configured in the Kony Fabric portal.

```
 // Sample code to fetch the integration service details
var integrationClient = null;
var serviceName = < your - service - name > ;
var operationName = < your - operation - name > ;
var params = {
    "your-input-keys": "your-input-values"
};
var headers = {
    "your-header-keys": "your-header-values"; //If there are no
headers,pass null

    try {
        integrationClient = client.getIntegrationService(serviceName);
    } catch (exception) {
        console.log("Exception" + exception.message);
```

```
    }
    integrationClient.invokeOperation(operationName, headers, params,
        function(result) {
            console.log("Integration Service Response is :" +
JSON.stringify(response));
        },
        function(error) {
            console.log("Integration Service Failure :" +
JSON.stringify(error));
        }
    );
```

> ***Note:*** The client is the kony.sdk(); object.

39.4.6 Invoking a Configuration Service

Admin Console provides an interface to define a set of key value pairs at the server level and the client level. You can find the interface at **Kony Fabric Console** > **Admin Console** > **Settings** > **Configurable Parameters** > **Client App Properties**.

You can configure the client specific properties in the server, so that client application can pull and consume the updated properties at runtime.

```
 // Sample code to fetch the client app properties from server.
function configTest() {
    var config = client.getConfigurationService();
    config.getAllClientAppProperties(
        function(res)

        {
            kony.print("client key value pairs retrieved : " +
JSON.stringify(res));
        },
        function(err)
```

```
        {
            kony.print(" Failed to retrieve client key value pairs : "
+ JSON.stringify(err));
        }
    );
}
```

39.4.7 Invoking a Logic Service

The **getLogicService** API creates an instance of logic service that is configured in the Kony Fabric portal. `logicClient = KNYMobileFabric.getLogicService(serviceName)`

The **invokeOperation** API invokes the backend operation using the object of logic service. The invoke operation function is as follows:

```
 // To access the operations defined in the KonyFabric portal for the
logic service:
logicClient.invokeOperation(serviceName, path, HttpMethodType,
headers, data, SuccessCB, FailureCB)
//Where,
serviceName = < your - service - name > ;
path = < path as defined in KonyFabric > ;
HttpMethodType = The type of call like "POST", "PUT", "GET", "DELETE"
header = the header like {
    "testget": "test"
}
data = the data field like {
    "LastName": "LNamePOST"
}
SuccessCB = Success CallBack
FailureCB = Failure CallBack
```

```
 // Sample code to fetch the logic service details
var response = konymbaas.getLogicService("mailapp");
alert("Response is :" + response.getLogicServiceUrl());
response.invokeOperation("mailapp", "/api/v1/contact", "POST", {
    "test": "test"
}, {
    "LastName": "LNamePOST"
}, function(res) {
    alert("successfully fetched logic service" + JSON.stringify(res));
}, function(res) {
    alert("error occurred in fetching logic service" + JSON.stringify
(res));
});
```

39.4.8 Invoking a Metrics Service Object

When the JS SDK is initialized, it automatically collects various standard metrics from a client and the standard metrics will be accessible using the Standard Reports within Kony Fabric Console.

The JS SDK also provides the ability for a developer to send additional custom metrics from a client app to Kony Fabric back-end to capture additional information. These custom data sets will be accessible using the Custom Reporting feature within Kony Fabric Console where a business analyst can design and share reports using a combination of standard and custom metrics.

Additionally, the JS SDK provides an Events API that allows an app to track user actions within the app to gain insight into the user journey. The developer can send various standard events such as form entry, touch events, service requests, gestures and errors. The developer can also send custom events to capture any app specific scenarios or transactions. These events can be analyzed within Kony Fabric Console by using the Standard Reports or user defined Custom Reports. For more details, refer to Custom Metrics and Reports Guide.

This section lists all `MetricsService` object APIs.

39.4.8.1 Create an instance of MetricsService

The **MetricsService** class sets the configuration for APM event reporting.

```
 //Sample code to create an instance of setEventConfig with variable
name "setEventConfig"
var metricsServiceObj = servicesObj.getMetricsService()
metricsServiceObj.setEventConfig("Instant", 50, 200);
```

setUserID

The **setUserID** API sets the user ID for the data gathered from an application. The user ID allows the data to be tracked on a user basis for broad analysis like how many different users used the application. It also helps to track activities of a specific user, which can help in seeing what activities were done before a crash, or what events led to a transaction not passing through. The user ID allows the same user to be tracked across different devices as well.

```
 //Sample code to set up the user ID of application uservar metrics
ServiceObj = servicesObj.getMetricsService()
metricsServiceObj.setUserID("myUserID");
```

> ***Note:*** The UserId related to metrics. The UserId length cannot be more than 100 characters.

sendEvent

The **sendEvent** API allows a developer to send event details from an application to server for analytics and reporting purposes. The event data is added to a buffer and sent to server as per configuration values set by the developer using setEventConfig API.

```
 //Sample code to send reports var metrics
ServiceObj = servicesObj.getMetricsService()
metricsServiceObj.sendEvent("FormEntry", "frmHome", "frmHome", null, {
    "key1": "value1"
});
```

setFlowTag

The **setFlowTag** API sets an event flow tag to be associated with all new events that are reported by using thesendEvent API. The flow tag is used to ease searching event data in terms of application flows like loginflow, searchflow. The setFlowTag also helps sorting and filtering data while building custom reports or running standard reports for the application events.

```
 //Sample code to setFlowTagmetrics
ServiceObj.setFlowTag("Myflowtag")
```

clearFlowTag

The **clearFlowTag** API clears the currently set event flow tag.

```
 //Sample code to clearFlowTag
var metricsServiceObj = servicesObj.getMetricsService()
metricsServiceObj.clearFlowTag();
```

getFlowTag

The **getFlowTag** API gets the currently set event flow tag.

```
 //Sample code to getFlowTag
var metricsServiceObj = servicesObj.getMetricsService()
var flowtag = metricsServiceObj.getFlowTag();
```

reportError

The **reportError** API enables an app to report an error event to metrics server.

```
 //Sample code to reportError
var metricsServiceObj = servicesObj.getMetricsService()
metricsServiceObj.reportError("1234", "SpecificError", "custom error
message", "{errfile:file.js}");
```

reportHandledException

The **reportHandledException** API enables apps to report a handled exception event.

```
 //Sample code to send exception to metrics server
var metricsServiceObj = servicesObj.getMetricsService()
metricsServiceObj.reportHandledException("1234", "SpecificException",
"custom exception message", "{errfile:file.js}");
```

flushEvents

The **flushEvents** API allows a developer to force events to be sent to the server. The entire current event buffer is loaded and sent to the server for processing.The flushEvents API is used as an override to send event data to server before the configured value or a service call that flushes the buffer.

```
 //Sample code to flushEvents
var metricsServiceObj = servicesObj.getMetricsService()
metricsServiceObj.flushEvents();
```

sendCustomMetrics

The **sendCustomMetrics** API sends custom metrics event.

```
 //Sample code to sendCustomMetrics
var metricsServiceObj = servicesObj.getMetricsService()
metricsServiceObj.sendCustomMetrics("formID", {
    "metric": "metricdata"
});
```

For more details about custom metrics and reports, refer to Custom Metrics and Reports Guide.

39.4.8.2 Event Details

For all event details, **ts** and **SID** values are automatically filled by MBaaS Client SDK, as part of the reportEvent, reportError and reportHandledException API calls. In case of automatically captured events, flowTag is also automatically filled with the currently set flowTag. Following are event specific details to be used while interfacing with MBaaS SDK.

FormEntry

- API to be used - sendEvent
- evtType - FormEntry
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - null

FormExit

- API to be used - sendEvent
- evtType - FormExit
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - Dictionary (hash table) that contains the following key value pairs:
    - formdur - Duration spent in form in milliseconds. Optional.

Touch

- API to be used - sendEvent
- evtType - Touch
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - value of the ID property of the widget on which the touch happened

- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget)
- metadata - null

ServiceRequest

- API to be used - sendEvent
- evtType - ServiceRequest (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata - null

ServiceResponse

- API to be used - sendEvent
- evtType - ServiceResponse (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata
  Dictionary (hash table) containing following key value pairs:
  - opstatus - Optional
    returned by Kony servers

- httpcode - HTTP status code
- resptime - time taken to get the reponse.

Gesture

- API to be used - sendEvent
- evtType - Gesture (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget where the gesture happened
- WidgetID - value of the ID property of the widget on which the gesture happened
- evtSubType [String]
  `GESTURETYPE_NUMBEROFINPUTS_DIRECTION`
  For example, two finger left swipe - SWIPE_2_LEFT
- metadata - null

Orientation

- API to be used - sendEvent
- evtType - Orientation (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String] - any one of the below constants is used
  - PORTRAIT_TO_LANDSCAPE
  - LANDSCAPE_TO_PORTRAIT
- metadata - null

Error

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ErrorCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errstacktrace - Optional
  - errcustommsg - Optional

HandledException

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ExceptionCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:

- exceptioncode - Optional
- exceptionev - Optional
- exceptionmsg - Optional
- exceptionfile - Optional
- exceptionmethod - Optional
- exceptionstacktrace - Optional
- exceptioncustommsg - Optional

Crash

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String]
- metadata
  Dictionary (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errline - Optional
  - errstacktrace - Optional
  - errcrashreport - Optional

Custom

- API to be used - sendEvent
- evtType - Custom
- FormID - any supplied form ID
- WidgetID - any supplied widget ID
- evtSubType - any supplied event subtype
- metadata - string or a dictionary

39.5 Cordova (PhoneGap) SDK

These steps show how to install Cordova (PhoneGap), install Node.js, and create an app in Cordova.

- Prerequisites
  - NodeJS
  - XCode
  - Cordova
- Downloading Kony PhoneGap SDK
- Installing Node.js

  Install Node.js from http://nodejs.org/
- Downloading Android SDK Files

  Download Android SDK http://developer.android.com/sdk/index.html
- Installing Cordova

  Install Cordova by following the steps from http://cordova.apache.org/docs/en/3.5.0//guide_cli_index.md.html#The%20Command-Line%20Interface
- Creating a Cordova App

- Accessing com.kony.sdk.doc for Cordova (PhoneGap)

  To view the Cordova (PhoneGap) Docset in native format, click Cordova docset.

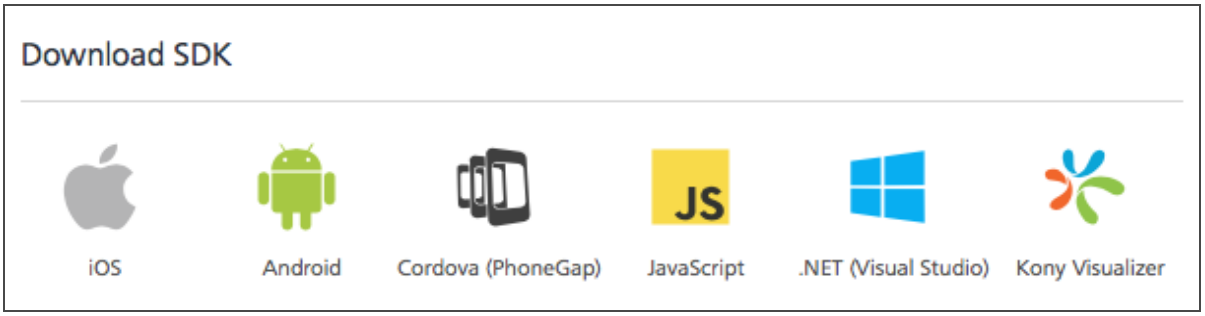
- Initializing the Cordova Client SDK
- Setting UserID
- Invoking an Identity Service
- Invoking an Integration Service
- Invoking a Configuration Service
- Invoking a Messaging Service
- Invoking a Sync Service
- Invoking a Reporting Service
- Invoking a Metrics Service
- Invoking an Object Service
- API Reference

  To view API Reference for Cordova, click Cordova docset.

39.5.1 Downloading Kony Cordova SDK Files

To download Kony Cordova SDK, follow these steps:

1. In the Kony Fabric console, navigate to **Apps** > **SDKs**, and click **Cordova (PhoneGap)**. The system prompts you to save the zip file in your local system.



2. Save the `kony-phonegap-sdk.zip` file in your local system.
3. Extract the `kony-phonegap-sdk.zip` file that you just downloaded.

   The `kony-phonegap-sdk` folder contains the following files:

   - com.kony.sdk
   - com.kony-sdk.doc
   - LICENSE.txt
   - version.txt

39.5.2 Creating a Cordova App

To create an application and build it using the Cordova command-line interface (CLI), follow these steps:

1. Run the following command in the command prompt:

```
npm install -g cordova
```

2. Now run the following command:

```
Cordova create AndroidPhoneGapApp com.kony MyApp
```

In this command, `AndroidPhoneGapApp` is the name of the project folder, `com.kony` is the namespace, and `MyApp` is the name of the project or application. The order of parameters is fixed.

> ***Note:*** If you do not specify **com.kony** and **MyApp** parameters, by default the namespace will as "com.hello" and creation of the project will require more time.
>
> The default app name is HelloCordova after the app gets installed.

3. Run the following command to navigate to the project folder.

```
cd AndroidPhoneGapApp
```

4. Run the following command to add the Android platform.

```
cordova platform add android
```

(You can add other platforms such as iOS and BlackBerry.)

5. Run the following command to install the plug-in device - such as UUID. This step is mandatory.

```
cordova plugin add cordova-plugin-device
```

6. Run the following command to install the plug-in, **InAppBrowser**. For example, if you use Oauth/SAML provider for authentication, this plug-in is required. This step is mandatory.

```
cordova plugin add cordova-plugin-inappbrowser
```

7. Run the following command to get the console logs. This step is optional.

```
cordova plugin add cordova-plugin-console
```

> ***Note:*** This plugin has been deprecated. For more information, click here.

8. Copy the `com.kony.sdk` from `kony-phonegap-sdk-sdk`, and paste `com.kony.sdk` into the app root folder.
9. Run the following command to add Kony Fabric plug-in into your applications:

```
cordova plugin add com.kony.sdk
```

10. Add the buttons or required widgets in the `index.html` file, and map them to the required functions by including the respective `JS Scripts`. Use Script tag, and include the required `.js` files.
11. Run the following command to deploy your app:
    - on Android platform:

```
cordova emulate android
or
cordova run android
```

    - on iOS platform:

```
cordova build ios
```

      After running this command, the system generates the xcode project under the `platform/ios` folder. Open the xcode and run the ios phonegap app.

39.5.3 Initializing the Cordova Client SDK

Initialize Kony Fabric client with the following code, and start using the services provided in Kony Fabric. The initialization method fetches the configuration from Kony Fabric, and saves it in the cache. Later, the application uses the cached configuration. It is a synchronous call.

39.5.3.1 init

```
//Sample code to initialize Kony Fabric Client
konySDKObject = new kony.sdk();
```

```
konySDKObject.init("<appKey>", "<appsecret>", "<serviceURL>",
successCallback, errorCallback);
```

39.5.4 Setting UserId

The `setUserId` allows a developer to set a unique ID to a Kony Fabric user. This user ID will be associated with each of the user login sessions that enables a developer to count unique users.

```
 // Sample code to set userid

konySDKObject.setUserId("userid");
```

39.5.5 Invoking an Identity Service

Identity service is used to authenticate user to use Kony Fabric integration service.

You can use the following methods for an Identity Service:

- Login with provider type as Basic
- Login with provider type as OAuth/SAML
- Get Profile
- Get Backend Token
- Logout

39.5.5.1 Login with provider type as Basic

```
 // Sample code to authenticate to   Kony Fabric Client
//For Kony user repository, use "userstore" in place of
"<IdentityName>"
var identity = konySDKObject.getIdentityService("<IdentityName>");
identity.login({
        "username": username,
        "password": password
```

```
    },
    successHandler, errorHandler);
```

Important: When you select Kony User Repository as the identity type, the system does not allow you to provide an identity name.

To use Kony User Repository as authentication service, ensure that the value for `providerName` is set as userstore. If you set providername with any other value (for example, Kony User Repository, User Store or Cloud Repository), the system throws an error.

39.5.5.2 Login with provider type as OAuth/SAML

```
 // Sample code to authenticate to Kony Fabric Client
var identity = konySDKObject.getIdentityService("<IdentityName>");
identity.login({}, successHandler, errorHandler);
```

39.5.5.3 Get Profile

```
 // Sample code to get profile information of the user
var fromserver = false;
identity.getProfile(fromserver, successHandler, errorHandler);
```

39.5.5.4 Get Backend Token

```

// Sample code to get backend token for provider
var fromserver = false;
identity.getBackendToken(fromserver, {}, successHandler,
errorHandler);
```

39.5.5.5 Logout

```
 // Sample code to logout from auth service

identity.logout(successHandler, errorHandler);
```

39.5.6 Invoking an Integration Service

This API invokes an integration service that is configured in the Kony Fabric portal.

```
 // Sample code to fetch the integration service details
var header = {
    "Content-Type": "application/json"
}; // In case no header is required, put null.
konySDKObject.getIntegrationService(serviceName).invokeOperation
(operationname, header, params,
    successHandler, errorHandler);
```

39.5.7 Invoking a Configuration Service

Admin Console provides an interface to define a set of key value pairs at the server and the client level. You can find the interface at **Kony Fabric Console** > **Admin Console** > **Settings** > **Configurable Parameters** > **Client App Properties**.

You can configure the Client specific properties any time in the server, so that client application can pull and consume the updated properties at runtime.

```
// Sample code to fetch the client app properties from server.
function configTest() {
    var config = konySDKObject.getConfigurationService();
    config.getAllClientAppProperties(
        function(res)

        {
            kony.print("client key value pairs retrieved : " +
JSON.stringify(res));
        },
        function(err)
```

```
        {
            kony.print(" Failed to retrieve client key value pairs : "
+ JSON.stringify(err));
        }
    );
}
```

39.5.8 Invoking a Messaging Service

A developer should register with Google Cloud Messaging (GCM) for Android services to get a deviceToken that is used to register with Kony Fabric Messaging. Also a developer should fetch the **deviceId** and **userfriendlyId** to create an instance of messaging service.

You can use the following methods for a messaging service:

- Register
- Unregister
- Update GeoLocation
- Fetch All Messages
- Mark Message as Read
- Fetch Message Content from Kony Fabric Messaging

39.5.8.1 Register

```
 // Sample code to register to messaging service
konySDKObject.getmessagingservice().register(osType, deviceId,
pnsToken, email, successHandler,
    errorHandler);
```

39.5.8.2 Unregister

```
 // Sample code to unregister from messaging service
konySDKObject.getmessagingservice().unregister(successHandler,
```

```
errorHandler);
```

39.5.8.3 Update GeoLocation

```
 // Sample code to update the geolocation
konySDKObject.getmessagingservice().updateGeoLocation(latitude,
longitude, locationName,
    successHandler, errorHandler);
```

39.5.8.4 Fetch All Messages

```
 // Sample code to fetch all messages
konySDKObject.getmessagingservice().fetchAllMessages(
    startIndex, pageSize, successHandler, errorHandler);
```

39.5.8.5 Mark Message as Read

```
 // Sample code to mark messages as read
konySDKObject.getmessagingservice().markMessageRead(fetchId,
successHandler, errorHandler);
```

39.5.8.6 Fetch Message Content from Kony Fabric Messaging

```
 // Sample code to fetch message content from Kony Fabric Messaging

konySDKObject.getmessagingservice().fetchMessageContent(fetchId,
successHandler, errorHandler);
```

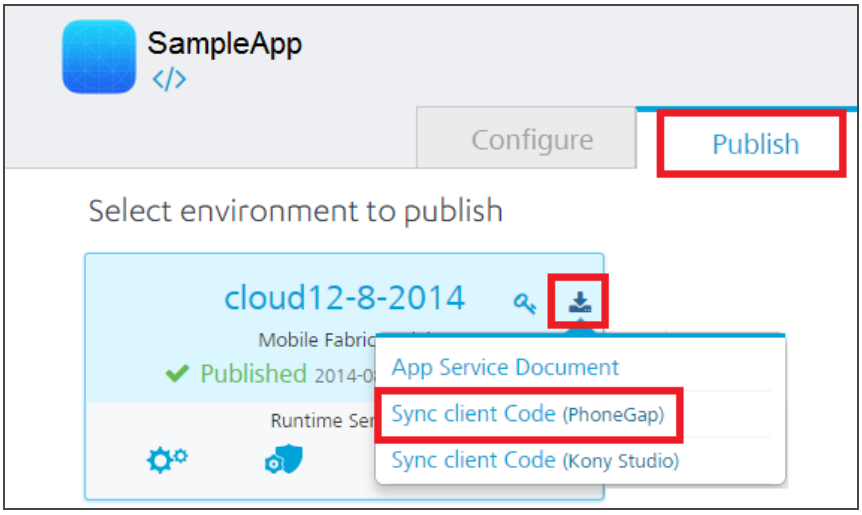
39.5.9 Invoking a Sync Service

39.5.9.1 Prerequisites

Download the required Kony Fabric Sync JS files for Cordova, and perform Init and relevant Sync operations.

To download Kony Fabric Sync JS files, follow these steps:

1. In your **Kony Fabric** portal > **Apps** > **Publish** tab, navigate to your **published app**.
2. Click **Download**, and then click **Sync client Code (PhoneGap)** shown below:



The system prompts you to save the zip (for example, Sync_kony.zip) file into your local system.

3. Extract the zip file that you just downloaded. The zip file contains `KonySyncDDL.js` and `KonySyncScopes.js`, and respective model files that are specific to an application.
4. Include these JS files in your app by using the following script tag. For example:

```
//say the files are KonySyncDDL.js and KonySyncScopes.js that are
placed in a folder called "js"

< script type = "text/javascript"
src = "js/KonySyncDDL.js" < /script>

<script type="text/javascript
" src="
js / KonySyncScopes.js "</script>
```

```
// say there is one model file called as CustomerModel.js

<script type="
text / javascript " src="
js / CustomerModel.js "</script>
```

Use the following code for instantiating sync object to call sync related APIs.

```
//Sample code to instantiate sync services

syncObj = konySDKObj.getSyncService();
```

The syncObj can call sync related APIs, as shown below:

39.5.9.2 Init

This method creates backend mobile SQLite database.

```
syncObj.init (syncInitSuccessCallback, syncInitFailureCallback);
```

For more information, refer to Sync Framework Documentation > sync.init.

39.5.9.3 startSession

This method initiates bi-directional sync between a server and mobile clientInformation.

```
syncObj.startSession(config);
```

Config: a JavaScript object containing various callbacks and parameters, and is required for every sync process. For more information, refer to Sync Framework Documentation > sync.startSession.

For other sync APIs, refer to Sync Framework Documentation.

To perform database operations, there are ORM APIs, which are exposed by Sync framework. For more details, click Sync Services ORM APIs.

39.5.10 Invoking a Reporting Service

Reporting service class helps a developer to get information on services used by number of users. The user access information helps for billing based on use of services.

39.5.10.1 Send Custom Metrics

```
 // Sample code to send custom metrics
var reportingService = konySDKObject.getReportingService();
reportingService.report("<your fid>", {
    "test metrics": "some test data"
});
```

39.5.11 Invoking a Metrics Service Object

When the Cordova SDK is initialized, it automatically collects various standard metrics from a client and the standard metrics will be accessible using the Standard Reports within Kony Fabric Console.

The Cordova SDK also provides the ability for a developer to send additional custom metrics from a client app to Kony Fabric back-end to capture additional information. These custom data sets will be accessible using the Custom Reporting feature within Kony Fabric Console where a business analyst can design and share reports using a combination of standard and custom metrics.

Additionally, the Cordova SDK provides an Events API that allows an app to track user actions within the app to gain insight into the user journey. The developer can send various standard events such as form entry, touch events, service requests, gestures and errors. The developer can also send custom events to capture any app specific scenarios or transactions. These events can be analyzed within Kony Fabric Console by using the Standard Reports or user defined Custom Reports. For more details, refer to Custom Metrics and Reports Guide.

39.5.11.1 Metrics Service Object APIs

This section lists all the Metrics Service object APIs.

clearFlowTag

The **clearFlowTag** method clears the previous event flow tag.

Syntax

```
clearFlowTag()
```

```
//Sample code for clearFlowTag
metricsServiceObj.clearFlowTag();
```

flushEvents

The **flushEvents** method allows a developer to force events to be sent to the server. The entire current event buffer is loaded and sent to the server for processing.The flushEvents API is used as an override to send event data to server before the value configured in seteventconfig for autoflushcount is reached.

Syntax

```
flushEvents()
```

```
//Sample code for flushEvents
metricsServiceObj.flushEvents();
```

getEventsInBuffer

The **getEventsInBuffer** method returns a list of the buffered events.

Syntax

```
getEventsInBuffer()
```

```
//Sample code for getEventsInBuffer
var events = metricsServiceObj.getEventsInBuffer();
```

getFlowTag

The **getFlowTag** method gets the currently set event flow tag.

Syntax

```
getFlowTag()
```

```
 //Sample code for getFlowTag
string flowtag = metricsServiceObj.getFlowTag();
```

getSessionId

The **getSessionId** method gets the current session Id.

Syntax

```
 getSessionId()
```

```
 //Sample code for getSessionId
var sessionId = metricsServiceObj.getSessionId();
```

getUserId

The **getUserId** method gets the Id of the current user.

Syntax

```
 getUserId()
```

```
 //Sample code for getUserId
var  userId = metricsServiceObj.getUserId();
```

reportError

The **reportError** method enables an app to report an error event to metrics server.

Syntax

```
 reportError(errorCode, errorType, errorMessage, errorDetails)
```

- errorCode - string. The error code of the reported error. This param can be empty if not applicable.
- errorType - string. The error type of the the reported error. This param can be empty if not applicable.

- errorMessage - string. The error message of the reported error. This param can be empty if not applicable.
- errorDetails - string. The error details of the reported error. This param is a json string that can have key-value pairs for the following keys: errfile, errmethod, errline, errstacktrace, formID, widgetID, and flowTag.

```
 //Sample code for reportError
metricsServiceObj.reportError("1234", "SpecificError", "custom error
message", "{errfile:file.js}");
```

> ***Note:*** This API is required to be used only if the application developer chooses to send their own error events. If Error event type is chosen for supported platforms via application properties or setEventTracking API, error tracking will automatically be done.

reportHandledException

The **reportHandledException** method enables apps to report a handled exception event. Application developers can use this API to report handled exceptions in the application code.

Syntax

```
 reportHandledException(exceptionCode, exceptionType,
exceptionMessage, exceptionDetails)
```

- exceptionCode - string. The code for the reported exception. This param an be empty if not applicable.
- exceptionType - string, The type of the reported exception, such as Eval Error or syntax error. This param can be empty if not applicable.
- exceptionMessage - string. The message of the reported exception. This param can be empty if not applicable.

- exceptionDetails - string. The details of the reported exception. This param is a JSON string that can have key-value pairs for the following keys: exceptioncode, exceptionfile, exceptionmethod, exceptionline, exceptionstacktrace, formID, widgetID, and flowTag.

```
 //Sample code to send an exception to metrics server
metricsServiceObj.reportHandledException ("1234", "SpecificException",
"custom exception message", "{errfile:file.js}");
```

sendCustomMetrics

The **sendCustomMetrics** method allows the developer to send custom metrics from the application. The custom metrics keys should already be registered in Kony Fabric Console for the application before data is sent from the application.

Syntax

```
 sendCustomMetrics(reportingGroupID, metrics)
```

- reportingGroupID - string. The reporting group ID.
- metrics - object. Specifies the metrics being reported.

```
 //Sample code to sendCustomMetrics
metricsServiceObj.sendCustomMetrics("formID",
{"metrics":"metricdata"});
```

For more details about custom metrics and reports, refer to Custom Metrics and Reports Guide.

sendEvent

The **sendEvent** method allows a developer to send event details from an application to the server for analytics and reporting purposes. The event data is added to a buffer and sent to the server as per configuration values set by the developer using setEventConfig API.

Syntax

```
 sendEvent(evttype, evtSubType, formID, widgetID, flowTag, metaData)
```

- evttype - string. Specifies the event type for the reported event. Can be one of the following constants: FormEntry, FormExit, Touch, ServiceRequest, ServiceResponse, Gesture, Orientation, Error, Exception, Crash, Custom
- evtSubType - string. String literal for eventSubType (max 256 chars).
- formID - string. String literal for formID (max 256 chars) that specifies the widget ID of the form where event happened.
- widgetID - string. String literal that identifies the widget on which the event happened (max 256 chars).
- flowTag - string. String literal to override flow tag (max 256 chars).
- metaData - string. Specifies event-specific metadata.

```
 //Sample code to send reports
metricsServiceObj.sendEvent("FormEntry", "frmHome", "frmHome", null,
{"key1":"value1"});
```

> ***Note:*** This API is required to be used only if the application developer chooses to send their own custom events. All event types chosen for automatic event tracking from the **Metrics APM** tab in application properties or set using the **setEventTracking** API will automatically be tracked.

setEventConfig

The **setEventConfig** method sets the event configuration values.

Syntax

```
 setEventConfig(confType, eventBufferAutoFlushCount,
eventBufferMaxCount)
```

- confType - string. Specifies the current configuration type. This value can only be "Buffer" .
- bufferAutoFlushCount - number. This value can be any positive integer. The default value is 15. This param specifies the number of events to be buffered before flushing.

- maxBufferCount - number. This value can be any positive integer. The default value is 1000. This param specifies the maximum number of events that can be buffered. Events exceeding the maxBufferCount will be ignored.

```
 //Sample code to set the configuration for application events.
metricsServiceObj.setEventConfig("Buffer", 50, 200);
```

setFlowTag

The **setFlowTag** method sets an event flow tag to be associated with all new events that are reported by using the **sendEvent** API. The flow tag is used to ease searching event data in terms of application flows like loginflow, searchflow. The setFlowTag method also helps in sorting and filtering data while building custom reports or running standard reports for the application events.

Syntax

```
 setFlowTag(flowTag)
```

- flow Tag - string, Specifies the flow tag for reporting events.

```
 //Sample code to setFlowTag
metricsServiceObj.setFlowTag("MyFlowTag");
```

setSessionId

The **setSessionId** method sets the Id of the session.

Syntax

```
 setSessionId(sessionId)
```

- sessionId - string, Specifies the Id of the session.

```
 //Sample code to setFlowTag
metricsServiceObj.setSessionId("MySessionId");
```

setUserId

The **setUserId** API sets the user ID for the data gathered from an application. The user ID allows the data to be tracked on a user basis for broad analysis, such as how many different users used the application. It also helps to track activities of a specific user, which can help in seeing what activities were done before a crash, or what events led to a transaction not passing through. The user ID allows the same user to be tracked across different devices as well.

Syntax:

```
setUserId(userId)
```

- userId - string. Specifies the user ID.

```
//Sample code to set up the user ID of application user
metricsServiceObj.setUserId("myUserID");
```

> ***Note:*** The userId related to metrics. The userId length cannot be more than 100 characters.

> ***Note:*** **KNYMetricsService.setUserId** has been removed from apps built with Kony Visualizer Enterprise.

39.5.11.2 Event Details

For all event details, timestamp of event and session identifier values are automatically filled by MBaaS Client SDK, as part of the reportEvent, reportError and reportHandledException API calls. In case of automatically captured events, flowTag is also automatically filled with the currently set flowTag. The following are event specific details to be used while interfacing with MBaaS SDK while manually invoking sendEvent API to send event data.

FormEntry

- API to be used - sendEvent
- evtType - FormEntry

- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - null

FormExit

- API to be used - sendEvent
- evtType - FormExit
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - Dictionary (hash table) that contains the following key value pairs:
    - formdur - Duration spent in form in milliseconds. Optional.

Touch

- API to be used - sendEvent
- evtType - Touch
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - value of the ID property of the widget on which the touch happened
- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget)
- metadata - null

ServiceRequest

- API to be used - sendEvent
- evtType - ServiceRequest (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata - null

ServiceResponse

- API to be used - sendEvent
- evtType - ServiceResponse (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata
  JSON object (hash table) containing following key value pairs:
  - opstatus - Optional
    returned by Kony servers
  - httpcode - HTTP status code
  - resptime - time taken to get the reponse.

Gesture

- API to be used - sendEvent
- evtType - Gesture (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget where the gesture happened
- WidgetID - value of the ID property of the widget on which the gesture happened
- evtSubType [String]
  `GESTURETYPE_NUMBEROFINPUTS_DIRECTION`
  For example, two finger left swipe - SWIPE_2_LEFT
- metadata - null

Orientation

- API to be used - sendEvent
- evtType - Orientation (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String] - any one of the below constants is used
  - PORTRAIT_TO_LANDSCAPE
  - LANDSCAPE_TO_PORTRAIT
- metadata - null

Error

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType
  ErrorCode - Optional
- metadata
  JSON object (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errstacktrace - Optional
  - errcustommsg - Optional

HandledException

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ExceptionCode - Optional
- metadata
  JSON object (hash table) containing following key value pairs:
  - exceptioncode - Optional
  - exceptionev - Optional
  - exceptionmsg - Optional
  - exceptionfile - Optional

- exceptionmethod - Optional
- exceptionstacktrace - Optional
- exceptioncustommsg - Optional

Crash

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String]
- metadata
  JSON object (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errline - Optional
  - errstacktrace - Optional
  - errcrashreport - Optional

Custom

- API to be used - sendEvent
- evtType - Custom
- FormID - any supplied form ID

- WidgetID - any supplied widget ID
- evtSubType - any supplied event subtype
- metadata - string or a dictionary

39.5.12 Invoking an Object Service

Kony supplies you with programmatic access to backend data, both online and offline. Access is gained in two steps:

1. Acquire a current instance of your object service.
2. Use the object service instance together with data transfer objects to communicate as needed with your backend, either online or offline with a local cache.

To know more about how to acquire a current instance of your object service, refer to getObjectService Method documentation.

To know more about the methods that act on the MF endpoint directly, refer to OnlineObjectServiceClass documentation.

To know more about the methods that act on the local sync database, refer to OfflineObjectServiceClass documentation.

To know more about the usage of the data transfer objects, refer to Data Transfer Objects documentation.

39.5.12.1 getObjectService Method

The **getObjectService Method** gets the current instance of the object service. The getObjectService method is invoked on the sdk instance; init must already have been successfully run before invoking this method.

Syntax

```
getObjectService(serviceName,{serviceType});
```

Parameters

Parameter	Type	Description
serviceName	string	Name of the object service
serviceType	JSON object	One of the following: • "access" : "online" *[default]* • "access" : "offline" • "access" : "registeredObjectServiceName" *[This is the name of the object service defined in the application]*

Response

Returns the object service instance based on the value specified in serviceType. The default value is "online", which returns an "OnlineObjectService Class" below instance. A value of "offline" returns an "OfflineObjectService Class" on page 1508 instance.

39.5.12.2 OnlineObjectService Class

Provides methods that perform operations acting on the Mobile Fabric endpoint, including basic CRUD, metadata, and binary-related functions. An instance of OnlineObjectService is returned by the getObjectService Method when the second parameter specifies {"access":"online"}.

Methods

The following methods are used by the OnlineObjectService class and its instantiations.

create Method

Creates an object in the Kony Fabric endpoint.

39.5.12.3 Syntax

```
create(options, successCallback, failureCallback);
```

39.5.12.4 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the primary key of the created object
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.5 Example

```
 function create() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
    });
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    var options = {
        "dataObject": dataObject
    };
    objSvc.create(options,     function(res) {
        alert("Record created");
    },     function(err) {
        alert("Error in record creation");
    }  );
}
```

> ***Note:*** When using object services for SAP, the general norm is to have character field values stored in upper case. However, if you need to pass in mixed/lower case values for an SAP field, ensure that this field is designated as mixed case in the SAP Add-in LDB workbench.

update Method

Updates an object in the Kony Fabric endpoint.

39.5.12.6 Syntax

```
update(options, successCallback, failureCallback);
```

39.5.12.7 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.8 Example

```
function update() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    dataObject.addField("primaryKeyField", "value");
    var options = {
```

```
        "dataObject": dataObject
    };
    objSvc.update(options,     function(res) {
        alert("Record updated");
    },     function(err) {
        alert("Error in record update");
    }  );
}
```

delete Method

Deletes an object in the Kony Fabric endpoint.

39.5.12.9 Syntax

```
deleteRecord(options, successCallback, failureCallback);
```

39.5.12.10 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the number of records deleted
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.11 Example

```
function deleteRecord() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
```

```
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    dataObject.addField("primaryKeyField", "value");
    var options = {
        "dataObject": dataObject
    };
    objSvc.deleteRecord(options,     function(res) {
        alert("Record deleted");
    },     function(err) {
        alert("Error in record deletion");
    }  );
}
```

customVerb Method

Performs a custom operation on an object in the Kony Fabric endpoint.

39.5.12.12 Syntax

```
customVerb(verbName, options, successCallback, failureCallback);
```

39.5.12.13 Parameters

Parameter	Description
verbName	Name of the custom verb defined in the Kony Fabric console
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.14 Example

```
 function customData() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    dataObject.addField("primaryKeyField", "value");
    var options = {
        "dataObject": dataObject
    };
    objSvc.customVerb("verbName", options,     function(res) {
        alert("Custom operation performed");
    },     function(err) {
        alert("Error in custom operation");
    } );
}
```

fetch Method

Fetches an object from the server.

39.5.12.15 Syntax

```
 fetch(options, successCallback, failureCallback);
```

39.5.12.16 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518. This instance must have the property *selectQueryObject*, which is an instance of kony.sdk.dto.SelectQuery, in order to fetch records based on the given criteria.
successCallback	Function invoked when the method succeeds, with the number of records fetched
failureCallback	Function invoked when fetch fails, with cause of failure

39.5.12.17 Example

```
 function fetchData() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    var dataObject = new kony.sdk.dto.DataObject("EAM_WO_ATTACHMENT");
    var selQuery = new kony.sdk.dto.SelectQuery("serviceName",
tblDto);
    var odataUrl = "$filter=ORDER_NUM eq 10001";
    dataObject.setSelectQueryObject(selQuery);
    var options = {
        "dataObject": dataObject
    };
    objSvc.fetch(options,     function(res) {
        alert("record::" + res.records);
    },     function(err) {
        alert("Failed to fetch : " + JSON.stringify(err));
```

```
    }   );
}
```

getMetadataOfAllObjects Method

Gets the metadata associated with the objects defined in the service from the server.

39.5.12.18 Syntax

```
getMetadataOfAllObjects(options, successCallback, failureCallback);
```

39.5.12.19 Parameters

Parameter	Description
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.20 Example

```
function getMetadata() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    objSvc.getMetadataOfAllObjects({},     function(res) {
        alert("Metadata::" + res);
    },     function(err) {
        alert("Error in metadata::" + err);
    }  );
}
```

getMetadataOfObject Method

Gets the metadata associated with an object defined in the service from the server.

39.5.12.21 Syntax

```
 getMetadataOfObject(objectName, options, successCallback,
failureCallback);
```

39.5.12.22 Parameters

Parameter	Description
objectName	The name of the desired object as defined in the service
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds, with the number of records gotten
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.23 Example

```
 function getMetadata() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    objSvc.getMetadataOfObject("objectName", {},     function(res) {
        alert("Metadata::" + res);
    },     function(err) {
        alert("Error in metadata::" + err);
    }  );
}
```

getBinaryContent Method

Gets binary content on the server.

39.5.12.24 Syntax

```
getBinaryContent(options, successCallback, failureCallback);
```

39.5.12.25 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518, and "binaryAttrName", which is the binary field name in the object
successCallback	Function invoked when the operation succeeds, with the number of records gotten
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.26 Example

```
 function getBinaryContent() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    var dataObject = new kony.sdk.dto.DataObject("media");
    //primary key details to get media object
    dataObject.addField("name", "4005174-002");
    objSvc.getBinaryContent(    {
        "dataObject": dataObject,
        "binaryAttrName": "data"
    },    function(bin) {
        alert("binary content is : " + JSON.stringify(bin));
        frmBinary.downloadImg.isVisible = true;
        frmBinary.downloadImg.base64 = bin;
```

```
    },      function(err) {
        alert("failed to get binary data : " + JSON.stringify(err));
    }  );
}
```

createBinaryContent Method

Creates binary content on the server.

39.5.12.27 Syntax

```
createBinaryContent(options, successCallback, failureCallback);
```

39.5.12.28 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518, and "binaryAttrName", which is the binary field name in the object
successCallback	Function invoked when the operation succeeds, with the number of records created
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.29 Example

```
function createBinaryContent() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "online"
    });
    var dataObject = new kony.sdk.dto.DataObject("media");
    dataObject.addField("name", imgName);
    dataObject.addField("data", binaryText);
```

```
    objSvc.createBinaryContent(    {
            "dataObject": dataObject,
            "binaryAttrName": "data"
        },    function(bin) {
            alert("Binary content created");
        },
        function(err) {
            alert("Failed: " + JSON.stringify(err));
        }  );
}
```

updateBinaryContent Method

Updates binary content on the server.

39.5.12.30 Syntax

```
updateBinaryContent(options, successCallback, failureCallback);
```

39.5.12.31 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518, and "binaryAttrName", which is the binary field name in the object
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.32 Example

```
function updateBinaryContent() {
   var objSvc = kony.sdk.getCurrentInstance().getObjectService
```

```
("serviceName", {
        "access": "online"
    });
    var dataObject = new kony.sdk.dto.DataObject("media");
    dataObject.addField("name", imgName);
    dataObject.addField("data", binaryText);
    objSvc.updateBinaryContent(    {
            "dataObject": dataObject,
            "binaryAttrName": "data"
        },    function(bin) {
            alert("Binary content updated");
        },
        function(err) {
            alert("Failed: " + JSON.stringify(err));
        }  );
}
```

39.5.12.33 OfflineObjectService Class

Provides methods that perform operations acting on the sync database, including basic CRUD, metadata, and binary-related functions. An instance of OfflineObjectService is returned by the getObjectService Method when the second parameter specifies {"access":"offline"}.

Methods

The following methods are used by the OfflineObjectService class and its instantiations.

create Method

Creates an object offline in the sync database.

39.5.12.34 Syntax

```
create(options, successCallback, failureCallback);
```

39.5.12.35 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the primary key of the created object
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.36 Example

```
 function create() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
    });
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    var options = {
        "dataObject": dataObject
    };
    objSvc.create(options,     function(res) {
        alert("Record created");
    },     function(err) {
        alert("Error in record creation");
    }  );
}
```

> ***Note:*** When using object services for SAP, the general norm is to have character field values stored in upper case. However, if you need to pass in mixed/lower case values for an SAP field, ensure that this field is designated as mixed case in the SAP Add-in LDB workbench.

update Method

Updates an object offline in the sync database (local store).

39.5.12.37 Syntax

```
update(options, successCallback, failureCallback);
```

39.5.12.38 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.39 Example

```
function update() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
    });
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    dataObject.addField("primaryKeyField", "value");
    var options = {
```

```
        "dataObject": dataObject
    };
    objSvc.update(options,     function(res) {
        alert("Record updated");
    },     function(err) {
        alert("Error in record update");
    }  );
}
```

delete Method

Deletes an object offline in the sync database.

39.5.12.40 Syntax

```
deleteRecord(options, successCallback, failureCallback);
```

39.5.12.41 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which must be an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the number of records deleted
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.42 Example

```
function deleteRecord() {
   var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
    });
```

```
    var dataObject = new kony.sdk.dto.DataObject("ObjectName");
    dataObject.addField("field1", "value1");
    dataObject.addField("primaryKeyField", "value");
    var options = {
        "dataObject": dataObject
    };
    objSvc.deleteRecord(options,     function(res) {
        alert("Record deleted");
    },     function(err) {
        alert("Error in record deletion");
    }  );
}
```

getMetadataOfAllObjects Method

Gets the metadata associated with the objects defined in the service from the local store.

39.5.12.43 Syntax

```
getMetadataOfAllObjects(options, successCallback, failureCallback);
```

39.5.12.44 Parameters

Parameter	Description
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds, with the number of records updated
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.45 Example

```
function getMetadata() {
   var objSvc = kony.sdk.getCurrentInstance().getObjectService
```

```
("serviceName", {
        "access": "offline"
    });
    objSvc.getMetadataOfAllObjects({},     function(res) {
        alert("Metadata::" + res);
    },     function(err) {
        alert("Error in metadata::" + err);
    }  );
}
```

getMetadataOfObject Method

Gets the metadata associated with an object defined in the service from the local store.

39.5.12.46 Syntax

```
 getMetadataOfObject(objectName, options, successCallback,
failureCallback);
```

39.5.12.47 Parameters

Parameter	Description
objectName	The name of the desired object as defined in the service
options	JSON object with the optional parameter "getFromServer"
successCallback	Function invoked when the operation succeeds, with the number of records returned
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.48 Example

```
 function getMetadata() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
```

```
        "access": "offline"
    });
    objSvc.getMetadataOfObject("objectName", {},     function(res) {
        alert("Metadata::" + res);
    },     function(err) {
        alert("error in metadata::" + err);
    }  );
}
```

executeSelectQuery Method

Executes a Select query on the sync database.

39.5.12.49 Syntax

```
executeSelectQuery(queryString, successCallback, failureCallback);
```

39.5.12.50 Parameters

Parameter	Description
queryString	SQL Select query string
successCallback	Function invoked when the operation succeeds, with the number of records operated on
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.51 Example

```
function executeSelectQuery() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
    });
    var queryString = "select * from table";
```

```
    objSvc.executeSelectQuery(queryString,     function(res) {
        alert("Metadata::" + res);
    },     function(err) {
        alert("Error in metadata::" + err);
    }  );
}
```

getBinaryContent Method

Gets binary content from the sync database.

39.5.12.52 Syntax

```
getBinaryContent(options, successCallback, failureCallback);
```

39.5.12.53 Parameters

Parameter	Description
options	JSON object with the mandatory parameter "dataObject", which is an instance of the "kony.sdk.dto.DataObject(string objectName)" on page 1518
successCallback	Function invoked when the operation succeeds, with the number of records gotten
failureCallback	Function invoked when the operation fails, with cause of failure

39.5.12.54 Example

```
function getBinaryContent() {
    var objSvc = kony.sdk.getCurrentInstance().getObjectService
("serviceName", {
        "access": "offline"
    });
    var dataObject = new kony.sdk.dto.DataObject("media");
    //primary key details to get media object
```

```
    dataObject.addField("name", "4005174-002");
    var options = {};
    options.dataObject = dataObject;
    //binary column name to get the binary data
    options.binaryAttrName = "data";
    options.responsetype = "base64string/filepath";
    //filepath is default if it is not set by developer
    objSvc.getBinaryContent(options, successcallback, errorcallback);

    function successcallback(response) {
        //response will contain base64string or filepath
        //  based on responsetype
        //value provided by the developer.
        var content = response.records[0].data;
    }

    function errorcallback(err) {
        Alert(err);
    }
}
```

39.5.12.55 Data Transfer Objects

A data transfer object (DTO) represents a database element in a programmatic context. Each DTO is used in various Object Services API methods. The DTOs comprise several classes:

- kony.sdk.dto.Column Class
- kony.sdk.dto.DataObject Class

kony.sdk.dto.Column Class

This class represents a column in a database table, which is in turn represented by a kony.sdk.dto.Table object. It is used when creating a query.

Constructors

The kony.sdk.dto.Column class has one constructor.

39.5.13 kony.sdk.dto.Column(table, columnName) Constructor

39.5.13.1 Signature

```
kony.sdk.dto.Column(table, columnName)
```

39.5.13.2 Parameters

Parameter name	Type	Description
table	kony.sdk.dto.Table	The table containing the column
columnName	string	The name of the column

Example

```
 var dataObject = new kony.sdk.dto.DataObject("EAM_WO_ATTACHMENT");
var tblDto = new kony.sdk.dto.Table("EAM_WO_ATTACHMENT", "EAM_WO_
ATTACHMENT", false);
var selQuery = new kony.sdk.dto.SelectQuery("serviceName", tblDto);
var colObj = new kony.sdk.dto.Column(tblDto, "ORDER_NUM");
var colObj2 = new kony.sdk.dto.Column(tblDto, "BINARY_NAME");
var colObj3 = new kony.sdk.dto.Column(tblDto, "ATTACH_DESC");
selQuery.addColumn(colObj);
selQuery.addColumn(colObj2);
selQuery.addColumn(colObj3);
dataObject.setSelectQueryObject(selQuery);
```

kony.sdk.dto.DataObject Class

This class represents a data object in the Object Service. An instantiation of this class is required as a parameter in many methods of the OnlineObjectService Class and the OfflineObjectService Class.

Constructors

The kony.sdk.dto.DataObject class has one constructor.

39.5.14 kony.sdk.dto.DataObject(string objectName)

Parameter	Type	Description
objectName	string	Name of object defined in the object service

Fields

Field name	Type	Description
setOdataUrl	string	Specifies the Odata URL

Methods

The kony.sdk.dto.DataObject class includes the following methods.

- addChildDataObject(child)
- addField(fieldName, value)
- setRecord(record)
- setSelectQueryObject(queryObject)

39.5.15 addChildDataObject(child) Method

Adds another DataObject as a child..

39.5.15.1 Signature

addChildDataObject(child)

39.5.15.2 Parameters

Parameter	Type	Description

child	kony.sdk.dto.DataObject	The DataObject to be added as a child

39.5.16 addField(fieldName, value) Method

Adds a field in the data object.

39.5.16.1 Signature

addField(fieldName, value)

39.5.16.2 Parameters

Parameter	Type	Description
fieldName	string	The name of the field being added.
value	string	The value of the added field

39.5.17 setRecord(record) Method

Specifies a record in the data object.

39.5.17.1 Signature

setRecord(record)

39.5.17.2 Parameters

Parameter	Type	Description
record	JSON object	The record to be set.

39.5.18 setSelectQueryObject(queryObject) Method

Sets the specific query object to be used in the query.

39.5.18.1 Signature

setSelectQueryObject(queryObject)

39.5.18.2 Parameters

Parameter	Type	Description
queryObject	kony.sdk.dto.SelectQuery	The object that contains the query.

Example

```
 // **Simple Object**
var dataObject = new kony.sdk.dto.DataObject("objectName");
var record = {};
record.field1 = "value1";
record.field2 = "value2";
record.field3 = "value3";
//sets the record to the dataObject
dataObject.setRecord(record);
// **Complex Object**
var parentDataObject = new kony.sdk.dto.DataObject("parentObject");
var record = {};
record.field1 = "value1";
record.field2 = "value2";
record.field3 = "value3";
//sets the record to the parent.
parentDataObject.setRecord(record);
var childDataObject = new kony.sdk.dto.DataObject("childObject");
var record = {};
record.field4 = "value3";
record.field5 = "value5";
record.field6 = "value6";
//sets the record to the child
childDataObject.setRecord(record);
```

```
parentDataObject.addChildDataObject(childDataObject); // **Adding
OdataUrl**
var dataObject = new kony.sdk.dto.DataObject("objectName");
dataObject.setOdataUrl("$filter=abc eq 123");
```

39.6 iOS SDK

These steps show how to add the Kony-iOS-SDK to your project and set up Kony Fabric Client.

- Prerequisites
- Downloading Kony iOS SDK
- Configuring the Framework to your Project
- Initializing the iOS Client SDK
- Invoking an Identity Service
- Invoking an Integration Service
- Invoking a Messaging Service
- Invoking a Sync Service
- Invoking a Metrics Service
- API Reference

  To view API Reference for Kony iOS, click com.kony.KonySDK.docset.

39.6.1 Prerequisites

- Mavericks OS X or Mountain Lion OS X
- Xcode 5.1
- If you are using an untrusted self-signed (SSL) certificate with Kony Fabric installation, enable

the [KNYClient acceptSelfSignedCertificates]; API, by default native apps do not allow untrusted SSL certificates for HTTPS connection. For more details, refer to SSL API.

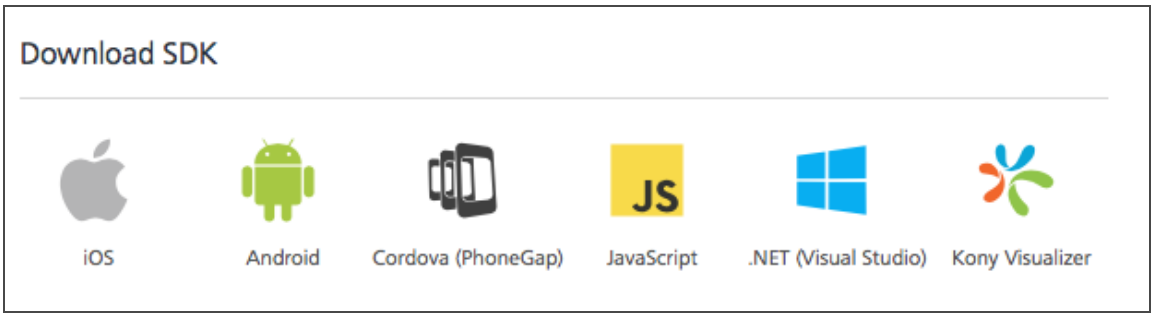
- If you are developing an iOS app extension using the `KonySDK.framework`, add the following frameworks in your app extension project build phases:
    - SDKCommons.framework
    - Binary.framework
    - CMS.framework
    - konyLogger.framework

  After adding these frameworks, add the `KNYSharedContainerIdentifier` key to the extension app `info.plist`.

39.6.2 Downloading Kony iOS SDK Files

To download Kony iOS SDK, follow these steps:

1. In Kony Fabric console, navigate to **Apps** > **SDKs**, and click **iOS**. The system prompts you to save the zip file in your local system.



2. Save the `kony-ios-sdk.zip` file in your local system.

3. Extract the `kony-ios-sdk.zip` file that you just downloaded.

   The `KonySDK` folder contains the following files:

   - KonySDK.framework
   - com.kony.KonySDK.docset
   - LICENSE.txt
   - version.txt

39.6.3 Configuring the Framework

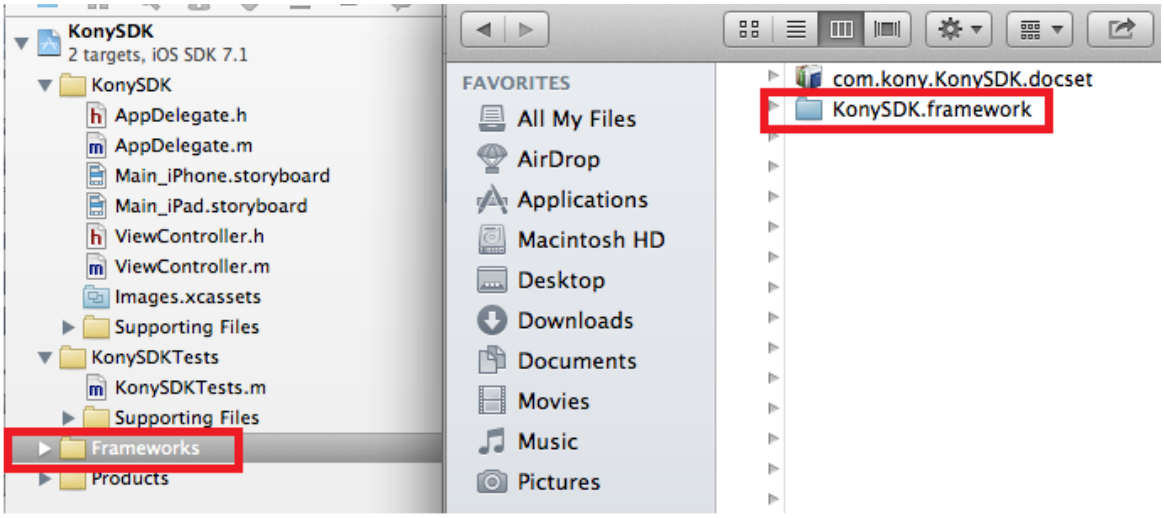
Before using Kony Fabric APIs for iOS, you must configure Kony iOS SDK into your IDEs, such as Xcode. Configuring the Kony iOS SDK involves the following steps:

1. Configuring KonySDK.framework to Project
2. Adding Framework Dependencies
3. Installing com.kony.KonySDK.docset in Xcode

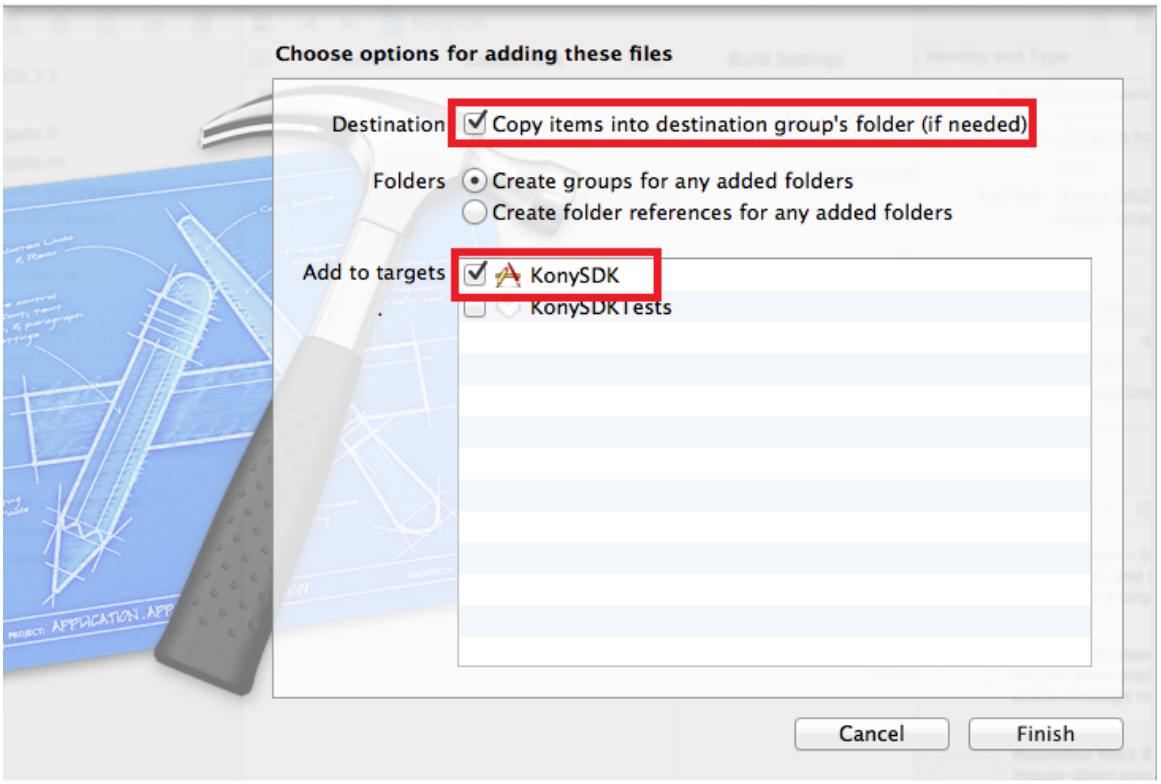
39.6.3.1 Configuring KonySDK.framework to Project

To configure KonySDK Framework to project, follow these steps:

1. Extract the downloaded `kony-ios-sdk.zip` file.
2. Drag `KonySDK.framework` to your **Framework** group in Xcode project.



3. Select the **Copy items into destination group's folder** check box. Make sure it adds these libraries to your main target.



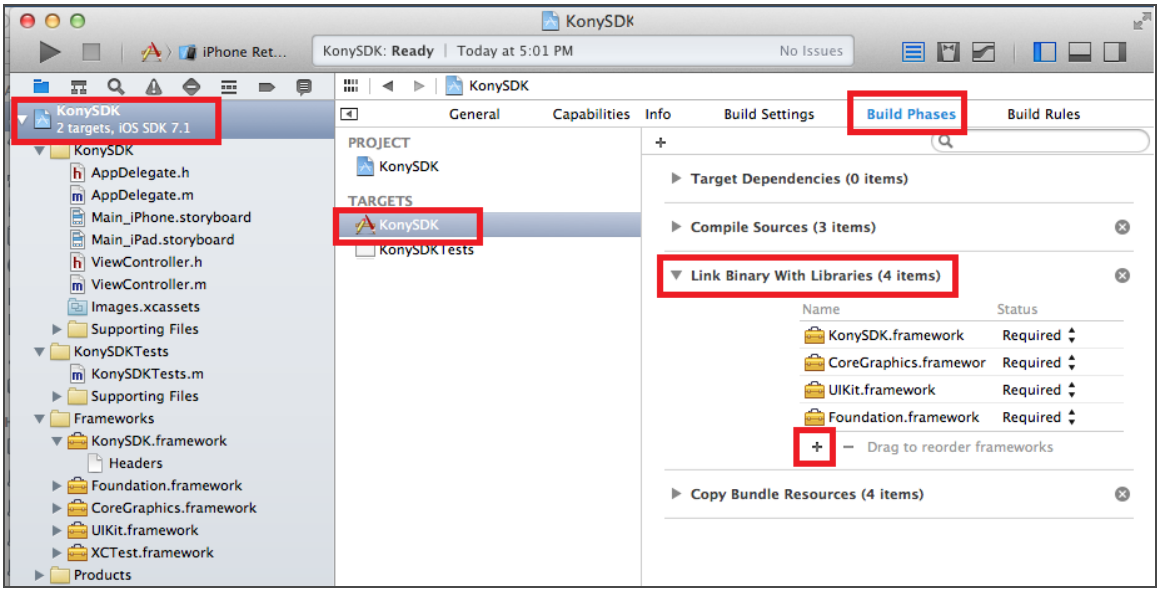
4. Click **Finish**.

39.6.3.2 Adding Framework Dependencies

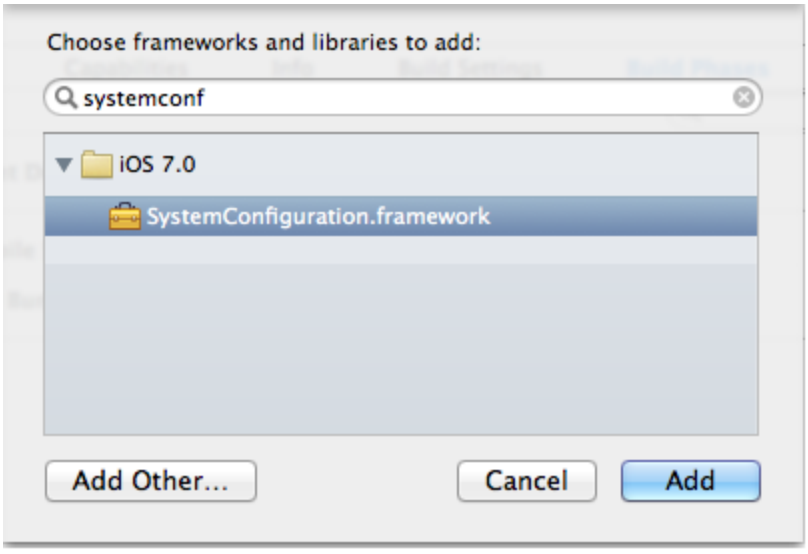
You need to link your binaries with the following libraries.

To configure binaries, follow these steps:

1. In the **Project Navigator**, select your project.



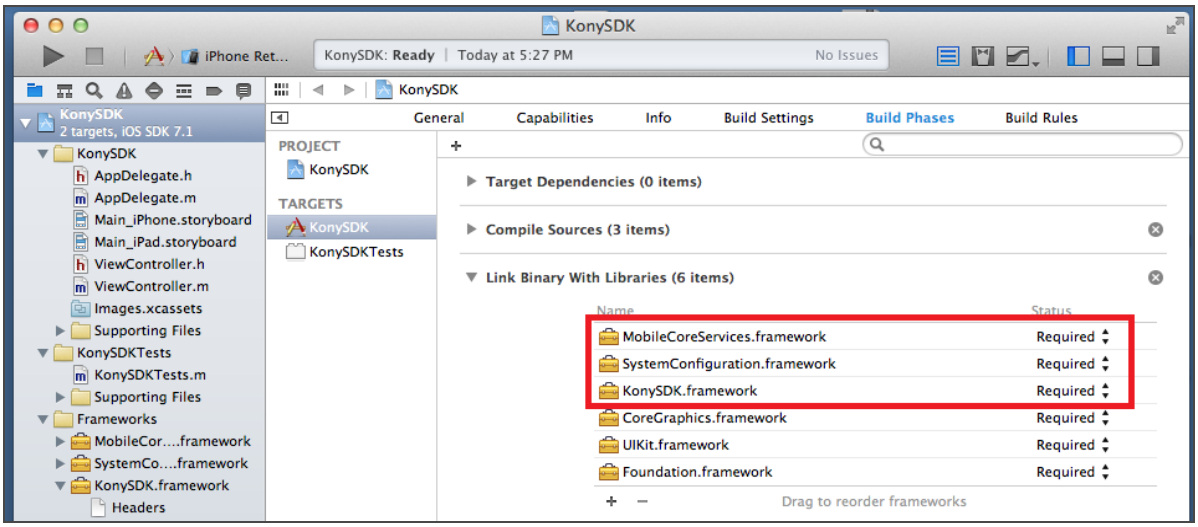
2. In the **Editor**, select your app target, navigate to **Build Phases** > **Link Binary With Libraries** and then click the **Add (+)** button.



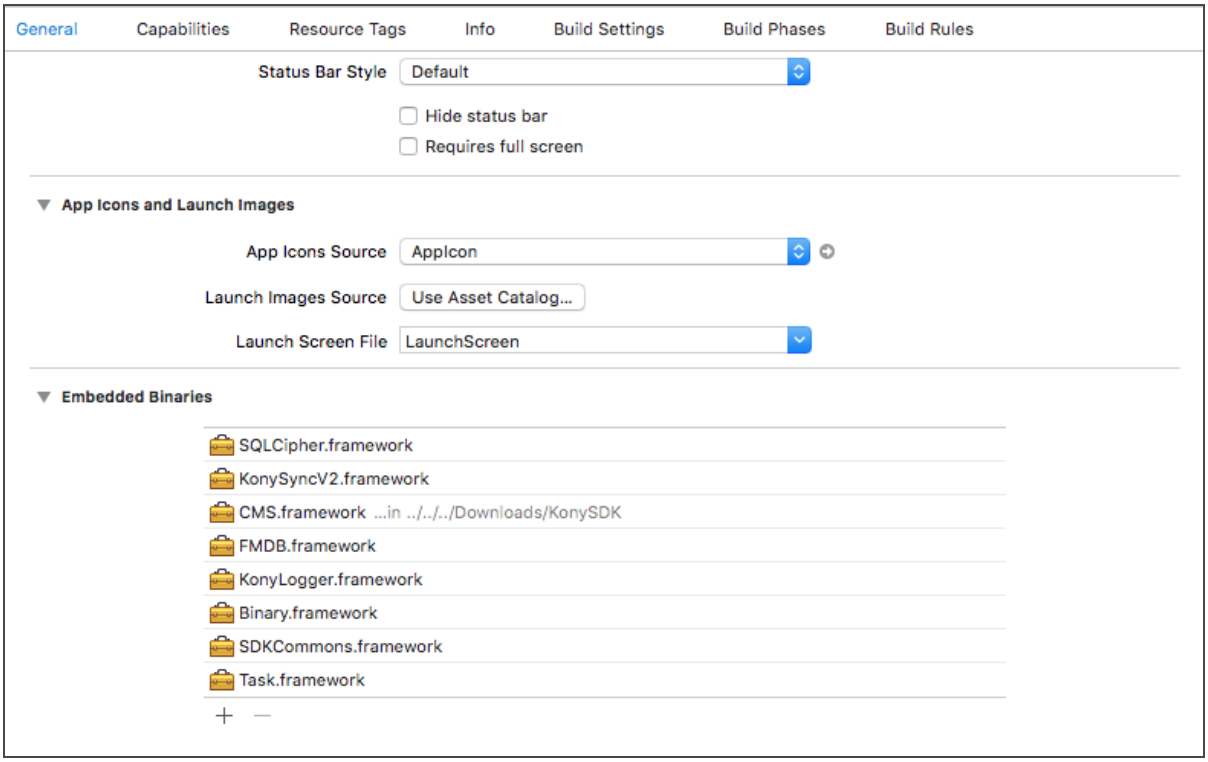
3. In the **Choose frameworks and libraries to add** dialog, type *systemconf*.

4. Select the `SystemConfiguration.framework`, and then click **Add**. The system adds the selected frameworks to the project under the **Link Binary With Libraries** section.
5. Add the `MobileCoreServices.framework` by following Step 3 through Step 4.

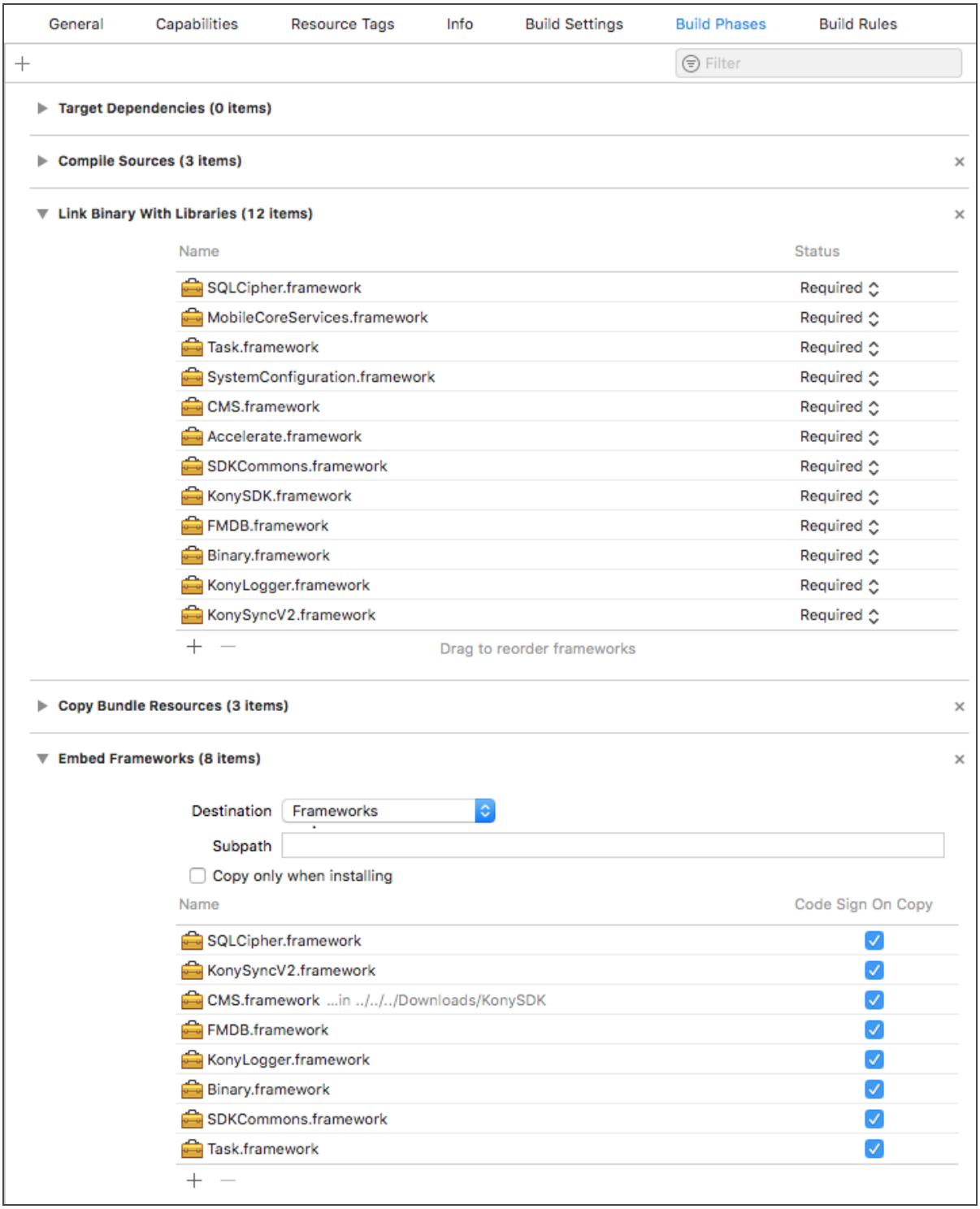
   The system adds the selected frameworks to the project under the **Link Binary With Libraries** section.



6. Now, add the remaining frameworks (shown in the following image) under the **Embedded Binaries** section in the **General** tab.



By doing so, these files will be listed in **Link Binary With Libraries** and **Embedded Frameworks** sections in the **Build Phases** tab.

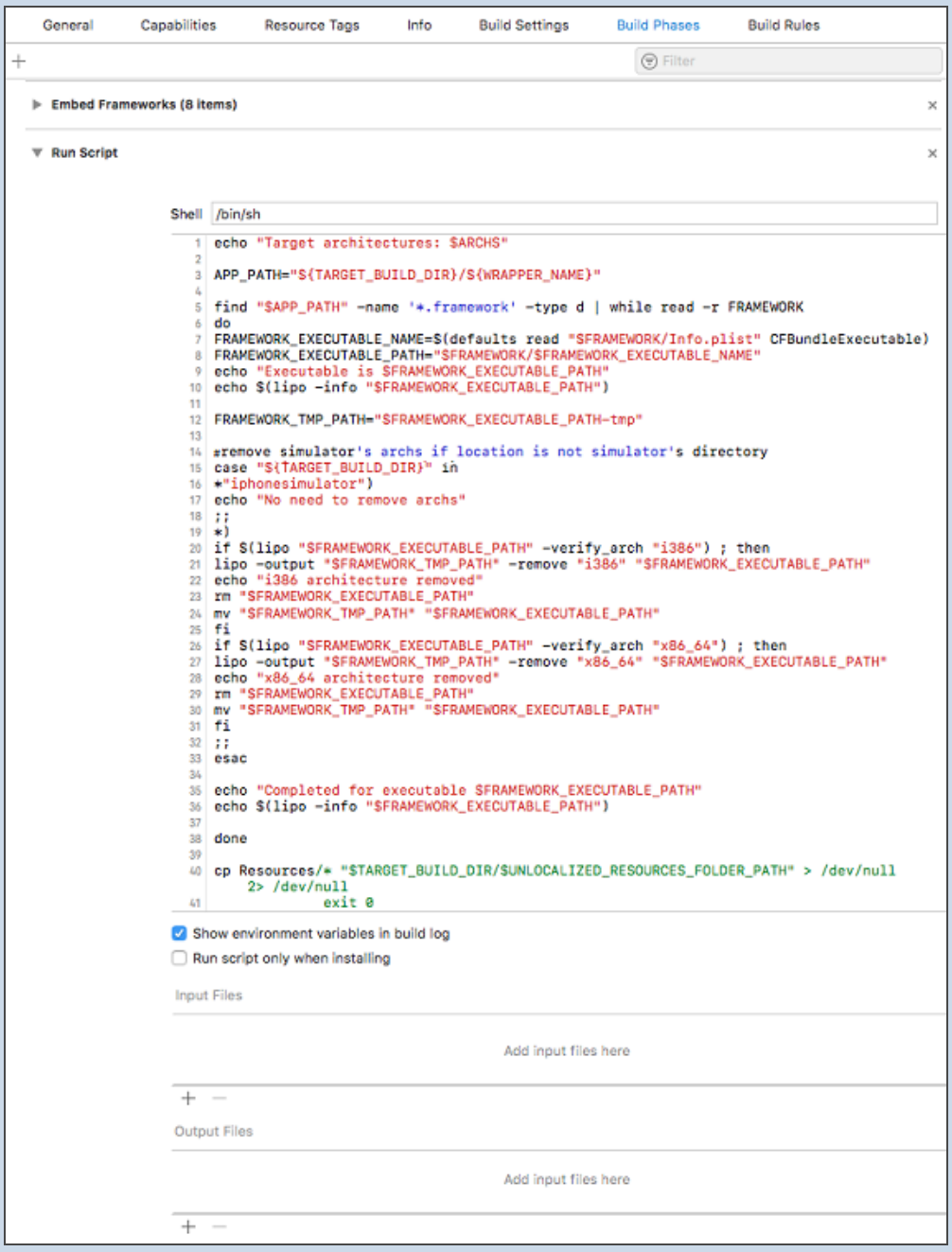


7. For example, if you want to release an app to the AppStore and not include i386 and x86_64 architectures, you must add a script before building your project. To do so, follow these steps.
    i. Under the **Build Phases** tab, click the **Add (+)** button and select **New Run Script Phase** from the list. The Run Script section appears.
    ii. Type the following script in the text box.

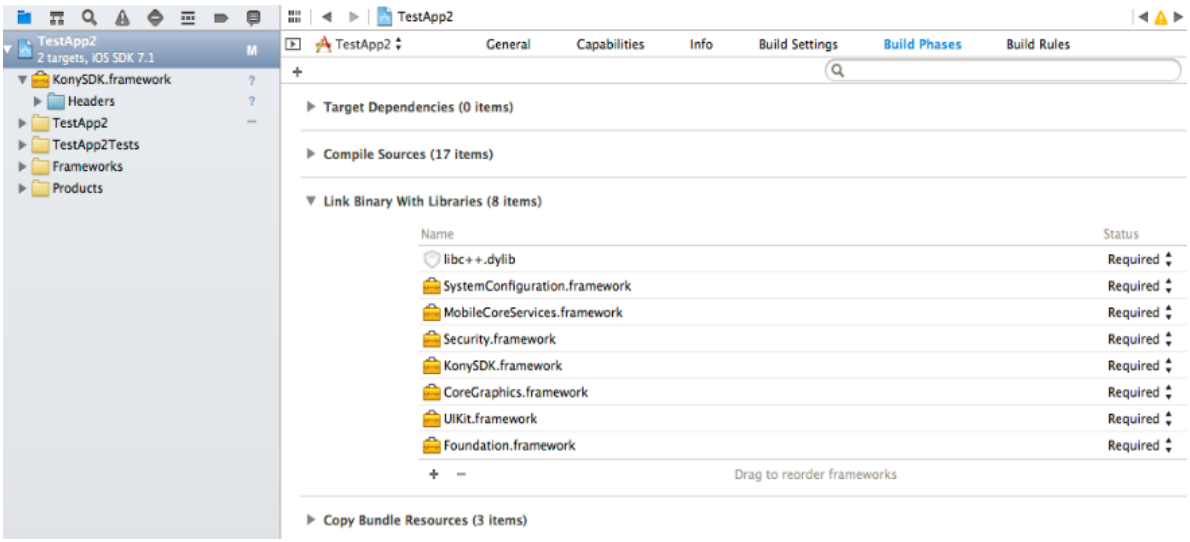
```
echo "Target architectures: $ARCHS"

APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"

find "$APP_PATH" -name '*.framework' -type d | while read -r
FRAMEWORK
do
FRAMEWORK_EXECUTABLE_NAME=$(defaults read
"$FRAMEWORK/Info.plist" CFBundleExecutable)
FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_
NAME"
echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"
echo $(lipo -info "$FRAMEWORK_EXECUTABLE_PATH")

FRAMEWORK_TMP_PATH="$FRAMEWORK_EXECUTABLE_PATH-tmp"
#remove simulator's archs if location is not simulator's
#directory
case "${TARGET_BUILD_DIR}" in
*"iphonesimulator")
echo "No need to remove archs"
;;
*)
if $(lipo "$FRAMEWORK_EXECUTABLE_PATH" -verify_arch "i386") ;
then
lipo -output "$FRAMEWORK_TMP_PATH" -remove "i386"
"$FRAMEWORK_EXECUTABLE_PATH"
```

```
echo "i386 architecture removed"
rm "$FRAMEWORK_EXECUTABLE_PATH"
mv "$FRAMEWORK_TMP_PATH" "$FRAMEWORK_EXECUTABLE_PATH"
fi
if $(lipo "$FRAMEWORK_EXECUTABLE_PATH" -verify_arch "x86_64")
; then
lipo -output "$FRAMEWORK_TMP_PATH" -remove "x86_64"
"$FRAMEWORK_EXECUTABLE_PATH"
echo "x86_64 architecture removed"
rm "$FRAMEWORK_EXECUTABLE_PATH"
mv "$FRAMEWORK_TMP_PATH" "$FRAMEWORK_EXECUTABLE_PATH"
fi
;;
esac

echo "Completed for executable $FRAMEWORK_EXECUTABLE_PATH"
echo $(lipo -info "$FRAMEWORK_EXECUTABLE_PATH")

done

cp Resources/* "$TARGET_BUILD_DIR/$UNLOCALIZED_RESOURCES_
FOLDER_PATH" > /dev/null 2> /dev/null
exit 0
```

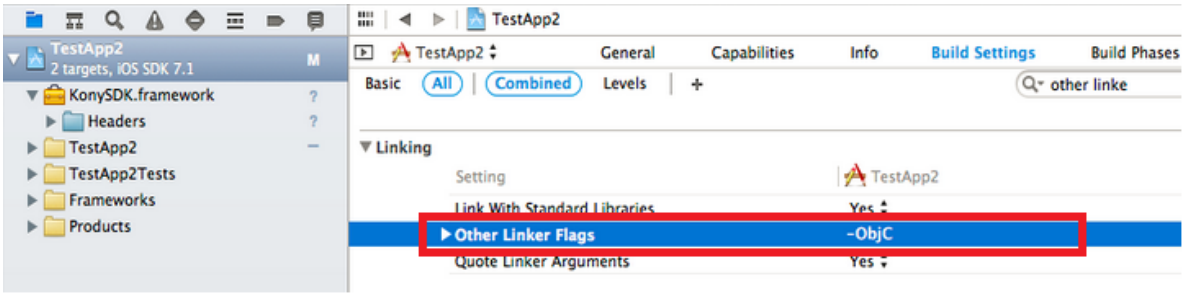
Note: This step should be the final step under the Build Phase.



8. Add dependent libraries to your project, shown below:



9. Add linker flags to your project , shown below:



39.6.3.3 Installing com.kony.KonySDK.docset in Xcode

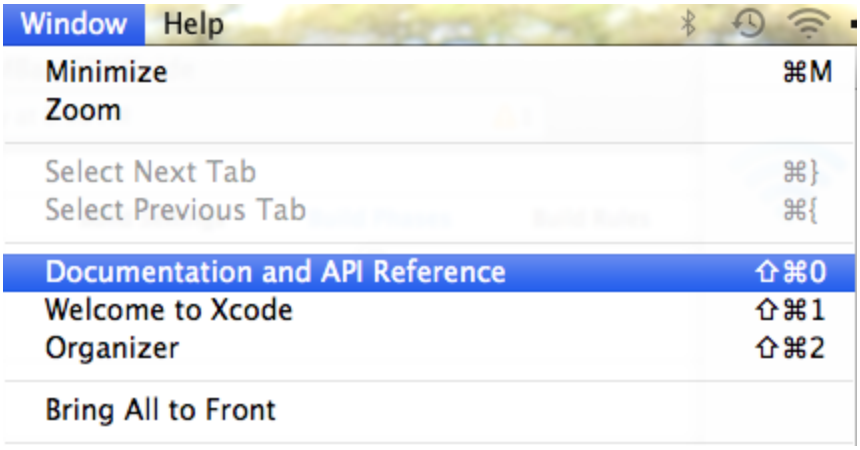
Kony provides Xcode docset to search and browse API documentation within Xcode. The docset also provides quick help in the code completion popup.

To view Kony iOS Docset help in native format, click com.kony.KonySDK.docset.

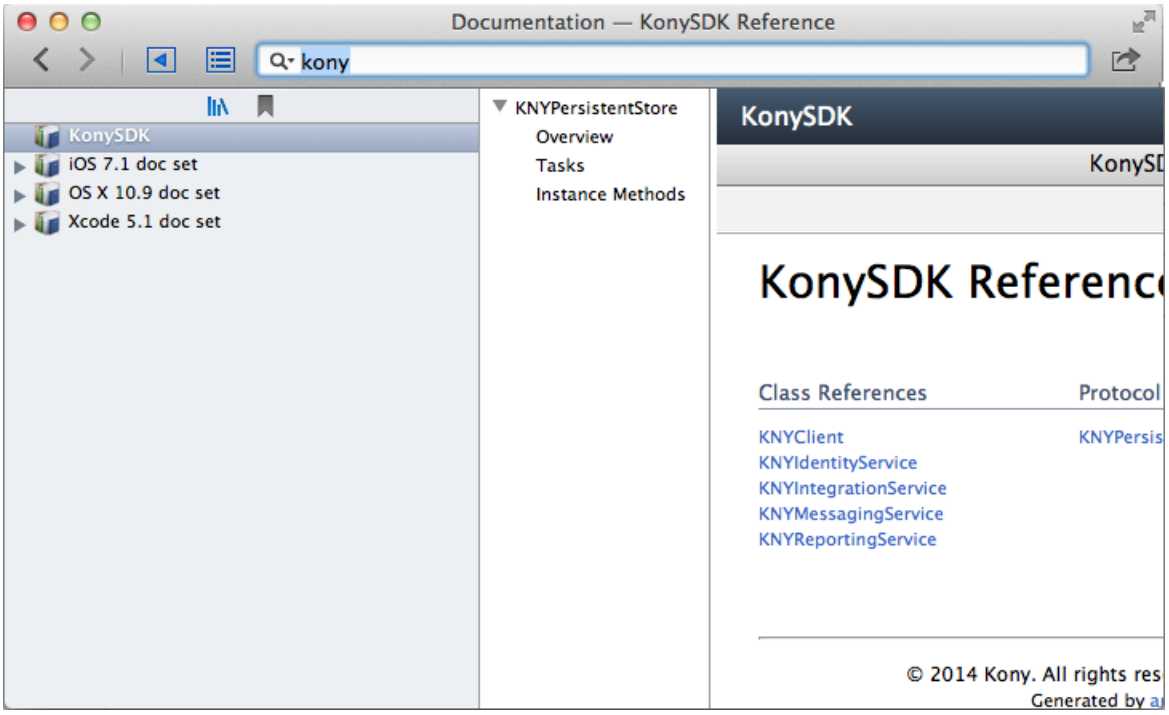
To install the com.kony.KonySDK.docset in Xcode, follow these steps:

1. Shutdown **Xcode**.
2. Copy `com.kony.KonySDK.docset` folder from `KonySDK` folder.

3. Navigate to `~/Library/Developer/Shared/Documentation/DocSets/` and then paste `com.kony.KonySDK.docset` folder.
4. Restart **Xcode**.
5. In **Xcode**, click **Window** menu, and then click **Documentation and API Reference**.



6. In the **Search** box, type *Kony*, and then press **Enter**. The system displays the configured Docset.



39.6.4 Initializing the iOS Client SDK

To use any Kony Fabric SDK functions, you must add `#import <KonySDK/KNYSDK.h>` in the header of your files. You must use the following code snippet to initialize the SDK Client before using any services. Do this in your *AppDelegate's application:didFinishLaunchingWithOptions:* method.

```
//Sample code to initialize Kony Fabric Client
KNYClient * client = [KNYClient sharedClient];
[client initializeInBackgroundWithAppKey: @"<app-key>"
    appSecret: @"<app-secret>"
    serviceURLString: @"<service-url>"
    completion: ^ (BOOL succeeded, NSError * error) {
        if (succeeded) {
            // handle the success case here
        } else {
            // handle the failure case here
            // use the details in 'error'
```

```
        }
    }
];
```

> ***Note:*** Only upon successful completion of -[KNYClient initializeInBackgroundWithAppKey:appSecret:serviceURLString:completion] method, you can call the other methods in the SDK such as Identity Service, Integration Service, Messaging Service, and Reporting Service.
>
> As -[KNYClient initializeInBackgroundWithAppKey:appSecret:serviceURLString:completion] method is asynchronous call, you must wait until the background process to be completed before you call the other methods in the SDK. If you attempt to call any other methods in the SDK before the initialization method completion, the system will not allow you to call the methods and throws exception.

If you are using an untrusted self-signed (SSL) certificate with Fabric installation, by default native apps do not allow untrusted SSL certificates for HTTPS connection.

To make your native apps work with untrusted SSL certificates, call the following API:

```
[KNYClient acceptSelfSignedCertificates];
```

> ***Note:*** Once a user calls the [KNYClient acceptSelfSignedCertificates]; API, a native application will accept SSL certificates throughout the app life cycle. A user cannot disable the API from a native app running on a device.

When SDK is initialized, the Kony SDK registers a session and sends its information to the Kony Fabric Server. If the device is offline, or the server is not reachable, the session information persists on the device until it can successfully send the information to the Kony Fabric server.

For more information on application session, refer Standard Report Docs.

> ***Note:*** The sessions created by native Fabric SDKs are interactive.

39.6.5 Invoking an Identity Service

The following are the methods you can use for an identity service.

- Login with provider type as Basic
- Login with provider type as OAuth/SAML
- Custom OAuth Login
- Get Backend Token
- Logout
- Login Status

39.6.5.1 Login with provider type as Basic

```
 // Sample code to log in using basic type provider
KNYIdentityService * identityService = [
    [KNYIdentityService alloc]
    initWithIdentityName: @"<identity-name>"
];

[identityService loginInBackgroundWithUsername: @"<user-name>"
    password: @"<password>"
    completion: ^ (BOOL succeeded, NSError * error) {
        if (succeeded) {
            // handle the success case here
        } else {
            // handle the failure case here
            // use the details in 'error'
        }
    }
];
```

> ***Note:*** Call these methods only after successful completion of -[KNYClient initializeInBackgroundWithAppKey:appSecret:serviceURLString:completion]. For more details, refer to Init method.

> ***Important:*** When you select Kony User Repository as the identity type, the system does not allow you to provide an identity name.
> To use Kony User Repository as authentication service, ensure that the value for `initWithIdentityName:` method must be set as `userstore` in the KNYIdentityService class. If you set it with any other value (for example, Kony User Repository, User Store or Cloud Repository), the system throws an error.

39.6.5.2 Login with provider type as OAuth/SAML

```
// Sample code to log in using Oauth/SAML type provider
KNYIdentityService * identityService = [
    [KNYIdentityService alloc] initWithIdentityName: @"&lt;identity-
name&gt;"
];

UIView * loginView = nil;
// Setup the view such that is visible in the app.
// The following call will add a WKWebView as subview to the
'loginView' and remove after the authentication
[identityService loginWithParentView: loginView completion: ^ (BOOL
succeeded, NSError * error) {
    if (succeeded) {
        // handle the success case here
    } else {
        // handle the failure case here
        // use the details in 'error'
    }
}];
```

> ***Important:*** This is a synchronous method, and it displays a WKWebView. This method must be invoked from your iOS application's main thread only.

39.6.5.3 Custom OAuth Login

```
 // Sample code to Custom OAuth / Custom login
KNYIdentityService *identityService =
[
    [KNYIdentityService alloc]
    initWithIdentityName: @"<identity-name>"
];

[identityService loginInBackgroundWithcustomParams: @{
        "username": @"username"
    }
    completion: ^ (BOOL succeeded, NSError * error) {
        if (succeeded) {
            // handle the success case here
        } else {
            // handle the failure case here
            // use the details in 'error'
        }
    }
];
```

39.6.5.4 Get Backend Token

```
 // Sample code to get backend token for provider
NSDictionary * backendToken = [identityService backendToken];
```

39.6.5.5 Logout

```
 // Sample code to logout from auth service
if (identityService) {
```

```
    [identityService logoutInBackgroundUsingCompletionBlock: ^ (BOOL
succeeded,
        NSError * errorL) {
        if (succeeded)
        //code if logout is successful

        else
        //code if logout fails
    }];
}
```

39.6.5.6 Login Status

```
 // Sample code to get login status
BOOL status = [identityService isLoggedIn];
If(status)
// code if logged in
else
//code if not logged in
```

39.6.6 Invoking an Integration Service

```
 // Sample code to fetch the integration service details
KNYIntegrationService * integration = [
    [KNYIntegrationService alloc]
    initWithServiceName: @"<service-name>"
];

NSDictionary * headers = @ {@
    "<header-name1>": @"<header-value1>",
    @"<header-name2>": @"<header-value2>"
};

NSDictionary * parameters = @ {@
```

```
    "<param-name1>": @"<param-value1>",
    @"<param-name2>": @"<param-value2>"
};

[integration invokeOperationInBackgroundWithOperationName:
@"<operation-name>"
    headers: headers
    parameters: parameters
    completion: ^ (NSDictionary * objects, NSError * error) {
        if (error == nil) {
            // use data in 'objects' returned by the operation
            // ...
        } else {
            // handle the failure case here
            // use the details in 'error'
        }
    }
];
```

39.6.7 Invoking a Messaging Service

The following are the methods you can use for a messaging service.

- Registering and Getting Messages from Kony Fabric Messaging
- Unregistering from Messaging Service
- Updating GeoLocation
- Fetching All messages in the Message Queue
- Fetching a complete message for long messages using message ID
- Mark a message as read

39.6.7.1 Registering and Getting Messages from Kony Fabric Messaging

To register, use the following code snippet to call the method. To send a message to the registered device, go to Kony Farbic portal and send the message. You will receive a push notification message.

In the - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions: (NSDictionary *), launchOptions method registers for remote notification and calls the KNYClient Intialize method. For more details, refer Initializing the Kony Fabric Client SDK

In the - (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *) deviceToken method, call the following code to register and receive push notifications.

```
 KNYMessagingService * messaging = [KNYMessagingService
sharedMessagingService];

//
// Register with Messaging Service:
//

// the device token received in app delegate method
// - (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken: (NSData * )
deviceToken
NSData * deviceToken;

[messaging registerInBackgroundWithDeviceToken: deviceToken
    alias: @"<alias>"
    completion: ^ (BOOL succeeded, NSError * error) {
        if (succeeded) {          // ...
                } else {        // handle the failure case
                    // use the details in 'error'
        }
    }
];
```

> ***Note:*** Call these methods only after successful completion of -[KNYClient initializeInBackgroundWithAppKey:appSecret:serviceURLString:completion]. For more details, refer to Init method.

39.6.7.2 Unregistering from Messaging Service

This API stops receiving messages.

```
 [messaging unregisterInBackgroundUsingCompletionBlock: ^ (BOOL
succeeded, NSError * error) {
    if (succeeded) {
        // ...
    } else {
        // handle the failure case
        // use the details in 'error'
    }
}];
```

39.6.7.3 Updating GeoLocation

```
[messaging updateGeoLocationInBackgroundWithLatitude: 28.449595 //
note: appropriate value must be used
    longitude: -81.481600 // note: appropriate value must be used
    locationName: @"Sand Lake Rd, Orlando, FL 32819" // note:
appropriate value must be used
    completion: ^ (BOOL succeeded, NSError * error) {
        if (succeeded) {
            // ...
        } else {
            // handle the failure case
            // use the details in 'error'
        }
```

```
    }
];
```

39.6.7.4 Fetching All messages in the Message Queue

```
[messaging fetchAllMessagesInBackgroundWithStartIndex: 0 // note:
appropriate value must be used
    pageSize: 10 // note: appropriate value must be used
    completion: ^ (NSDictionary * objects, NSError * error) {
        if (error == nil) {
            // use data in 'objects' returned by the operation
            // ...
        } else {
            // handle the failure case
            // use the details in 'error'
        }
    }
];
```

39.6.7.5 Fetching a complete message for long messages using message ID

```
 [messaging fetchMessageContentInBackgroundWithMessageId: @"<message-
id>"
    completion: ^ (NSString * string, NSError * error) {
        if (error == nil) {
            // use message in 'string' returned by the operation
            // ...
        } else {
            // handle the failure case
            // use the details in 'error'
        }
    }
];
```

39.6.7.6 Mark a message as read

```
[messaging markMessageReadInBackgroundWithMessageId: @"<message-id>"
    completion: ^ (BOOL succeeded, NSError * error) {
        if (succeeded) {
            // ...
        } else {
            // handle the failure case
            // use the details in 'error'
        }
    }
];
```

39.6.8 Invoking a Sync Service

- Getting Sync Instance
- Initializing a Sync Service
- Encrypting the Offline Database
- Creating a Sync Object
- Error Codes
- Create, Read, Update, and Delete (CRUD) operations in Native SDK
- Updating a Sync Object
- Retrieving an Object
- Deleting an Object
- Pushing (or syncing) Changes to the Sync Server

39.6.8.1 Getting Sync Instance

To get sync service instance pass context of the activity.

```
 KNYSync * sync = nil;@
try {
    sync = [KNYSync sharedInstance];
}@
catch (KNYException * exception) { //failed to encrypt offline
database
} //start to be removed
```

39.6.8.2 Initializing a Sync Service

Before using any sync related API you have to initialize sync services.

Initializing Sync

```
 @try {
    [
        [KNYSync sharedInstance] initSync
    ];
    //init sucess
}@catch (KNYException * exception) {
    //init failure
}
```

Initializing Sync in Background

```
  [
    [KNYSync sharedInstance] initSyncInBackground: ^ (BOOL succeeded,
NSError * error) {
        if (error) {
            //init failed
        } else {
            //init success
        }
    }
];
```

39.6.8.3 Encrypting the Offline Database

To encrypt offline database (SQLite) on device we use SQL Cipher. Encryption has to be specified while initializing sync services. After the sync service is initialized without encryption, you cannot encrypt it. If you want to use encryption, you must initialize the sync service with encryption for the first time. After you encrypt the database, you cannot decrypt it.

```
 KNYSync * sync = nil;
@try {
    sync = [KNYSync sharedInstance];
    [sync initSyncWithPassPhrase: @"somekey"];
    //successfully encrypted offline database
}
@catch (KNYException * exception) {
    //failed to encrypt offline database
}
}
```

39.6.8.4 Creating a Sync Object

1. Create a Sync Object

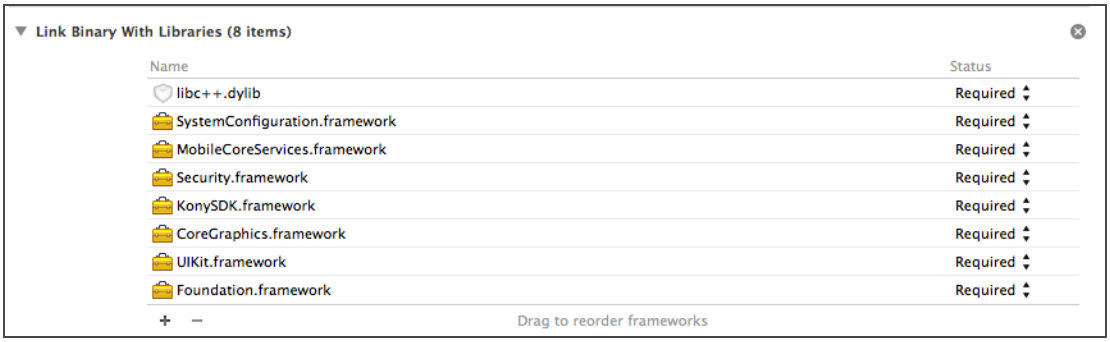
   You need to have the following files to use Sync SDKs:

   Once you upload the sync config file, the system generates code. For iOS, the generated code will contain the following two packages:

   - Metadata Files:
     - SyncGeneratedMetatadata.h
     - SyncGeneratedMetatadata.mm

- Sync Object Files:
  - SyncObjectName.h
  - SyncObjectName.mm

2. Configure native SDK in iOS.
   a. Add the generated files in step1 in your project.
   b. Link the following libraries in your project.



39.6.8.5 Error Codes

The following is a list of error codes for iOS platform along with the corresponding causes and error messages:

Error Code	Cause	Error Message	Comments
KNY1000E		Unknown Error	
KNY1001E		Please initialize init before calling any other API	
KNY1002E		Out of memory	cplusplus > bad_alloc

Error Code	Cause	Error Message	Comments
KNY1003E		Input Output Exception	cplusplus > ios_base
KNY1004E		Error occurred while creating database path:<dbPath>	
KNY1005E		Null Pointer Exception	
KNY1006E		Error occurred while parsing metadata	
KNY1007E		Row Doesn't Exist	
KNY1008E		Session in progress	
KNY1009E		Cannot delete as child record(s) with cascades false exist for this record	
KNY8888E		whatever message comes from server	
KNY1010E		UpgradeRequired In Progress	
KNY2006E	sync.initSync	class not found exception	Supported Doc
KNY3000E	any native NSExceptoin	native iOS error	

> *Important:* For asynchronous APIs, NSError Codes are the same as other error codes. However, the NSError Codes will not have a KNY prefix or an E suffix, because as errors have errorcode as numbers, not string. Also, all errors will have com.kony.mobileFabric.sync as their ErrorDomain.

Predefined iOS Exception

If any API throws a predefined iOS exception, refer to the below link:
https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Exceptions/Concepts/PredefinedExceptions.html #//apple_ref/doc/uid/20000057-BCIGHECA

For any network errors, refer to the below link:
https://developer.apple.com/library/mac/documentation/Networking/Reference/CFNetworkErrors/index.html#//apple_ref /c/tdef/CFNetworkErrors

39.6.8.6 Create, Read, Update, and Delete (CRUD) operations in Native SDK

Create/Update

For both Create and Update, we have the same APIs. If object passed is a new object then create happens, otherwise if object is fetched from device update happens.

Creating a Record

API Used: public <T extends SyncObject> void save(T syncObject) throws KonyException

Sample Code

```
       API Used: -(void) saveSyncObject: (KNYSyncObject * ) knySyncObject;
CODE: @try {
    Categories * cat = [
        [Categories alloc] init
    ];
    [cat setCategoryName: @"Fruits"];
    [cat setDescription: @"Apple"];
    [
        [KNYSyncDataStore sharedInstance] saveSyncObject: cat
```

```
    ];
    //save success
}@
catch (KNYException * e) {
    //save failed
}
```

Creating a Record in Background

API Used: public <T extends SyncObject> void saveInBackground(T dataObject, SyncObjectCallback<T> callback)

Sample Code:

```
 API Use - (void) saveSyncObjectInBackground: (KNYSyncObject * )
knySyncObject
withCompletionBlock: (KNYSyncObjectResultBlock) result;

Categories * cat;@
try {
    cat = [
        [Categories alloc] init
    ];
    [cat setCategoryName: @"Fruits"];
    [cat setDescription: @"Apple"];
}@
catch (KNYException * e) {
    NSLog(@"Exception occurred: % @", e);
}
[
    [KNYSyncDataStore sharedInstance] saveSyncObjectInBackground: cat
    withCompletionBlock: ^ (KNYSyncObject * syncObject, NSError *
error) {
        if (error) {
```

```
            //error in saving syncobject in background
        } else {
            //successfully saved the syncobject in background
        }
    }
];
```

Bulk Create

```
 -(void) bulkSave: (NSArray * ) knySyncObjects;
@try {
    Categories * cat = [
        [Categories alloc] init
    ];
    [cat setCategoryName: @"Fruits"];
    [cat setDescription: @"Apple"];

    Categories * cat1 = [
        [Categories alloc] init
    ];
    [cat1 setCategoryName: @"Fruits1"];
    [cat1 setDescription: @"Apple1"];

    Categories * cat2 = [
        [Categories alloc] init
    ];
    [cat2 setCategoryName: @"Fruits2"];
    [cat2 setDescription: @"Apple2"];

    NSArray * catArray = @ [cat, cat1, cat2];
    [
        [KNYSyncDataStore sharedInstance] bulkSave: catArray
    ];
    //bulk save success
```

```
}
@catch (KNYException * exception) {
    //bulk save failed
}
```

Bulk Create in Background

```
 KNYQuery * query;
NSArray * catArray;

@try {
    query = [
        [KNYSyncDataStore sharedInstance] createQueryWithClassObj:
[Categories class]
    ];
    [query addWhereClause: @"CategoryName='Fruits'"];
    catArray = [
        [KNYSyncDataStore sharedInstance] executeQuery: query
    ];
    for (Categories * cat in catArray) {
        [cat setDescription: @"Fruits Updated"];
    }
}
@catch (KNYException * exception) {
    // Catch the exception and print if required.
}
[
    [KNYSyncDataStore sharedInstance] bulkSaveInBackground: catArray
    withCompletionBlock: ^ (NSArray * objects, NSError * error) {
        if (error) {
            //bulk save failed in background
        } else {
            //bulk save success in background
        }
```

```
    }
];
```

Bulk Update

```
 @try {
    KNYQuery * query = [
        [KNYSyncDataStore sharedInstance] createQueryWithClassObj:
[Categories class]
    ];
    [query addWhereClause: @"CategoryName='Fruits'"];
    NSArray * catArray = [
        [KNYSyncDataStore sharedInstance] executeQuery: query
    ];
    for (Categories * cat in catArray) {
        [cat setDescription: @"Fruits Updated"];
    }
    [
        [KNYSyncDataStore sharedInstance] bulkSave: catArray
    ];
    //bulk save success
}
@catch (KNYException * exception) {
    //bulk save failed
}
```

Bulk Update in Background

```
  KNYQuery * query;
NSArray * catArray;
@try {
    query = [
        [KNYSyncDataStore sharedInstance]
        createQueryWithClassObj: [Categories class]
```

```
    ];
    [query addWhereClause: @"CategoryName='Fruits'"];
    catArray = [
        [KNYSyncDataStore sharedInstance]
        executeQuery: query
    ];
    for (Categories * cat in catArray) {
        [cat setDescription: @"Fruits Updated"];
    }
}
@catch (KNYException * exception) {
    // Catch the exception and print if required.
}
[
    [KNYSyncDataStore sharedInstance]
    bulkSaveInBackground: catArray
    withCompletionBlock: ^ (NSArray * objects, NSError * error) {
        if (error) {
            //bulk save failed in background
        } else {
            //bulk save success in background
        }
    }
];
```

39.6.8.7 Updating a Sync Object

Updating a Record

```
        @try {
    KNYPrimaryKey * pk = [
        [KNYPrimaryKey alloc] init
    ];
    [pk setAttributeForKey: @"CategoryID"
```

```
        value: @"1"
    ];
    Categories * cat = (Categories * )[[KNYSyncDataStore
        sharedInstance
    ] getSyncObjectWithClass: [Categories class] primaryKey: pk];

    //update the record
    [cat setCategoryName: @"Fruits Updated"];
    [cat setDescription: @"Apple Updated"];
    [
        [KNYSyncDataStore sharedInstance] saveSyncObject: cat
    ];
    //save success
}
@catch (KNYException * exception) {
    //save failed
}
```

Updating a Record in Background

```
 KNYPrimaryKey * pk = [
    [KNYPrimaryKey alloc] init
];
[pk setAttributeForKey: @"CategoryID"
    value: @"1"
];
[
    [KNYSyncDataStore sharedInstance]
    getSyncObjectWithClassInBackground: [Categories class]
    primaryKey: pk completionBlock: ^ (KNYSyncObject * syncObject,
        NSError * error) {
        if (error) {
            //get syncObject failed
        } else {
```

```
            Categories * cat = (Categories * )(syncObject);
            //update the record
            [cat setCategoryName: @"Fruits Updated"];
            [cat setDescription: @"Apple Updated"];
            [
                [KNYSyncDataStore sharedInstance]
                saveSyncObjectInBackground: cat
                withCompletionBlock: ^ (KNYSyncObject * syncObject,
NSError * error) {
                    if (error) {
                        //save failed
                    } else {
                        //save success
                    }
                }
            ];
        }
    }
];
```

39.6.8.8 Retrieving an Object

Executing Queries

Query class can be used to define queries.

Execute Query

```
@try {
    KNYQuery * query = [
        [KNYQuery alloc] initWithSyncObject: [Categories class]
    ];
    [query addSelectColumn: @[@"CategoryId"]];
    [query addWhereClause: @"CategoryName='Fruits'"];
    NSArray * catArray = [
```

```
        [KNYSyncDataStore sharedInstance] executeQuery: query
    ];
    //query success
}
@catch (KNYException * exception) {
    //query failed
}
```

Execute Query in Background

```
 KNYQuery * query = [
    [KNYQuery alloc] initWithSyncObject: [Categories class]
];
[query addSelectColumn: @[@"CategoryId"]];
[query addWhereClause: @"CategoryName='Fruits'"];
[
    [KNYSyncDataStore sharedInstance] executeQueryInBackground: query
    withCompletionBlock: ^ (NSArray * objects, NSError * error) {
        if (error) {
            //query failed
        } else {
            //query success
        }
    }
];
```

Getting an Object

To retrieve an object from database using its primary key.

getObject

```
 @try {
    KNYPrimaryKey * pk = [
        [KNYPrimaryKey alloc] init
    ];
```

```
    [pk setAttributeForKey: @"CategoryID"
        value: @"1"
    ];
    Categories * cat = (Categories * )[[KNYSyncDataStore
sharedInstance] getSyncObjectWithClass: [Categories class] primaryKey:
pk];
    //getSyncObject success
}
@catch (KNYException * exception) {
    //getSyncObject failed
}
```

getObject in background

```
    KNYPrimaryKey * pk = [
    [KNYPrimaryKey alloc] init
];
[pk setAttributeForKey: @"CategoryID"
    value: @"1"
];
[
    [KNYSyncDataStore sharedInstance]
getSyncObjectWithClassInBackground: [Categories class]
    primaryKey: pk completionBlock: ^ (KNYSyncObject * syncObject,
NSError * error) {
        if (error) {
            //getSyncObject failed
        } else {
            //getSyncObject success
        }
    }
];
```

Fetching an Object

To fetch all the details of a partially fetched object.

Fetching an Object

```
 @try {
    KNYQuery * query = [
        [KNYQuery alloc] initWithSyncObject: [Categories class]
    ];
    [query addSelectColumn: @[@"CategoryId"]];
    [query addWhereClause: @"CategoryName='Fruits'"];
    NSArray * catArray = [
        [KNYSyncDataStore sharedInstance] executeQuery: query
    ];
    //fill the partial object
    for (Categories * cat in catArray) {
        [
            [KNYSyncDataStore sharedInstance] fetchDetails: cat
        ];
    }
    //fetch success
}
@catch (KNYException * exception) {
    //fetch failed
}
```

Fetching an Object in Background

```
 KNYQuery * query = [
    [KNYQuery alloc]
    initWithSyncObject: [Categories class]
];
[query addSelectColumn: @[@"CategoryId"]];
```

```
[query addWhereClause: @"CategoryName='Fruits'"];
NSArray * catArray = [
    [KNYSyncDataStore sharedInstance] executeQuery: query
];
if (0 & lt;
    [catArray count]) {
    Categories * cat = [catArray objectAtIndex: 0];
    [
        [KNYSyncDataStore sharedInstance]
        fetchDetailsInBackground: cat
        completionBlock: ^ (KNYSyncObject * syncObject, NSError *
error) {
            if (error) {
                //fetch failed
            }
        }
    ];
}
```

39.6.8.9 Deleting an Object

Delete

```
  @try {
    KNYPrimaryKey * pk = [
        [KNYPrimaryKey alloc] init
    ];
    [pk setAttributeForKey: @"CategoryID"
        value: @"1"
    ];
    Categories * cat = (Categories * )[[KNYSyncDataStore
sharedInstance] getSyncObjectWithClass: [Categories class] primaryKey:
pk];
    [
```

```
        [KNYSyncDataStore sharedInstance] deleteSyncObject: cat
    ];
    //delete success
}
@catch (KNYException * exception) {
    //delete failure
}
```

Delete in Background

```
 KNYPrimaryKey * pk = [
    [KNYPrimaryKey alloc] init
];
[pk setAttributeForKey: @"CategoryID"
    value: @"1"
];
Categories * cat = (Categories * )[[KNYSyncDataStore sharedInstance]
getSyncObjectWithClass: [Categories class] primaryKey: pk];
[
    [KNYSyncDataStore sharedInstance]
    deleteSyncObjectInBackground: cat
    withCompletionBlock: ^ (NSError * error) {
        if (error) {
            //delete failure
        } else {
            //delete success
        }
    }
];
```

Bulk Delete

```
 @try {
    KNYQuery * query = [
```

```
        [KNYSyncDataStore sharedInstance]
        createQueryWithClassObj: [Categories class]
    ];
    [query addWhereClause: @"CategoryName='Fruits'"];
    NSArray * catArray = [
        [KNYSyncDataStore sharedInstance]
        executeQuery: query
    ];
    [
        [KNYSyncDataStore sharedInstance] bulkDelete: catArray
    ];
    //bulk delete success
}
@catch (NSException * exception) {
    //bulk delete failed
}
```

Bulk Delete in Background

```
 KNYQuery * query;
NSArray * catArray;

@try {
    query = [
        [KNYSyncDataStore sharedInstance]
        createQueryWithClassObj: [Categories class]
    ];
    [query addWhereClause: @"CategoryName='Fruits'"];
    catArray = [
        [KNYSyncDataStore sharedInstance]
        executeQuery: query
    ];
}
@catch (KNYException * exception) {
```

```
    // Catch the exception and print if required.
}
[
    [KNYSyncDataStore sharedInstance]
    bulkDeleteInBackground: catArray
    withCompletionBlock: ^ (NSError * error) {
        if (error) {
            // bulk delete failed
        } else {
            //bulk delete success
        }
    }
];
```

39.6.8.10 Pushing (or syncing) Changes to the Sync Server

The startSessionInBackgroundAPI can be used to sync (upload and download/push and pull) changes between device and sync server. This is purely asynchronous API. To get notifications during the API execution you can implement SyncListener interface and pass object of implementing class to synchronize.

```
 KNYSync * sync = [KNYSync sharedInstance];
//implement <KNYSyncEventDelegate>; protocol and setSynceventDelegate
sync.syncEventDelegate = self; //or any class object
that implemented & lt;
KNYSyncEventDelegate & gt;
protocol[sync startSessionInBackground];
```

Configuring Various Sync Options

You can use sync options class to configure various sync options like:

- **Upload Error Policy**: This policy can be used to configure whether to continue or abort on getting upload error. Default value is continue on error.

- **Schema Upgrade Policy**: This policy is used to configure what action to be taken when schema upgrade is available. Default value is upload and upgrade.
- **Enabling or disabling download/upload for a scope**: Default value is continue on error.
- **Adding filter for syncing**: This can be used to set download filters. By default, all pass filters are used.
- **Configuring remove after upload policy**: this can be used to configure removal of records instance from device after successful upload. By default, this option is disabled for all scopes.

Configuring NetworkOptions

The NetworkOptions class can be used to configure various options for APIs that use the network.

For example, startSessionInBackground, performUpgradeInBackground, and isUpgradeRequiredInBackground.

Following options can be configured:

1. **Network Timeout**: the time for which device should wait for server to respond. If server does not respond in the specified time, an error is thrown. Default value is ten seconds.
2. **Retry Count**: Retry count specifies how many times a request can be sent to server in case of a server error. Default value is five.
3. **Retry Wait Time**: Time between two retries. Default value is five seconds.
4. **Retry Listener**: The listener which should be invoked on each retry.

39.6.9 Invoking an Object Service

Kony provides programmatic access to the backend data through Online Object Services. You must perform the following steps to gain the access:

1. Acquire a current instance of your object service.

2. Use the object service instance with data transfer objects to communicate as needed with your backend.

To acquire a current instance of your object service, refer to getObjectService Method documentation.

To know more about the methods that act on the Kony Fabric endpoint directly, refer to OnlineObjectService Class documentation.

getObjectService Method

The **getObjectService Method** gets the current instance of the object service. The getObjectService method is invoked on the SDK instance; **init** must run successfully before invoking this method.

39.6.9.1 Syntax

```
(KNYObjectService*)getObjectService:(NSString*)serviceName
                               error:(NSError **)error
```

39.6.9.2 Return Type

ObjectService Instance

39.6.9.3 Parameters

Input Parameter	Type	Description	Required
servicename	String	Name of the object service	Yes

39.6.9.4 Code

```
 NSError *error = nil;
KNYObjectService *objSvc = [client
getObjectService:@"<objectservicename>" error:&error];
if (error != nil) {
    NSLog(@"Error in getting objectservice : %@", error);

} else {
```

```
    NSLog(@"Successfully retrieved objectservice instance");
}
```

39.6.9.5 OnlineObjectService Class

OnlineObjectService Class provides methods that perform operations acting on the Kony Fabric endpoint, including basic CRUD. An instance of OnlineObjectService is returned by the getObjectService Method.

Methods

The following methods are used by the OnlineObjectService class and its instantiations:

- Create
- Update
- Delete
- Fetch
- getMetaDataofAllObjects
- getMetaDataofObject

Create

Creates an object in the Kony Fabric endpoint.

39.6.9.6 Syntax

```
(void)createWithDataObject:(KNYDataObject *)dataobject
                   headers:(NSDictionary *)headers
               queryparams:(NSDictionary *)params
                completion:(KNYDictionaryResultBlock)completion;
```

39.6.9.7 Return Type

None

39.6.9.8 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	NSDictionary	Key/value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	NSDictionary	Key/ value pairs of query parameters that are appended to the URL while making a network call	Optional
KNYResultDictionaryCallback	KNYResultDictionaryCallback	Callback to handle success response and error on completion of the API.	Yes

39.6.9.9 Code

```
NSError *error = nil;
objSvc = [client getObjectService:@"<ObjectServiceName>"
```

```
error:&error];
if(error != nil)
{
    KNYDataObject *dataObj = [[KNYDataObject alloc]
initWithServiceName:@"<objectname>"];
    NSMutableDictionary *record = [[NSMutableDictionary alloc]init];
    record[@"<columnname>"] = @"<columnvalue>";
    dataObj.record = record;

    NSMutableDictionary *_headers = [[NSMutableDictionary alloc]init];
     _headers[@"<headername>"] = @"<headervalue>";

    [objSvc createWithDataObject:dataObj
                         headers:_headers
                     queryparams:nil
                      completion:^(NSDictionary *objects, NSError
*error) {
                          if(!error){
                              NSLog(@"Successfully created object!");

                          }else{
                              NSLog(@"Failed creating object with
Error:%@", error);

                          }
                      }];

}
```

Update

Updates an object in the Kony Fabric endpoint.

39.6.9.10 Syntax

```
(void)updateWithDataObject:(KNYDataObject *)dataobject
                   headers:(NSDictionary *)headers
               queryparams:(NSDictionary *)queryparams
                completion:(KNYDictionaryResultBlock)completion;
```

39.6.9.11 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	NSDictionary	Key/value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	NSDictionary	Key/ value pairs of query parameters that are appended to the URL while making a network call	Optional

Input Parameter	Type	Description	Required
KNYResultDictionaryCallback	KNYResultDictionaryCallback	Callback to handle success response and error on completion of the API.	Yes

39.6.9.12 Code

```
NSError *error = nil;
KNYObjectService *objSvc = [client
getObjectService:@"<ObjectServiceName>" error:&error];
if(error != nil)
{

KNYDataObject *dataObj = [[KNYDataObject alloc]
initWithServiceName:@"<objectname>"];
    NSMutableDictionary *_record = [[NSMutableDictionary alloc]init];
    _record[@"<columnname>"] = @"columnvalue";
    dataObj.record = _record;

NSMutableDictionary *_headers = [[NSMutableDictionary alloc]init];
_headers[@"<headername>"] = @"<headervalue>";
    [objSvc updateWithDataObject:dataObj
                         headers:_headers
                     queryParams:nil
                      completion:^(NSDictionary *objects, NSError
*error) {
                          if(!error){
                              NSLog(@"Successfully updated object!");

                          }else{
```

```
                                NSLog(@"Failed updating object with
Error:%@", error);

                            }
                        }];
}
```

Delete

Deletes an object in the Kony Fabric endpoint.

39.6.9.13 Syntax

```
(void)deleteWithDataObject:(KNYDataObject *)dataobject
                   headers:(NSDictionary *)headers
               queryparams:(NSDictionary *)queryparams
                completion:(KNYDictionaryResultBlock)completion;
```

39.6.9.14 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	NSDictionary	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional

Input Parameter	Type	Description	Required
queryParams	NSDictionary	Key/ value pairs of query parameters that are appended to the URL while making a network call	Optional
KNYResultDictionaryCallback	KNYResultDictionaryCallback	Callback to handle success response and error on completion of the API.	Yes

39.6.9.15 Code

```
NSError *error = nil;
KNYObjectService *objSvc = [client
getObjectService:@"<ObjectServiceName>" error:&error];
if(error != nil)
{
 KNYDataObject *dataObj = [[KNYDataObject alloc]
initWithServiceName:@"<objectname>"];
    NSMutableDictionary *_record = [[NSMutableDictionary alloc]init];

_record[@"<columnname>"] = @"columnvalue";

    dataObj.record = _record;
NSMutableDictionary *_headers = [[NSMutableDictionary alloc]init];
_headers[@"<headername>"] = @"<headervalue>";
    [objSvc deleteWithDataObject:dataObj
                         headers:_headers
```

```
                    queryParams:nil
                     completion:^(NSDictionary *objects, NSError
*error) {
                         if(!error){
                             NSLog(@"Successfully deleted object!");

                         }else{
                             NSLog(@"Failed deleting object with
Erro:%@", error);

                         }
                     }];

}
```

Fetch

Fetches an object from the server.

39.6.9.16 Syntax

```
(void)fetchWithDataObject:(KNYDataObject *)dataobject
                  headers:(NSDictionary *)headers
              queryparams:(NSDictionary *)queryparams
               completion:(KNYDictionaryResultBlock)completion;
```

39.6.9.17 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes

Input Parameter	Type	Description	Required
headers	NSDictionary	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	NSDictionary	Key/ value pairs of query parameters that are appended to the URL while making a network call.	Optional
KNYResultDictionaryCallback	KNYResultDictionaryCallback	Callback to handle success response and error on completion of the API.	Yes

39.6.9.18 Code

```
 NSError *error = nil;
 objSvc = [client getObjectService:@"<ObjectServiceName>"
error:&error];
if(error != nil)
{
KNYDataObject *dataObj = [[KNYDataObject alloc]initWithServiceName:@"
:@"<objectname>"];
    dataObj.ODataUrl = [NSString
stringWithFormat:@"$filter=primarykeycolumn eq
```

```
%@",@"<primarykeyvalue>"];

    NSMutableDictionary *_headers = [[NSMutableDictionary alloc]init];
_headers[@"<headername>"] = @"<headervalue>";
    [objSvc fetchWithDataObject:dataObj
                        headers:_headers
                    queryParams:nil
                     completion:^(NSDictionary *objects, NSError
*error) {
                         if(!error){
                             NSLog(@"Successfully fetch object!");
                         }else{
                             NSLog(@"Failed fetching object with
Erro:%@", error);

                         }
                     }];
}
```

getMetaDataofAllObjects

Gets the metadata associated with the objects that are defined in the service from the server.

39.6.9.19 Syntax

```
(void)getMetadataOfAllObjectsFromServer:(BOOL)fromServer
                                headers:(NSDictionary *)headers
                            queryparams:(NSDictionary *)params
                             completion:(KNYDictionaryResultBlock)
completion
```

39.6.9.20 Parameters

Input Parameter	Type	Description	Required
getFromServer	Boolean	Flag that retrieves the metadata of the object from the server or local store. • **True** - from server • **False** - from local store	Yes
headers	NSDictionary	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	NSDictionary	Key/ value pairs of query parameters that are appended to the URL while making a network call.	Optional

Input Parameter	Type	Description	Required
KNYResultDictionaryCallback	KNYResultDictionaryCallback	Callback to handle success response and error on completion of the API.	Yes

39.6.9.21 Code

```
NSError *error = nil;
KNYObjectService *objSvc = [client
getObjectService:@"<ObjectServiceName>" error:&error];
if(error != nil)
{
[objSvc getMetadataOfAllObjectsFromServer:true
                                    headers:nil
                                queryparams:nil
                                 completion:^(NSDictionary *objects,
NSError *error) {
                                     if(!error){
                                         NSLog(@"successfully
getmetadata of all objects");

                                     }else{
                                         NSLog(@"Failed to
getmetadata of all objects:%@", error);

                                     }
                                 }];

}
```

getMetaDataofObject

Gets the metadata associated with an object that is defined in the service from the server or local store.

39.6.9.22 Syntax

```
(void)getMetadataOfObjectFromServer:(BOOL)fromServer
                         objectName:(NSString *)objectName
                            headers:(NSDictionary *)headers
                        queryparams:(NSDictionary *)params
                         completion:(KNYDictionaryResultBlock)
completion
```

39.6.9.23 Parameters

Input Parameter	Type	Description	Required
getFromServer	Boolean	Flag that retrieves the metadata of the object from the server or local store. • **True** from server • **False** - from local store	Yes
objectName	String	Name of the object	Yes

Input Parameter	Type	Description	Required
headers	NSDictionary	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	NSDictionary	Key/ value pairs of query parameters that are appended to the URL while making a network call.	Optional
KNYResultDictionaryCallback	KNYResultDictionaryCallback	Callback to handle success response and error on completion of the API.	Yes

39.6.9.24 Code

```
 NSError *error = nil;
KNYObjectService *objSvc = [client
getObjectService:@"<ObjectServiceName>" error:&error];


    [objSvc getMetadataOfObjectFromServer:true
                                 objectName:@"<objectName>"
                                    headers:nil
                                queryparams:nil
                                 completion:^(NSDictionary *objects,
```

```
NSError *error) {
                                        if(!error){
                                            NSLog(@"successfully
getmetadata of object");
                                        }else{
                                            NSLog(@"Failed getmetadata
of object:%@", error);

                                        }
                                    }];
```

39.6.10 Invoking a Metrics Service

To use metrics service of Kony Fabric SDK, you must add `#import <CMS/CMS.h>` in the header of your files.

When the iOS SDK is initialized, it automatically collects various standard metrics from a client and the standard metrics will be accessible using the Standard Reports within Kony Fabric Console.

The iOS SDK also provides the ability for a developer to send additional custom metrics from a client app to Kony Fabric back-end to capture additional information. These custom data sets will be accessible using the Custom Reporting feature within Kony Fabric Console where a business analyst can design and share reports using a combination of standard and custom metrics.

Additionally, the iOS SDK provides an Events API that allows an app to track user actions within the app to gain insight into the user journey.The developer can send various standard events such as form entry, touch events, service requests, gestures and errors. The developer can also send custom events to capture any app specific scenarios or transactions. These events can be analyzed within Kony Fabric Console by using the Standard Reports or user defined Custom Reports. For more details, refer to Custom Metrics and Reports Guide.

This section lists all `KNYMetricsService` object APIs.

39.6.10.1 Create an instance of KNYMetricsService

The KNYMetricsService class sets the configuration for APM event reporting.

```
 //Sample code to create an instance of metricService with variable
name "sharedMetricsService"
KNYMetricsService * metricService = [KNYMetricsService
sharedMetricsService];
```

setUserID

The setUserID API sets the user ID for the data gathered from an application. The user ID allows the data to be tracked on a user basis for broad analysis like how many different users used the application. It also helps to track activities of a specific user, which can help in seeing what activities were done before a crash, or what events led to a transaction not passing through. This user ID allows the same user to be tracked across different devices as well.

```
 //Sample code to set up the user ID of application user
[metricService setUserId: @"userid"];
```

> ***Note:*** The UserId related to metrics. The UserId length cannot be more than 100 characters.

sendEvent

The sendEvent API allows a developer to send event details from an application to server for analytics and reporting purposes. The event data is added to a buffer and sent to server as per configuration values set by the developer using setEventConfig API.

```
 //Sample code to send reports
[metricService sendEvent: < eventtype > eventSubType: @"<value1>"
    formID: @"<value2>"
    widgetID: @"<value3>"
    flowTag: @"<value4>"
];
```

sendEvent with meta data

The sendEvent API allows a developer to send event details from an application to server for analytics and reporting purposes. The event data is added to a buffer and sent to server as per configuration values set by the developer using setEventConfig API.
The sendEvent API allows sending custom information for the event as metadata, which takes a dictionary type object.

```
 //Sample code to send reports with metadata
[metricService sendEvent: < eventtype > eventSubType: @"<value1>"
    formID: @"<value2>"
    widgetID: @"<value3>"
    flowTag: @"<value4>"
    metaData: @{
        any custom data which is a dict
    }
];
```

The following are the enums event types with values:

- KNYEventTypeFormEntry = 0
- KNYEventTypeFormExit = 1
- KNYEventTypeTouch = 2
- KNYEventTypeServiceRequest = 3
- KNYEventTypeServiceResponse = 4
- KNYEventTypeGesture = 5
- KNYEventTypeOrientation = 6
- KNYEventTypeError = 7
- KNYEventTypeHandledException = 8

- KNYEventTypeCrash = 9
- KNYEventTypeCustom = 10
- KNYEventTypeServiceCall = 11
- KNYEventTypeAppTransition = 12
- KNYEventTypeAppLoad = 13

> ***Note:*** The EventType is an ENUM.
> The eventSubType, formId, widgetId, and flowTag fields can have max of 256 characters.

The following are the parameters for an event type:

- eventType - string literal for formID can be null
- eventSubType - string literal for eventSubType (max 256 characters)
- formID - string literal for formID (max 256 characters)
- widgetID - string literal for widgetID (max 256 characters)
- flowTag - string literal to override flow tag (max 256 characters)
- metaData - string literal that can be set by developer while setting a custom event for sending custom data as part of an event.

flushEvents

The flushEvents API allows a developer to force events to be sent to the server. The entire current event buffer is loaded and sent to the server for processing. The flushEvents API is used as an override to send event data to server before the configured value or a service call that flushes the buffer.

```
 //Sample code to flushEvents
[metricService flushEvents];
```

eventsInBuffer

The eventsInBuffer returns a list of the buffered events.

```
 //Sample code to eventsInBuffer
[metricService eventsInBuffer];
```

setFlowTag

The setFlowTag API sets an event flow tag to be associated with all new events that are reported by using thesendEvent API.The flow tag is used to ease searching event data in terms of application flows like loginflow, searchflow. The setFlowTag also helps sorting and filtering data while building custom reports or running standard reports for the application events.

```
 //Sample code to setFlowTag
[metricService setFlowTag: @"value"];
```

clearFlowTag

The clearFlowTag API clears the currently set event flow tag.

```
 //Sample code to clearFlowTag
[metricService setFlowTag: nil];
```

getFlowTag

The getFlowTag API gets the currently set event flow tag.

```
 //Sample code to getFlowTag
[metricService flowTag];
```

reportError

The reportError API enables an app to report an error event to metrics server.

```
 //Sample code to reportError
[metricService reportError: @"<report error>"
```

```
    errorType: @"<value1>"
    "errorMessage:@" < value2 > " errorDetails:@" < value3 > "];
```

Parameters:

- errorCode - string errorCode can be nil if not applicable.
- errorType - string errorType can be nil if not applicable.
- errorMessage - string errorMessage can be nil if not applicable.
- errorDetails - string(JSON) errorDetails is a json string that can have key-value pairs for the following keys errfile, errmethod, errline, errstacktrace, errcustommsg, errcrashreport, formID, widgetID, and flowTag.

reportHandledException

The reportHandledException API enables apps to report a handled exception event.

```
 //Sample code to send exception to metrics server
[metricService reportHandledException: @"<report error>"
    exceptionType: @"<value1>"
    exceptionMessage: @"<value2>"
    exceptionDetails: @"<value3>"
];
```

Parameters:

- exceptionCode - string exceptionCode can be nil if not applicable.
- exceptionType - string type of exception, such as Eval Error or syntax error. The exceptionType can be nil if not applicable.
- exceptionMessage - string exceptionMessage can be nil if not applicable.
- exceptionDetails - string(JSON) exceptionDetails is a JSON string that can have key-value pairs for the following keys exceptionfile, exceptionmethod, exceptionline, exceptionstacktrace, formID, widgetID, and flowTag.

setEventConfig

The setEventConfig API takes the required values to set the event configuration values. When eventConfigType is - KNYEventConfigTypeBuffer eventBufferAutoFlushCount and eventBufferMaxCount are considered.

```
 //Sample code to setEventConfig

[metricService setEventConfig: < eventConfigType >
eventBufferAutoFlushCount: 10 eventBufferMaxCount: 500];
```

Parameters:

- eventConfigType - sets the current configuration type

  > ***Note:*** Only buffer mode is supported for the eventConfigType currently.

- eventBufferAutoFlushCount - number of events to be buffered before a network call is triggered to post the event data to the server Possible values any positive integer. Default value is 15.
- eventBufferMaxCount - Maximum event buffer count to store the events possible values any positive integer. Default value os 1000.

setBatchSize

The setBatchSize API allows a developer to specify the batch size to be set to flush events. Default batch size is 50.

```
 //Sample code to set batch size to flush events
[metricService setBatchSize: 20];
```

sendCustomMetrics

The sendCustomMetrics API sends custom metrics event.

```
 //Sample code to send data to reporting service with group id as
"formID"
```

```
[metricService sendCustomMetrics: @"formID"
    andData: @{
        @"prodName": @"iPad 16 GB",
        @"prodPrice": @"450.0"];
```

For more details about custom metrics and reports, refer to Custom Metrics and Reports Guide.

Parameters:

- groupId - formID length cannot be more than 250 characters.
- data - data to be send

39.6.10.2 Event Details

For all event details, ts and SID values are automatically filled by MBaaS Client SDK, as part of the reportEvent, reportError and reportHandledException API calls. In case of automatically captured events, flowTag is also automatically filled with the currently set flowTag. Following are event specific details to be used while interfacing with MBaaS SDK.

FormEntry

- API to be used - sendEvent
- evtType - FormEntry
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - null

FormExit

- API to be used - sendEvent
- evtType - FormExit

- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - Dictionary (hash table) that contains the following key value pairs:
    - formdur - Duration spent in form in milliseconds. Optional.

Touch

- API to be used - sendEvent
- evtType - Touch
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - value of the ID property of the widget on which the touch happened
- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget). Touch event is extended to the below widgets along with button onclick.
    - Flex Container/Scroll Container

      onClick

      onTouchStart (if registered)

      onTouchEnd (if registered)
    - Segment

      onRowClick
    - Button

      onClick (already in place, no new changes)

- Image

  onTouchStart (if registered)

  onTouchEnd (if registered)

- Switch

  onSlide

- metadata - null

ServiceRequest

- API to be used - sendEvent
- evtType - ServiceRequest (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata - null

ServiceResponse

- API to be used - sendEvent
- evtType - ServiceResponse (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata
  Dictionary (hash table) contains the following key value pairs:
  - opstatus - Optional
    returned by Kony servers
  - httpcode - HTTP status code
  - resptime - time taken to get the response.

Gesture

- API to be used - sendEvent
- evtType - Gesture (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget where the gesture happened
- WidgetID - value of the ID property of the widget on which the gesture happened
- evtSubType [String]
  `GESTURETYPE_NUMBEROFINPUTS_DIRECTION`
  For example, two finger left swipe - SWIPE_2_LEFT
- metadata - null

Orientation

- API to be used - sendEvent
- evtType - Orientation (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType [String] - any one of the below constants is used
    - PORTRAIT_TO_LANDSCAPE
    - LANDSCAPE_TO_PORTRAIT
- metadata - null

Error

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ErrorCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:
    - **Error – JSON string with below parameters.**
        - Error Code (errcode) (optional)
        - Error Message (errmsg) (optional)
        - Module name (errfile) (optional)
        - Method name (errmethod) (optional)
        - Line number (errline) (optional)
        - Stack trace (errstacktrace) (optional)
        - Custom error message (errcustommsg) (optional). To be used for sending custom error message or params that are not usually expected for a regular error

message. For example, {"errmsg": “failure to parse integer”,"errfile": “accounthandler.js”, “errline”: “189”

Sample Error:

```
{
    "errcode": 1028,
    "errmsg": "Nullpointerexception",
    "errstacktrace":
"java.lang.NullPointerExceptionatMaze.getNumRandOccupants
(Maze.java:118)atP4TestDriver.testMaze(P4TestDriver.java:995)
atP4TestDriver.main(P4TestDriver.java: 116)at__SHELL8.run(__
SHELL8.java: 7)atsun.reflect.NativeMethodAccessorImpl.invoke0
(NativeMethod)  atsun.reflect.NativeMethodAccessorImpl.invoke
(NativeMethodAccessorImpl.java: 39)
atsun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java: 25)
atjava.lang.reflect.Method.invoke(Method.java: 597)
atbluej.runtime.ExecServer$3.run(ExecServer.java: 814)"
}
```

HandledException

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ExceptionCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:

- exceptioncode - Optional
- exceptionev - Optional
- exceptionmsg - Optional
- exceptionfile - Optional
- exceptionmethod - Optional
- exceptionstacktrace - Optional
- exceptioncustommsg - Optional

Sample HandledException

```
{
    "exceptioncode": 1028,
    "exceptionmsg": "Nullpointerexception",
    "exceptionstacktrace":
"java.lang.NullPointerExceptionatMaze.getNumRandOccupants
(Maze.java:118)atP4TestDriver.testMaze(P4TestDriver.java:995)
atP4TestDriver.main(P4TestDriver.java:116)at__SHELL8.run(__
SHELL8.java:7) atsun.reflect.NativeMethodAccessorImpl.invoke0
(NativeMethod)atsun.reflect.NativeMethodAccessorImpl.invoke
(NativeMethodAccessorImpl.java:39)
atsun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:25)
atjava.lang.reflect.Method.invoke(Method.java:597)
atbluej.runtime.ExecServer$3.run(ExecServer.java: 814)"
}
```

Crash

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen

- WidgetID - null
- evtSubType [String]
- metadata
  Dictionary (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errline - Optional
  - errstacktrace - Optional
  - errcrashreport - Optional
  - Plugin version - platform (pluginverplat) (optional). To be used for sending the platform plugin version of the application binary Ex: iOS-6.0.4.GA
  - Plugin versions - IDE (pluginveride) (optional). To be used for comma separated list of name/value pairs representing information of all studio plugins like ide, codegen. Ex: ide:6.0.3.GA, codegen:6.0.3.1.GA.
  - Session ID (sessionid) (optional). Session ID in which the crash occurred. This is usually the previous session's ID as crash data cannot be set in the same session that it crashed.

Sample Crash

```
{
    "errcode": 1028,
    "errmsg": "Nullpointerexception",
    "errstacktrace":
"java.lang.NullPointerExceptionatMaze.getNumRandOccupants
(Maze.java:118)atP4TestDriver.testMaze(P4TestDriver.java:995)
```

```
atP4TestDriver.main(P4TestDriver.java:116)at__SHELL8.run(__
SHELL8.java:7) atsun.reflect.NativeMethodAccessorImpl.invoke0
(NativeMethod)atsun.reflect.NativeMethodAccessorImpl.invoke
(NativeMethodAccessorImpl.java:39)
atsun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:25)
atjava.lang.reflect.Method.invoke(Method.java:597)
atbluej.runtime.ExecServer$3.run(ExecServer.java: 814)"
}
```

Custom

- API to be used - sendEvent
- evtType - Custom
- FormID - any supplied form ID
- WidgetID - any supplied widget ID
- evtSubType - any supplied event subtype
- metadata - string or a dictionary

AppTransition

- API to be used - sendEvent
- evtType - AppTransition
- FormID - Value of the ID property of the form widget where the touch happened
- WidgetID - null

- evtSubType [String] - any one of the below string constants will obtain:
  - Background
  - Foreground
  - metadata
- Dictionary (hash table) that contains the following key value pairs:
  foredur - Duration spent in milliseconds.

AppLoad

- API to be used - sendEvent
- evtType - AppLoad
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - null
- evtSubType [String] – AppId which is the Application ID.
- metadata
- Dictionary (hash table) that contains the following key value pairs:
  loaddur - Duration spent in milliseconds.

39.7 Android SDK

These steps show how to add the Kony-Android-SDK to your project and set up Kony Fabric client.

- Prerequisites
- Downloading Kony Android SDK
- Configuring the Kony Android SDK
- Initializing the Android Client SDK
- Invoking an Identity Service

- Invoking an Integration Service
- Invoking a Messaging Service
- Invoking a Sync Service
- Invoking a Metrics Service
- Invoking an Object Service
- API Reference

  To view the API Reference for Kony Android, click Kony Android docset.

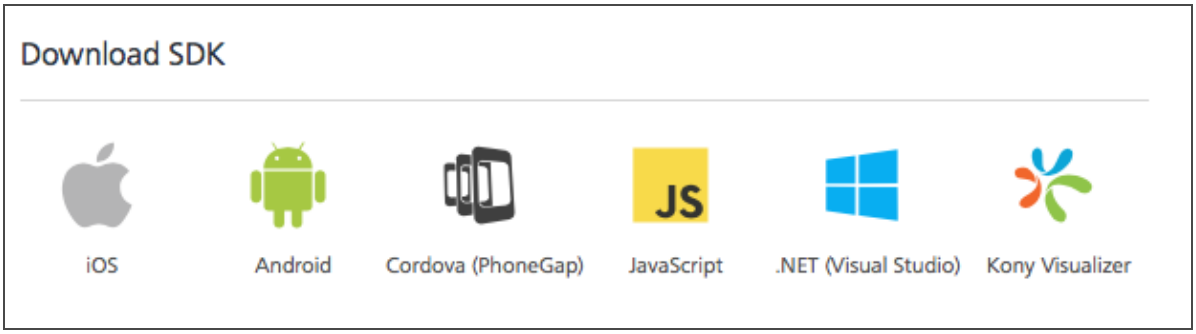
39.7.1 Prerequisites

- Java Development Kit (JDK) - http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u32-oth-JPR
- Eclipse with ADT installed or Android Studio 1.5 or 2.1.
- If you are using an untrusted self-signed (SSL) certificate with Kony Fabric installation, enable the KonyClient.acceptSelfSignedCertificates(); API. By default, native apps do not allow untrusted SSL certificates for HTTPS connection. For more details, refer to SSL API.

39.7.2 Downloading Kony Android SDK Files

To download Kony Android SDK, follow these steps:

1. In the Kony Fabric console, navigate to **Apps** > **SDKs**, and click **Android**. The system prompts you to save the zip file in your local system.



2. Save the `kony-android-sdk.zip` file in your local system.
3. Extract the `kony-android-sdk.zip` file that you just downloaded.

   The `kony-android-sdk` folder contains the following files:

   - kony-sdk.jar
   - kony-sdk.doc
   - LICENSE.txt
   - version.txt
   - libs.zip

39.7.3 Configuring Kony Android SDK

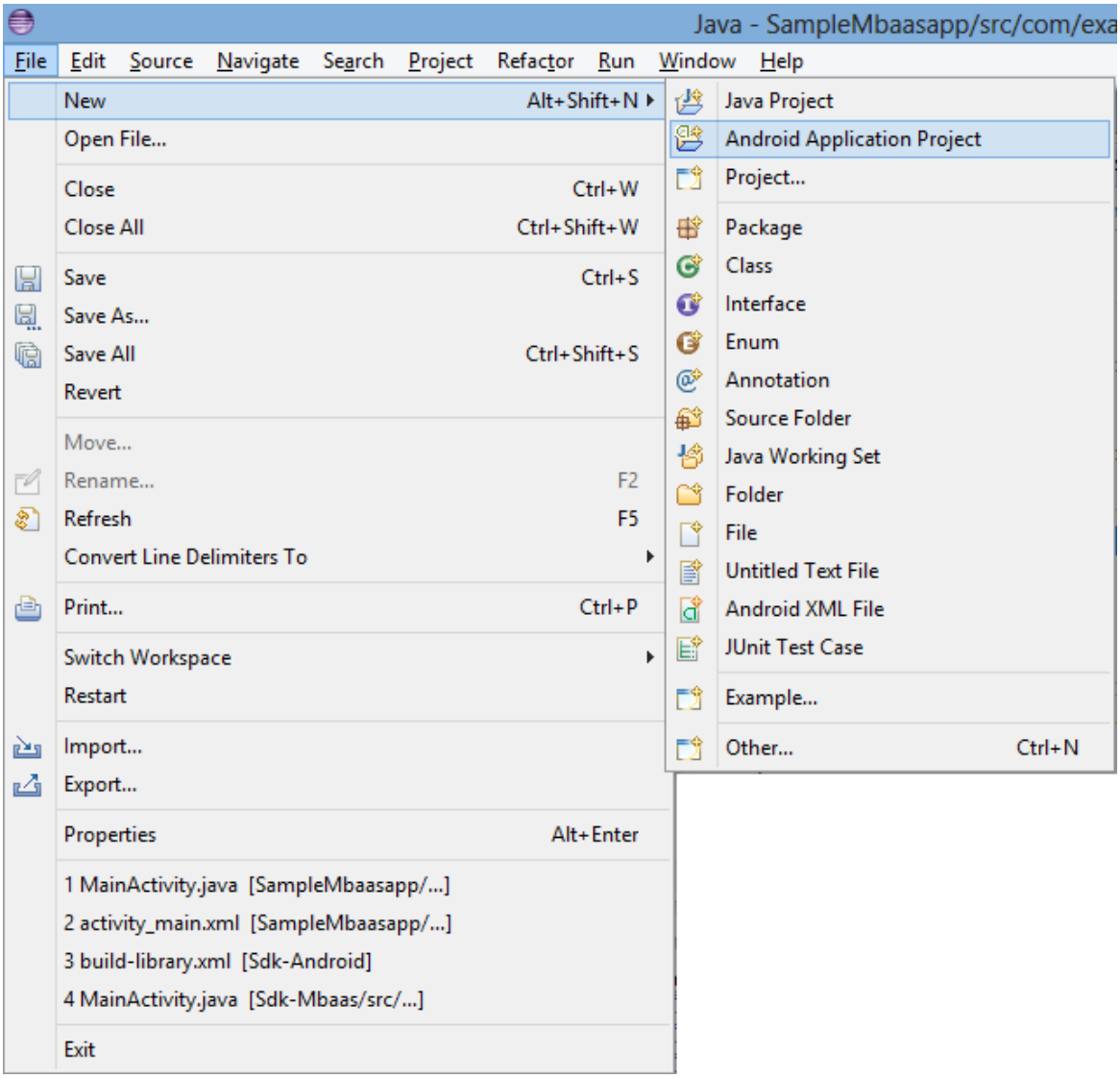
Before using Kony SDK APIs for Android, you must configure Kony Android SDK into your IDEs, such as Eclipse or Android Studio. Configuring the Kony Android SDK involves these steps:

- Configuring kony-sdk.jar to Project - Eclipse
- Configuring kony-sdk.doc to Project - Eclipse
- Configuring kony-sdk.jar to Project - Android Studio

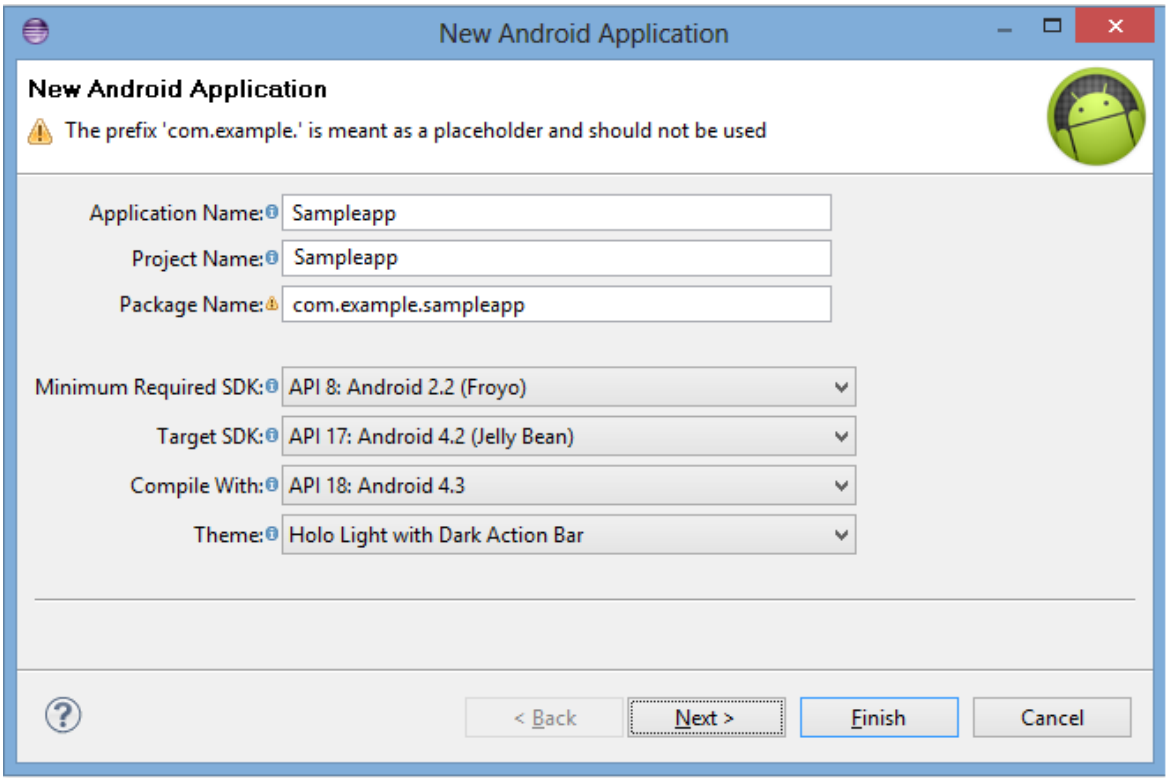
39.7.3.1 Configuring kony-sdk.jar to Project - Eclipse

To configure kony-sdk.jar file, follow these steps:

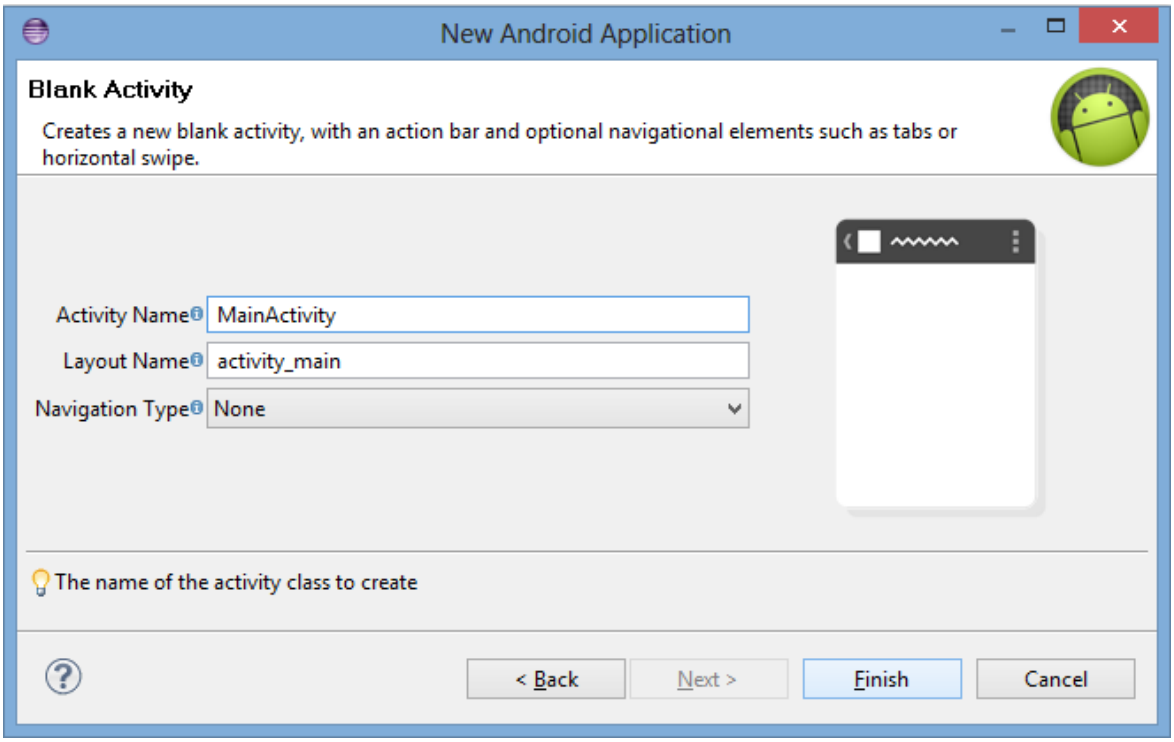
1. Launch Eclipse.
2. In Eclipse, click **File > New > Project > Android Application Project**.



The **New Android Application** screen appears.

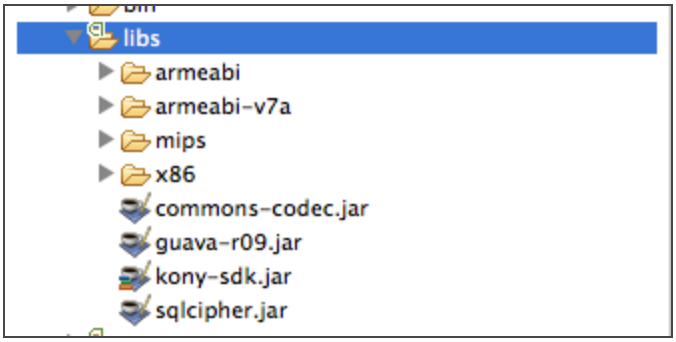


3. In the **New Android Application** dialog, enter the following and click **Next**. (You can leave the rest of the fields unchanged.)
    - **Application Name**: Enter the name of the application.
    - **Project Name**: Enter the name of the project.
    - **Package**: Enter the name of the package.



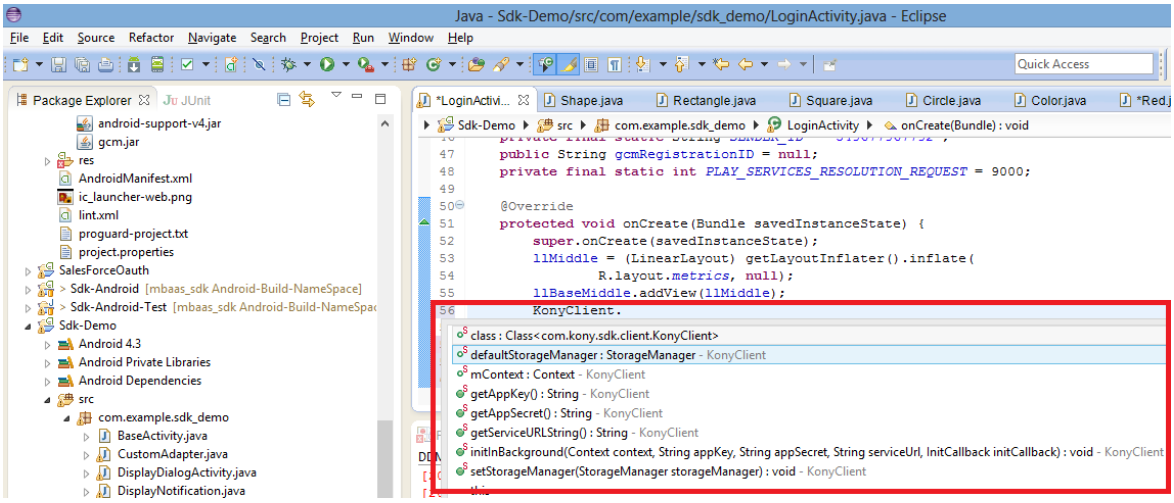
4. In the **Blank Activity** screen, enter the following details for your app and click **Finish**. (You can leave the rest of the fields unchanged. For example, Navigation Type as None)
    - Activity Name
    - Layout Name
5. To import SDK libraries to your project, follow these steps:
    a. Unzip the `Kony Android SDK.zip`.
    b. Unzip `libs.zip`.
    c. Copy libs file into your project **libs** folder.

d. Copy `kony-sdk.jar` inside the **libs** folder. Your project **libs** folder should look like:

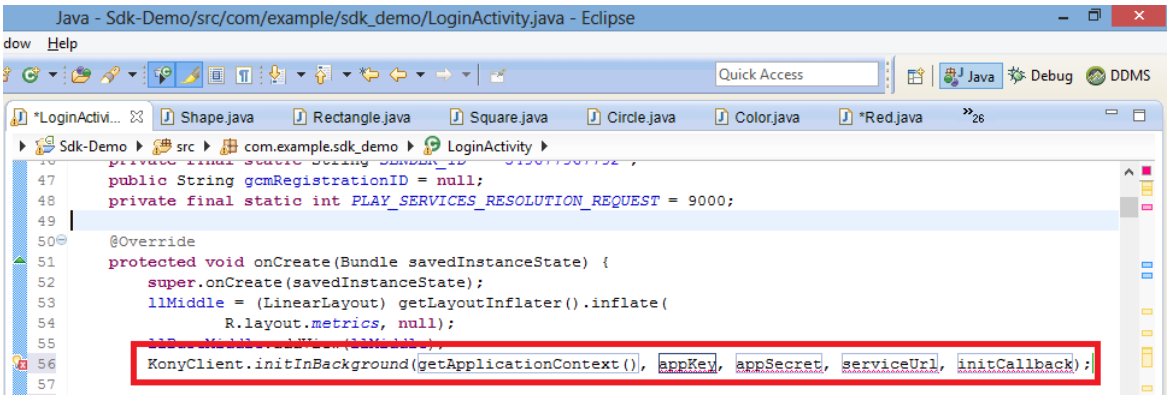


6. Add the following permissions in your `AndroidManifest.xml`:
   a. <uses-permission android:name="android.permission.INTERNET" />
   b. <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   c. <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
   d. <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

7. Under your **Sampleapp** folder, navigate to **src > com.example.sample app**, and then click **MainActivity.java**.



The `Kony-Android-SDK` is now successfully configured in your project. Now you can use Kony APIs for the Android platform.

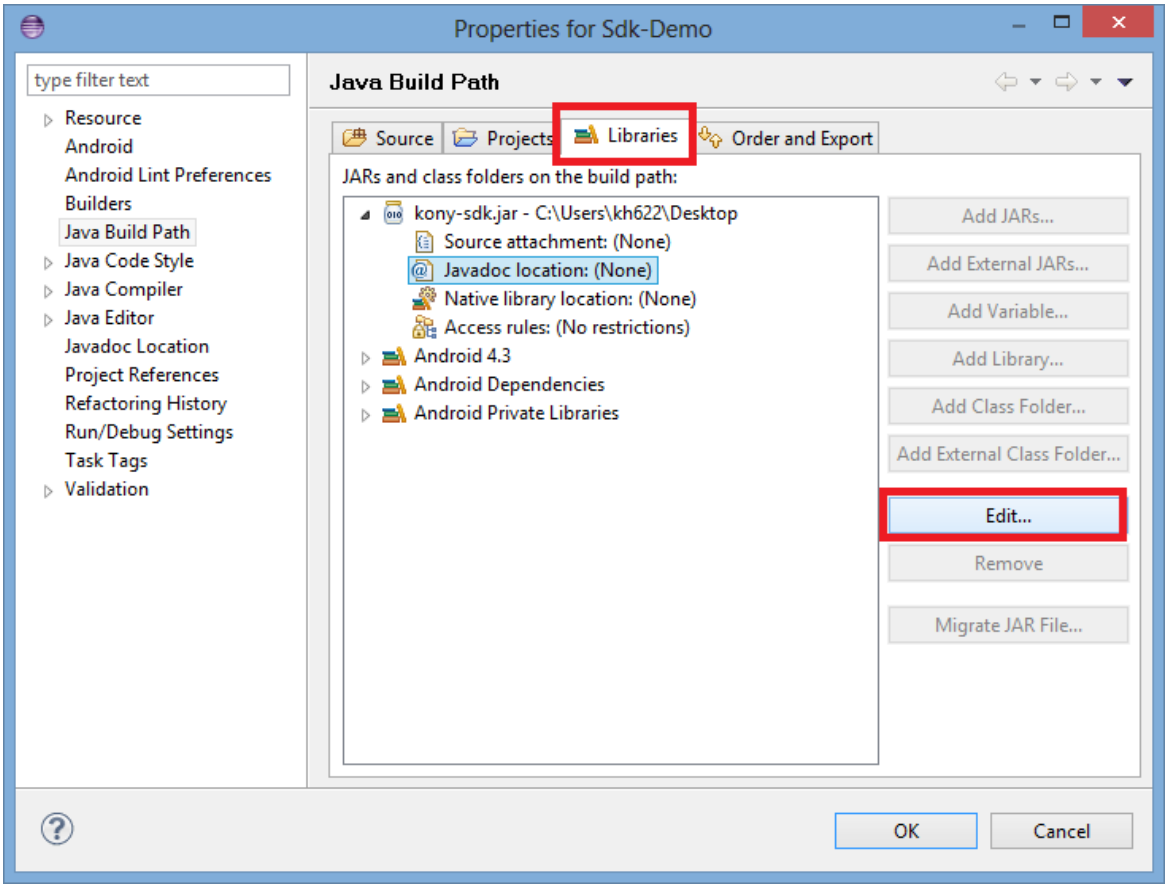


39.7.3.2 Configuring kony-sdk.doc to Project - Eclipse

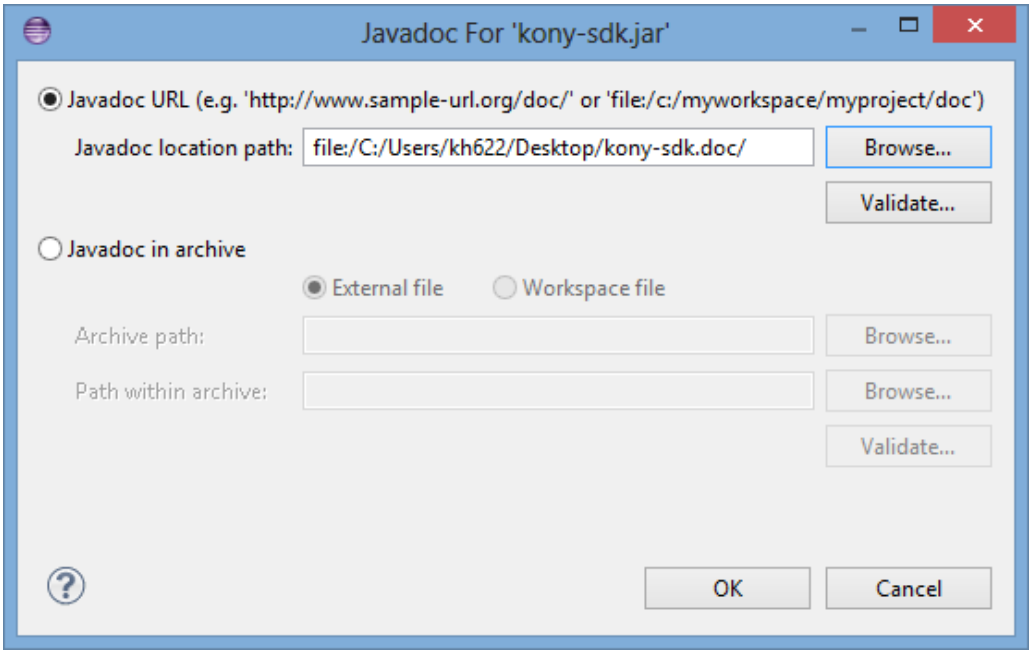
Kony provides APIs docset to search and browse API documentation within Eclipse. The docset also provides quick help in the code completion pop-up.

To configure kony-sdk.doc, follow these steps:

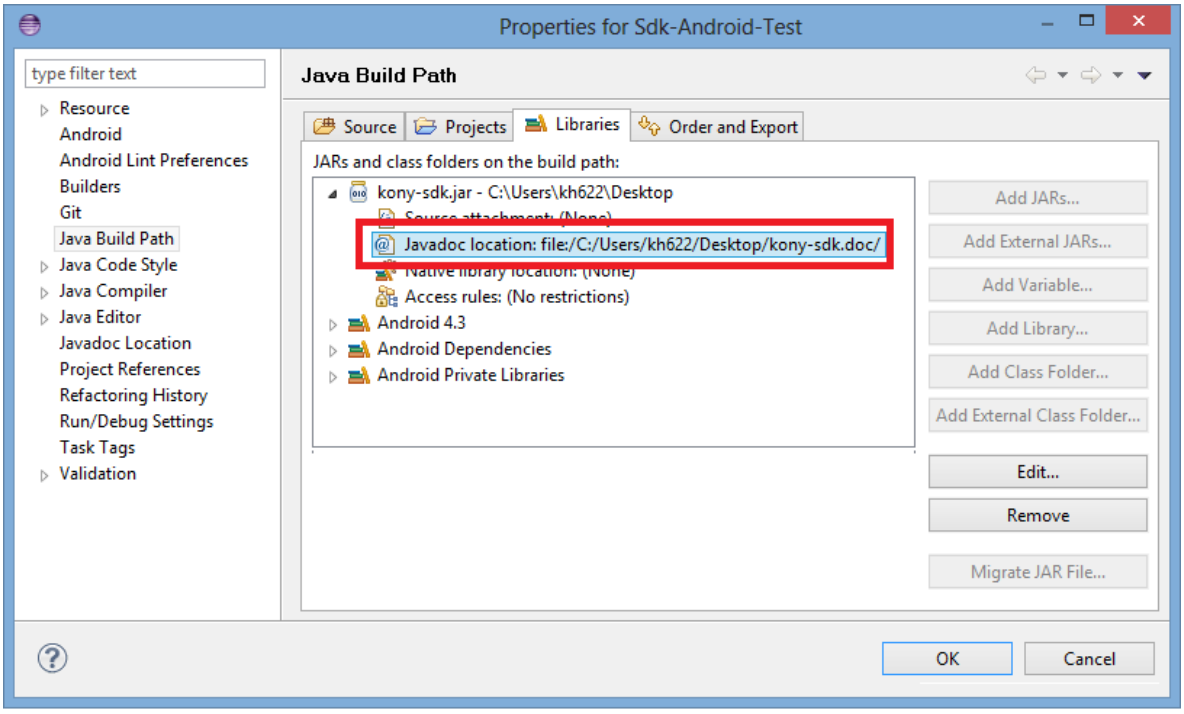
1. Under **Package Explorer**, right-click **Sampleapp**, and then click **Properties**. The **Properties** window appears.
2. Click the **Libraries** tab, and then click the **Edit** button.



3. Click **Browse** to navigate your `kony-sdk.doc` folder and then click **OK**.



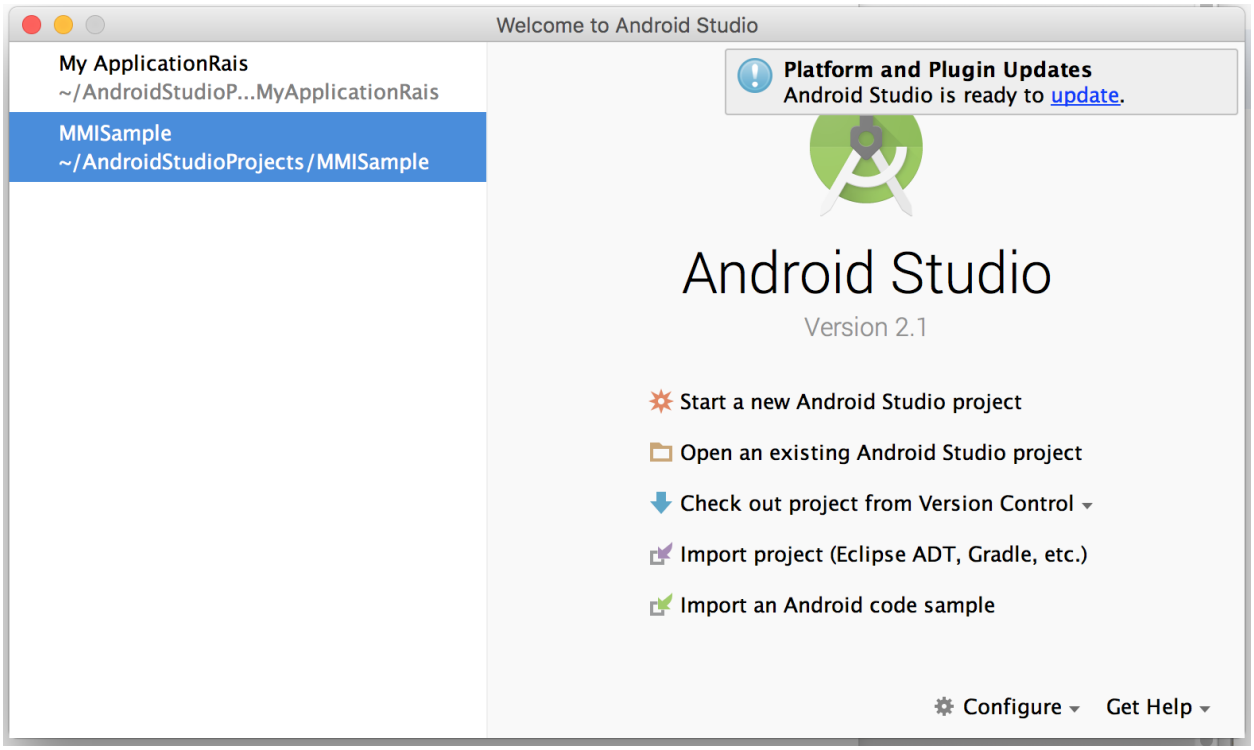
The system adds the `kony-sdk.doc` folder under the **Libraries** tab.



39.7.3.3 Configuring kony-sdk.jar to Project - Android Studio

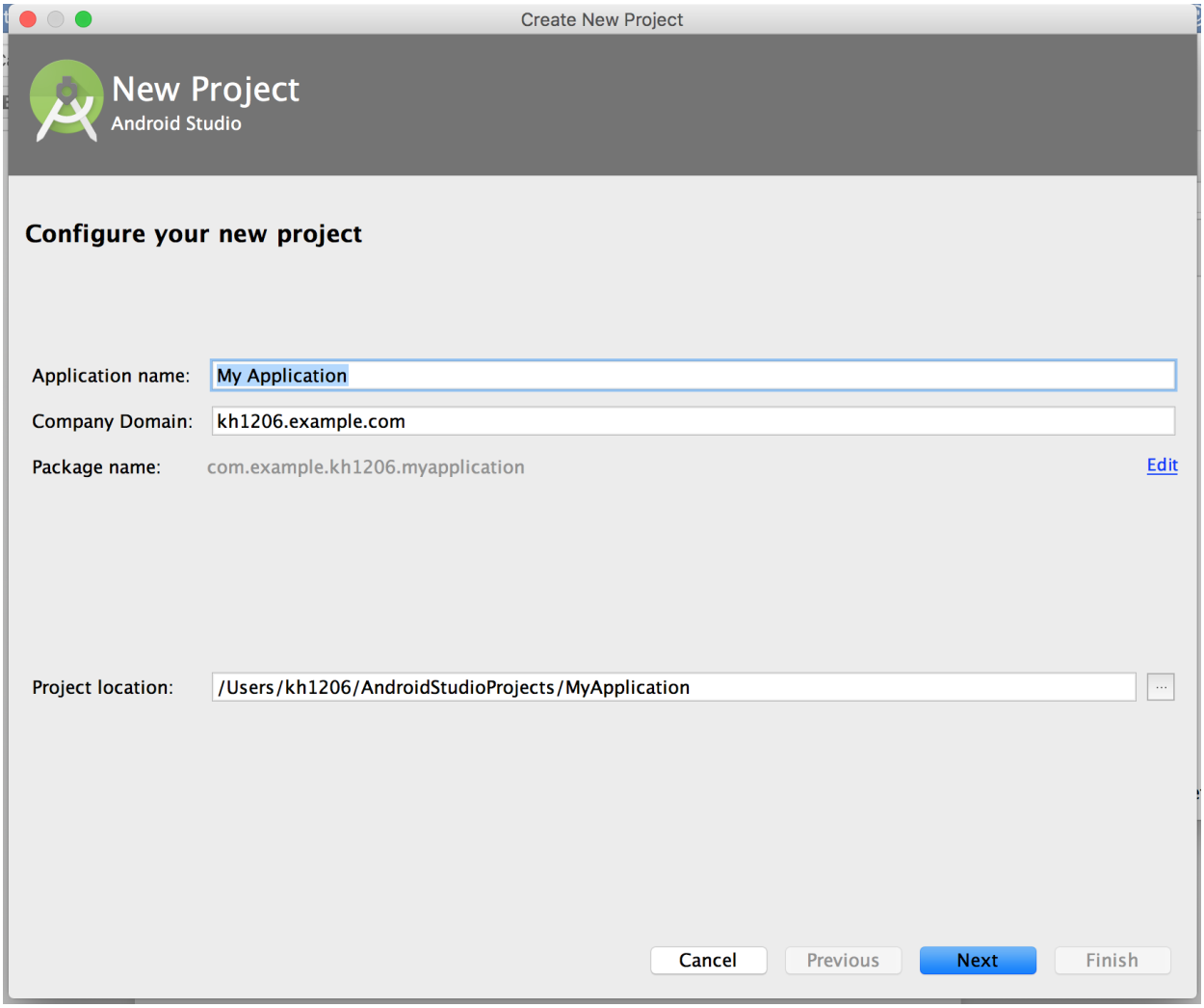
To configure kony-sdk.jar file, follow these steps:

1. Launch **Android Studio**.



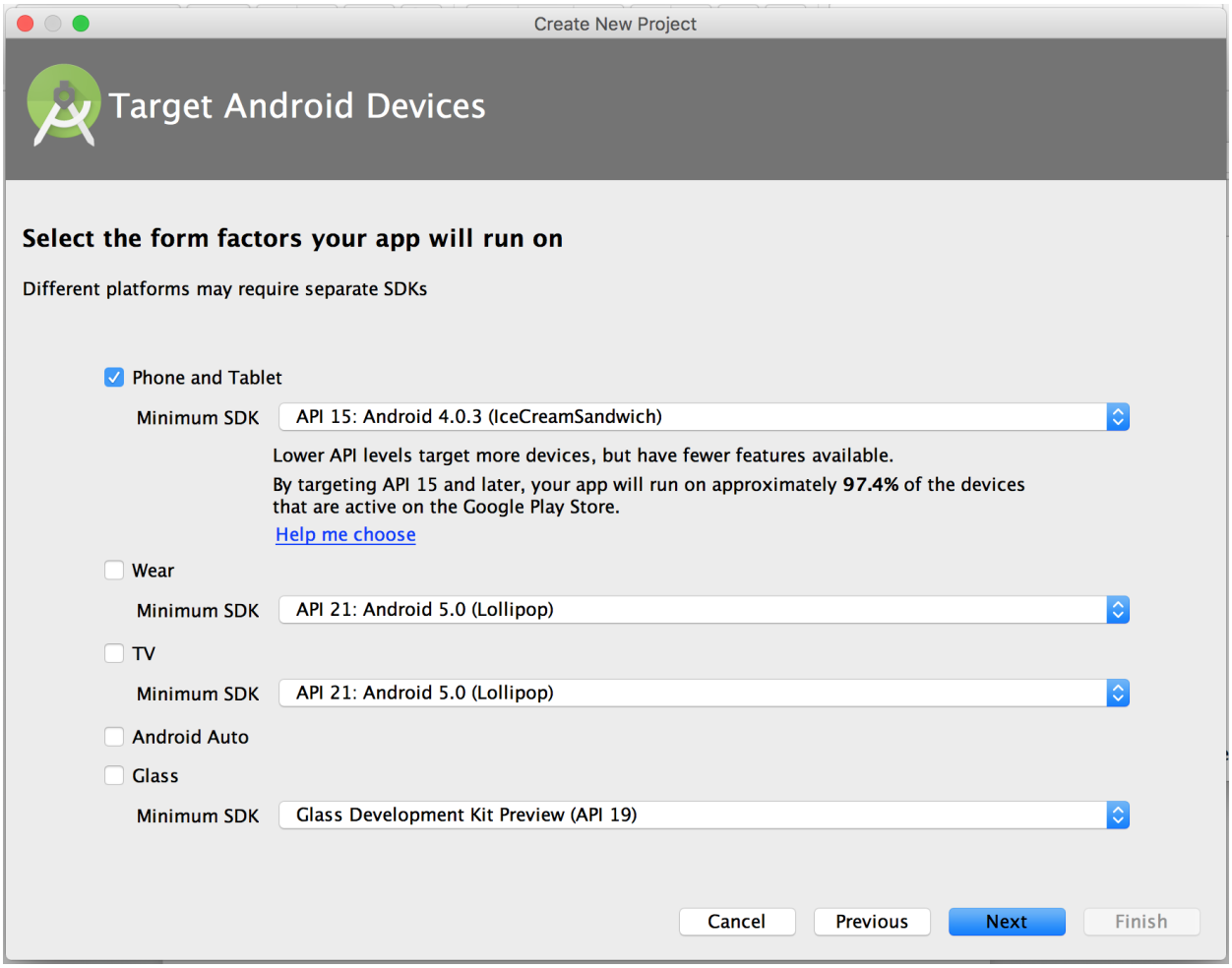
2. In Android Studio, click **Start a new Android Studio Project**.

   The **New Project** screen appears.

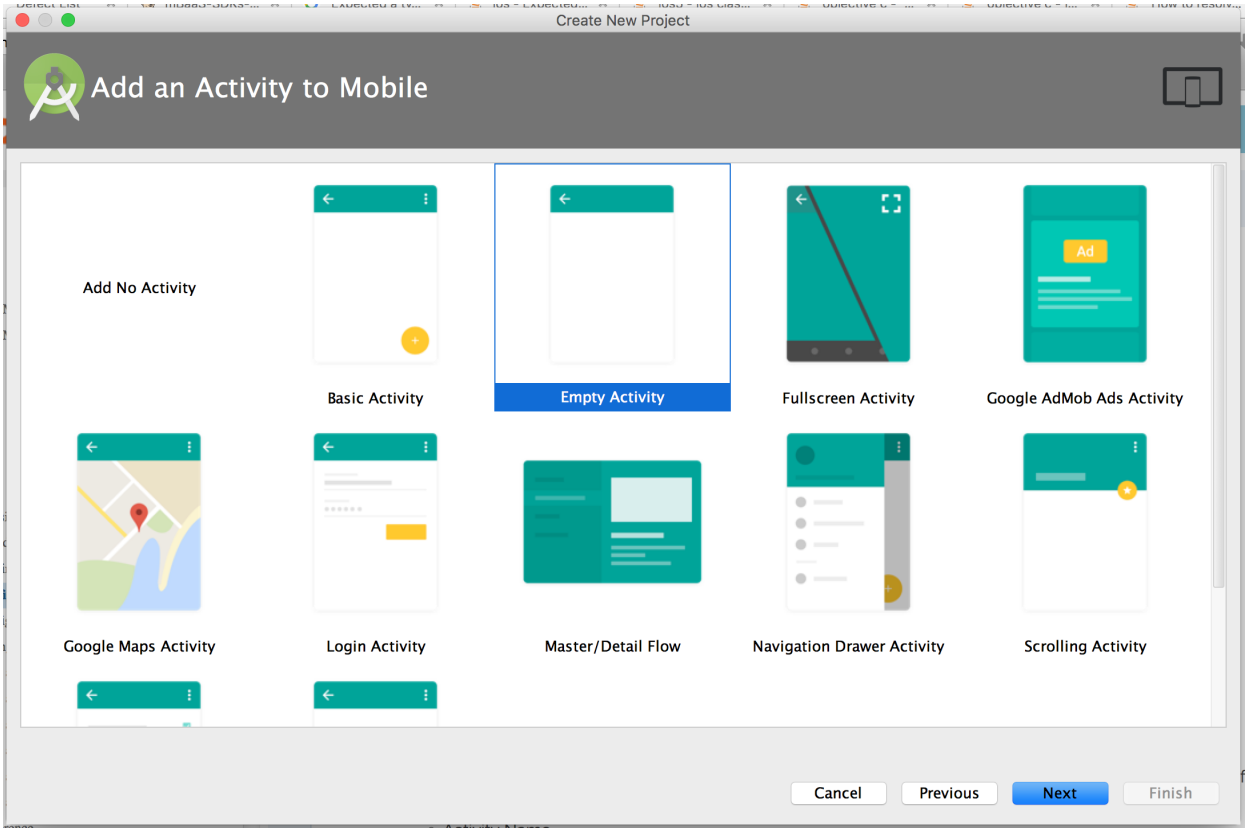


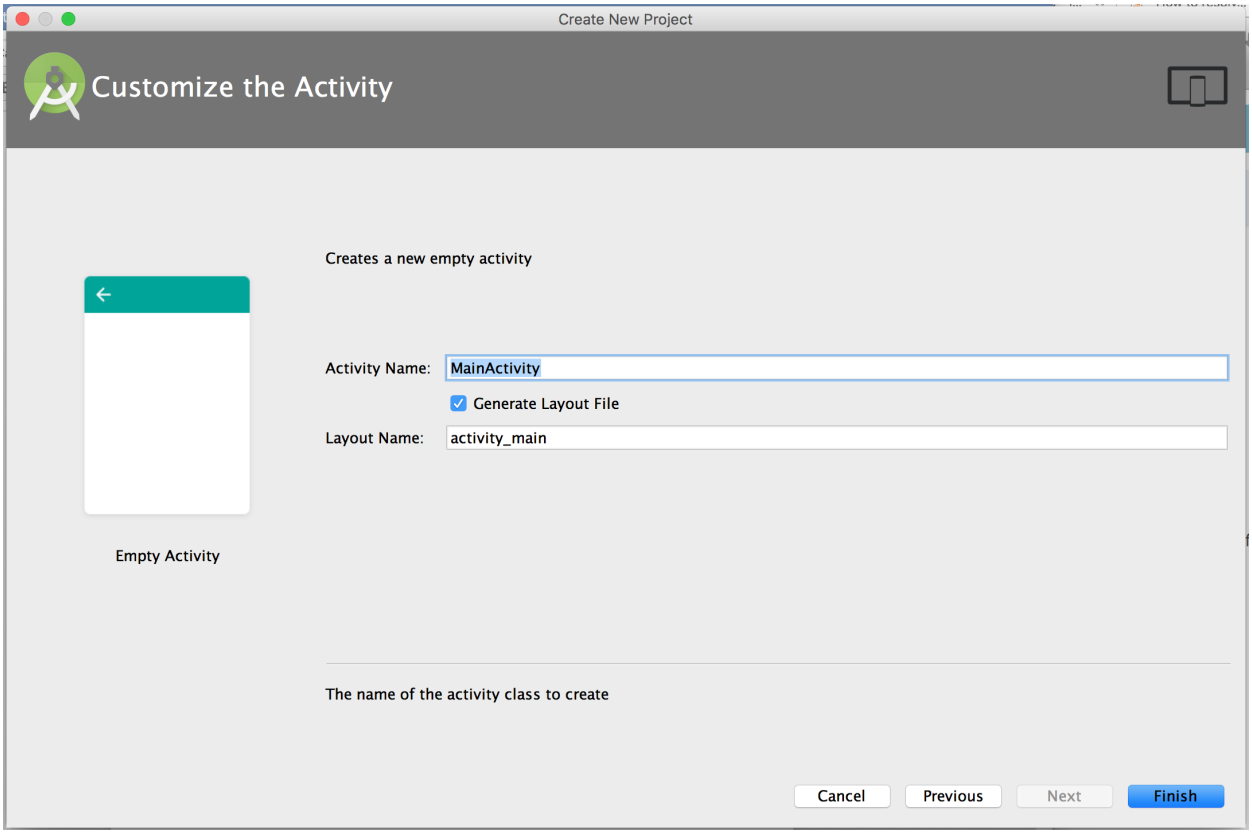
3. In the **New Android Application** dialog, enter the following and click **Next**. (You can leave the rest of the fields unchanged.)
    - **Application Name**: Enter the name of the application.
    - **Company Domain**: Enter company domain. (Android Studio generates the package name in reverse order).
    - **Package Name**: The field is auto-filled based on the company domain. You can edit the package name, if required.
    - **Project Location**: Select a path for your package.

4. Click **Next**.



5. Select the form factors and platforms, and click **Next**.
6. Add activity as follows:
    - If you select an **Activity**, the **Next** button is activated. Provide the name for the activity and click **Finish**.
    - If you select **Add No Activity**, the **Finish** button is activated. The project is created without user interface.

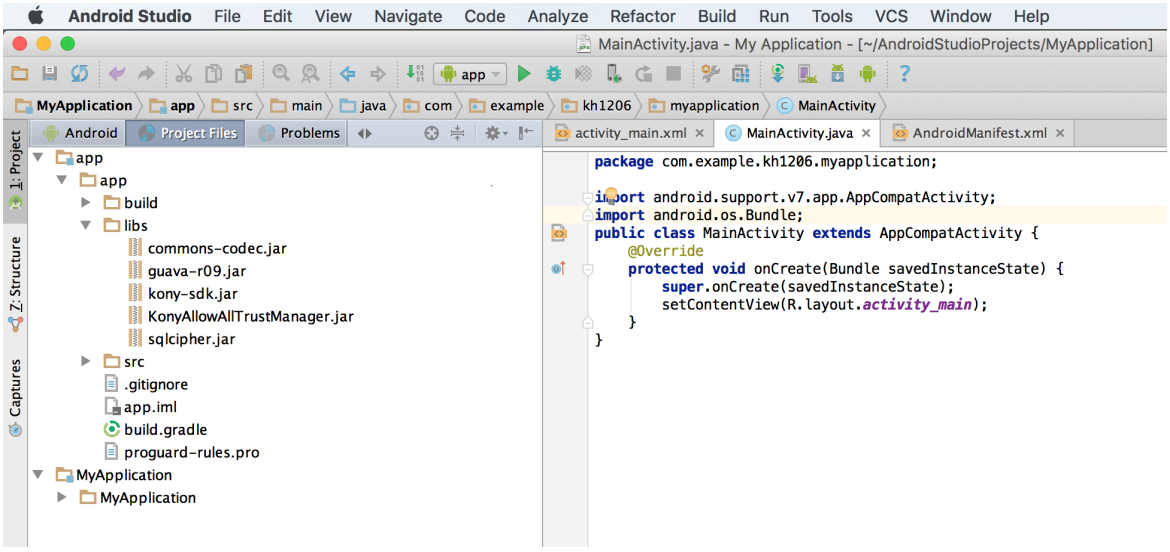




Your project is created. Now You need add the required Kony SDK libraries to the project.

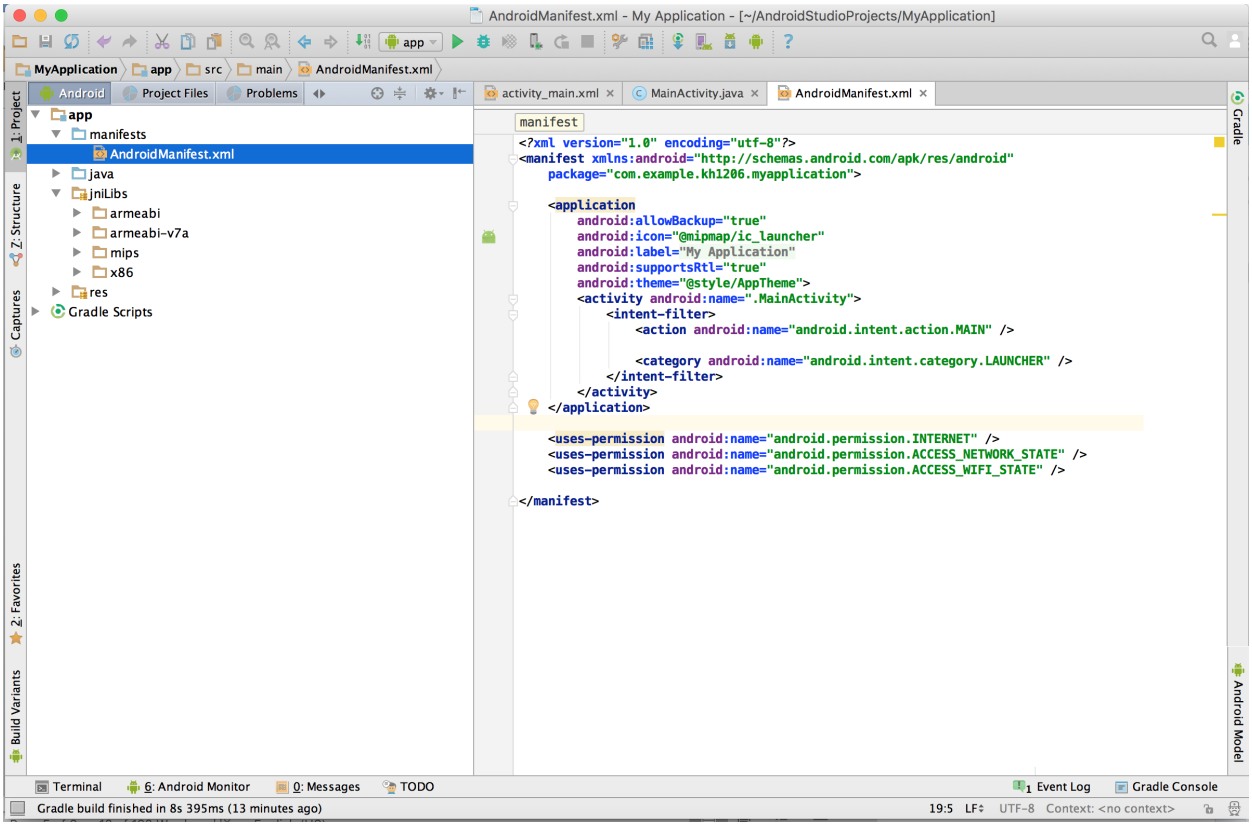
7. To import SDK libraries to your project, follow these steps:
    a. Unzip the `Kony Android SDK.zip`.
    b. Copy the `kony-sdk.jar` and paste it to `/app/libs/` folder in your project.
    c. Unzip `libs.zip` (downloaded from kony sdk.) You will get all the required SDK libraries for your project.
    d. Copy all the jar files from the SDK libraries and paste them to `/app/libs/` folder in your project.
8. Create a folder named **jniLibs** (if not present) at `/app/src/main/` location. Copy the all the

folders containing *.so files (armeabi, armeabi-v7a, mips, x86) into the **jniLibs**. Your project folder structure should look like.



9. Add the following permissions in your `AndroidManifest.xml`:
    a. <uses-permission android:name="android.permission.INTERNET" />
    b. <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    c. <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    d. <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

The `Kony-Android-SDK` is now successfully configured in your project. Now you can use Kony APIs for Android platform.



39.7.4 Initializing the Android Client SDK

Initialize Kony Fabric client with the following code, and start using the services provided in Kony Fabric. The initialization method fetches the configuration from Kony Fabric and saves it in the cache. Later, the application uses the cached configuration. It is a synchronous call.

When SDK is initialized, the Kony SDK registers a session and sends its information to the Kony Fabric Server. If the device is offline, or the server is not reachable, the session information persists on the device until it can successfully send the information to the Kony Fabric server.

For more information on application session, refer Standard Report Docs.

> ***Note:*** The sessions created by Native Fabric SDKs are interactive.

Import the following Libraries:

- com.kony.sdk.callback.InitCallback
- com.kony.sdk.client.KonyClient
- com.kony.sdk.common.KonyException

39.7.4.1 init

```
 //Sample code to initialize  ClientKonyClient myClient = new
KonyClient();
String appkey = "<your-app-key>";
String appsecret = "<your-app-secret>";
String serviceURL = "<your-service-url>";
Context context = getApplicationContext();
try {
    myclient.initAsync(getApplicationContext(), appkey, appsecret,
serviceURL new InitCallback() {
        @Override
        public void onSuccess(JSONObject response) {
            Log.d("Init", "Success");
        }
        @Override
        public void onFailure(KonyException error) {
            Log.d("Init", "Failure");
        }
    });
} catch (KonyException exception) {
    Log.d("Init", "Exception");
}
```

If you are using an untrusted self-signed (SSL) certificate with Kony Fabric installation, by default native apps do not allow untrusted SSL certificates for HTTPS connection.

To make your native apps work with untrusted SSL certificates, call the following API:

```
KonyClient.acceptSelfSignedCertificates();
```

> ***Note:*** Once a user calls the KonyClient.acceptSelfSignedCertificates(); API, a native application will accept SSL certificates throughout the app life cycle. A user cannot disable the API from a native app running on a device.

39.7.5 Invoking an Identity Service

The following are the methods you can use for an identity service.

- Login with provider type as Basic
- Login with provider type as OAuth/SAML
- Custom OAuth or Custom Login
- Get Backend Token
- User Profile
- Logout

39.7.5.1 Login with provider type as Basic

Import the following Libraries:

- com.kony.sdk.callback.LoginCallback
- com.kony.sdk.client.KonyFactory
- com.kony.sdk.client.KonyUser
- com.kony.sdk.common.IdentityServiceException
- com.kony.sdk.common.KonyException
- com.kony.sdk.services.identity.IdentityService

Sample Code

```
 // Sample code to log in using basic type provider
String providerName = "<your-provider-name>";
String username = "<username-for-your-provider>";
String password = "<password-for-your-provider>";

KonyClient myClient = new KonyClient();
IdentityService authClient = null;
try {
    authClient = myClient.getIdentityService(providerName);
} catch (KonyException exception) {
    Log.d("Failure", exception.getMessage());
}
authClient.loginInBackground(username, password, new LoginCallback() {
    @Override
    public void onSuccess(KonyUser user) {
        Log.d("Login", "Success");
    }
    @Override       public void onFailure(IdentityServiceException
identityServiceException) {
        Log.d("Login Failure",
identityServiceException.getErrorMessage());
    }
});
```

Important: When you select Kony User Repository as the identity type, the system does not allow you to provide identity name.

To use Kony User Repository as authentication service, ensure that the value for `providerName` must be set as `userstore`. If you set it with any other value (for example, Kony User Repository, User Store or Cloud Repository), the system throws an error.

39.7.5.2 Login with provider type as OAuth/SAML

Import the following Libraries:

- com.kony.sdk.callback.WebViewCallback
- com.kony.sdk.client.KonyFactory
- com.kony.sdk.client.KonyUser
- com.kony.sdk.common.IdentityServiceException
- com.kony.sdk.common.KonyException
- com.kony.sdk.services.identity.IdentityService

Sample Code

```
 // Sample code to log in using Oauth/SAML type provider
String providerName = "<your-provider-name>";
String loginOptions = {};
Context context = getApplicationContext();
Webview webview = (WebView) findViewById(R.id.webview);
KonyClient myClient = new KonyClient();
IdentityService authClient = null;
try {
    authClient = myClient.getIdentityService(providerName);
} catch (KonyException exception) {
    Log.d("Failure", exception.getMessage());
}
authClient.loginInBackground(context, webview, new WebViewCallback() {
    @Override
    public void onSuccess(KonyUser user) {
        Log.d("Login", "Success");
    }
    @Override
```

```
    public void onFailure(IdentityServiceException
identityServiceException) {
        Log.d("Login Failure",
identityServiceException.getErrorMessage());
    }
});
```

39.7.5.3 Custom OAuth / Custom Login

Sample Code

```
 //Sample code for custom Auth or Custom Login

String servname = "<your-provider-name>";
Hashtable < String, String > customTable = new Hashtable < String,
String > ();
customTable.put("userid", "<username-for-your-provider>");
customTable.put("custom1", "<custom-value-one>");
customTable.put("password", "<password-for-your-provider>");
customTable.put("custom2", "<custom-value-two>");
customTable.put("custom5", "<custom-value-five>");
customTable.put("custom3", "<custom-value-three>");
customTable.put("custom4", "<custom-value-four>");
authClient.loginInBackground(customTable, new LoginCallback() {
    @Override
    public void onFailure(IdentityServiceException
identityServiceException) {
        Log.d("Login Failure",
identityServiceException.getErrorMessage());
    }
    @Override
    public void onSuccess(KonyUser user) {
        Log.d("Login", "Success");
```

```
    }
});
```

39.7.5.4 Get Backend Token

Import the following Libraries:

com.kony.sdk.services.identity.IdentityService

Sample Code

```
// Sample code to get backend token for provider
String backendtoken = authClient.getBackendToken().toString();
```

> ***Note:*** This method returns a valid token only after login is successful.

39.7.5.5 User Profile

Import the following Libraries:

- com.kony.sdk.client.KonyUser
- com.kony.sdk.services.identity.IdentityService

Sample Code

```
// Sample code to get user profile details
KonyUser user = < object - returned - in -login - success - callback >
;
String firstName = user.getFirstName();
String lastName = user.getLastName();
String email = user.getEmail();
String userid = user.getUserId();
```

> ***Note:*** KonyUser object is returned after login is successful. The operations on the KonyUser object are given above.

39.7.5.6 Logout

Import the following Libraries:

- com.kony.sdk.common.IdentityServiceException
- com.kony.sdk.services.identity.IdentityService
- com.kony.sdk.callback.LogoutCallback

Sample Code

```
 // Sample code to logout from auth service
authClient.logoutInBackground(new LogoutCallback {
    @Override
    public void onSuccess(boolean result) {
        Log.d("Logout", "Successful");
    }
    @Override
    public void onFailure(IdentityServiceException exception) {
        Log.d("Logout failed", exception.getErrorMessage());
    }
});
```

39.7.6 Invoking an Integration Service

Import the following Libraries:

- com.kony.sdk.callback.IntegrationServiceCallback
- com.kony.sdk.client.KonyFactory
- com.kony.sdk.common.IntegrationServiceException

- com.kony.sdk.common.KonyException
- com.kony.sdk.services.integration.IntegrationService

39.7.6.1 Code

This APIs invokes integration service that is configured in the Kony Fabric portal.

```
 // Sample code to fetch the integration service details
String serviceName = < integration - service - name > ;
String operationName = < operation - name > ;
Map < String, String > params = new HashMap < String, String > ();
params.put("your-input-key", "your input value");
Map < String, String > headers = new HashMap < String, String > ();
headers.put("your-header-key", "your-header-value");
KonyClient myClient = new KonyClient();
IntegrationService integClient = null;


try {
    integClient = myClient.getIntegrationService(serviceName);
} catch (KonyException exception) {
    Log.d("Exception", exception.getMessage());
}
integClient.invokeOperationInBackground(operationName, headers,
parameters,
    new IntegrationServiceCallback() {
        @Override
        public void onSuccess(JSONObject response) {
            Log.d("Response" + response.toString());
        }

        @Override
        public void onFailure(IntegrationServiceException exception) {
            Log.d("Service Failure" + exception.getErrorMessage());
```

```
        }
    });
```

39.7.7 Invoking a Messaging Service

A developer should register with Google Cloud Messaging (GCM) for Android services to get the deviceToken that is used to register with Kony Fabric Messaging. Also a developer should fetch the **deviceId** and **userfriendlyId** to create an instance of messaging service.

The following are the methods you can use for a messaging service.

- Register
- Unregister
- Update GeoLocation
- Fetch All Messages
- Mark Message as Read
- Get Message Content

39.7.7.1 Register

Import the following Libraries:

- com.google.android.gms.common.ConnectionResult
- com.google.android.gms.common.GooglePlayServicesUtil
- com.google.android.gms.gcm.GoogleCloudMessaging
- com.kony.sdk.callback.MessagingCallback
- com.kony.sdk.common.MessagingServiceException
- com.kony.sdk.services.messaging.MessagingService

- com.kony.sdk.client.KonyFactory
- com.kony.sdk.common.KonyException

Sample Code

```
 // Sample code to register to messaging

Context context = getApplicationContext();
String SENDER_ID = < your - project - number - from - google > ;
GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(context);
String gcmRegistrationId = gcm.register(SENDER_ID);
String deviceId = Secure.getString(context.getContentResolver(),
Secure.ANDROID_ID);
String ufid = < your - ufid > ;

try {
    messagingClient = new KonyClient().getMessagingService();
} catch (KonyException exception) {
    Log.d("Exception", exception.getMessage());
}
messagingClient.registerInBackground(gcmRegistrationID, UFID,
deviceID, new MessagingCallback() {
    @Override
    public void onSuccess(boolean result) {
        Log.d("Registration", "success");
    }

    @Override
    public void onFailure(MessagingServiceException exception) {
        Log.d("Registration Failure", exception.getErrorMessage());
    }
});
```

39.7.7.2 Unregister

Import the following Libraries:

- com.kony.sdk.callback.MessagingCallback
- com.kony.sdk.common.MessagingServiceException
- com.kony.sdk.services.messaging.MessagingService

Sample Code

```
 // Sample code to unregister from messaging service
messagingClient.unregisterInBackground(new MessagingCallback() {
    @Override
    public void onSuccess(boolean result) {
        Log.d("Un Registration", "Success");
    }

    @Override
    public void onFailure(MessagingServiceException exception) {
        Log.d("Un Registration Failure" + exception.getErrorMessage
());
    }
});
```

39.7.7.3 Update GeoLocation

Import the following Libraries:

- import com.kony.sdk.callback.MessagingCallback
- import com.kony.sdk.common.MessagingServiceException
- import com.kony.sdk.services.messaging.MessagingService

Sample Code

```
 // Sample code to update the geolocation
Double latitude = < latitude - value > ;
Double longitude = < longitude - value > ;
String locationName = < location - name > ;
messagingClient.updateGeoLocationInBackground(latitude, longitude,
locationName,
    new MessagingCallback() {
        @Override
        public void onSuccess(boolean result) {
            Log.d("Location Update", "Successful");
        });
}

@Override
public void onFailure(MessagingServiceException exception) {
    Log.d("Location Update Faliure", exception.getErrorCode());
});
}
});
```

39.7.7.4 Fetch All Messages

Import the following Libraries:

- com.kony.sdk.callback.MessageContentCallback
- com.kony.sdk.common.MessagingServiceException
- com.kony.sdk.services.messaging.MessagingService

Sample Code

```
 // Sample code to get all messagesint startIndex = 0;
int pageSize = 1000;
messagingClient.fetchAllMessagesInBackground(startIndex, pageSize, new
MessageContentCallback() {
    @Override
    public void onSuccess(JSONObject response) {
        Log.d("Fetched Messages", response.toString());
    }

    @Override
    public void onFailure(MessagingServiceException exception) {
        Log.d("Failed to fetch Messages", exception.getMessage());
    }
});
```

> ***Note:*** The Fetch all messages method returns only part of a message. To get full content of the message, use the Get Message Content method.

39.7.7.5 Mark Message as Read

Import the following Libraries:

- com.kony.sdk.callback.MessageContentCallback
- com.kony.sdk.common.MessagingServiceException
- com.kony.sdk.services.messaging.MessagingService

Sample Code

```
 // Sample code to mark messages as read
String messageid = < fetch - id - of - the - message > ;
messagingClient.markMessageReadInBackground(messageid, new
```

```
MessageContentCallback() {
    @Override
    public void onSuccess(boolean result) {
        Log.d("Message", "marked");
    }

    @Override
    public void onFailure(MessagingServiceException exception) {
        Log.d("Failed to mark message as read",
exception.getErrorMessage());
    }
});
```

39.7.7.6 Get Message Content

Import the following Libraries:

- com.kony.sdk.callback.MessageContentCallback
- com.kony.sdk.common.MessagingServiceException
- com.kony.sdk.services.messaging.MessagingService

Sample Code

```
 // Sample code to fetch message content from  Messaging
String messageid = < fetch - id - of - the - message > ;
messagingClient.fetchMessageContentInBackground(messageid, new
MessageContentCallback() {
    @Override
    public void onSuccess(JSONObject response) {
        Log.d("Message Content", response.toString());
    }

    @Override
```

```
    public void onFailure(MessagingServiceException exception) {
        Log.d("Failed to fetch Message Content",
exception.getErrorMessage());
    }
});
```

39.7.8 Invoking a Sync Service

- Getting Sync Instance
- Creating a Sync Object
- Error Codes
- Create, Read, Update, and Delete (CRUD) operations in Native SDK
- Updating a Sync Object
- Retrieving an Object
- Deleting an Object
- Pushing (or syncing) Changes to the Sync Server

39.7.8.1 Getting Sync Instance

To get sync service instance pass context of the activity.

```
 //getting and initialising sync service
Sync s = null;
try {
    s = new KonyClient().getSyncService();
    //init success
} catch (KonyException e) {
    //init failed
}
```

39.7.8.2 Creating a Sync Object

1. Create a Sync Object

   You need to have the following files to use Sync SDKs:

   Once you upload the sync config file, the system generates code. For Android, the generated code will contain the following two packages:

   - `com.kony.sdk.services.sync.codegen`: This contains model class for each object in sync config and named as <yourobject>.java
   - `com.kony.sdk.services.sync.metadata`: This contains single class containing ddls needed to create device database and named as SyncGeneratedMetadata.java

2. Import above packages into your project.

39.7.8.3 Error Codes

The following is a list of error codes for Android platform along with the corresponding causes and error messages:

Error Code	Cause	Error Message	Comments
KNY1000E		Unknown Error	
KNY1001E		Please initialize init before calling any other API	
KNY1002E		Out of memory	cplusplus > bad_alloc
KNY1003E		Input Output Exception	cplusplus > ios_base

Error Code	Cause	Error Message	Comments
KNY1004E		Error occurred while creating database path:<dbPath>	
KNY1005E		Null Pointer Exception	
KNY1006E		Error occurred while parsing metadata	
KNY1007E		Row Doesn't Exist	
KNY1008E		Session in progress	
KNY1009E		Cannot delete as child record(s) with cascades false exist for this record	
KNY8888E		whatever message comes from server	
KNY1010E		UpgradeRequired In Progress	
1000	Unknown error occurred at network layer	Unknown Error occurred	
1005	Invalid input url. Please check the url	Invalid input url. Please check the url	
1012	Network Request Failed	Request Failed	
1011		Device has no data connectivity. Please try the operation after establishing connectivity.	

Error Code	Cause	Error Message	Comments
KNY2005E		Invalid Arguments for the API (Messages Differ as per the API Calls) For example, eventType can not be null. Provide a proper eventType eventSubType value is greater than 256 characters. Provide a Event Subtype with at most 256 chars etc.	
KNY2000E	1. Not able to add header or param to service 2. Not able to start a thread for making network call	<Generic exception, error message will be thrown by android layer>	
KNY2001E	fetchDetailsIn Background	http://docs.oracle.com/ javase/7/docs/ api/java/lang/ InstantiationException.html	
KNY2002E	1.Fetch Details In Background 2. initializing sync service	http://docs.oracle.com/javase/ 7/docs/api/java/lang/ IllegalAccessException.html	
KNY2003E	initializing sync service	http://docs.oracle.com/javase/7 /docs/api/java/lang/ NoSuchFieldException.html	

Error Code	Cause	Error Message	Comments
KNY2004E	initializing sync service	http://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html	
KNY2009E	initializing sync service	http://docs.oracle.com/javase/7/docs/api/java/lang/IllegalAccessException.html	
KNY2006E	initializing sync service	http://docs.oracle.com/javase/7/docs/api/java/lang/ClassNotFoundException.html	
KNY2010E	Security Exception in networkcalls		
KNY2011E	IOException in networkcalls		
KNY2012E	Connection Time out error		
KNY2013E	No HttpResponseException in network calls		
KNY2007E	When no object is found in fetchDetails()		
KNY2008E	When encounters an improper JSON		

39.7.8.4 Create, Read, Update, and Delete (CRUD) operations in Native SDK

Create/Update

For both Create and Update, we have the same APIs. If object passed is a new object then create happens, otherwise if object is fetched from device update happens.

Creating a Record

API Used: public <T extends SyncObject> void save(T syncObject) throws KonyException

Sample Code

```
 try {
    Categories cat = new Categories();
    cat.setCategoryName("Fruits");
    cat.setDescription("Apple");
    SyncClient.getSyncDataStoreInstance().save(cat);
    //save success
} catch (KonyException e) {
    //save failed
}
```

Creating a Record in Background

API Used: public <T extends SyncObject> void saveInBackground(T dataObject, SyncObjectCallback<T> callback)

Sample Code:

```
 Categories c = new Categories();
c.setCategoryName("Fruit");
c.setDescription("Apple");

SyncClient.getSyncDataStoreInstance().saveInBackground(c, new
SyncObjectCallback < Categories > () {
    @Override
    public void onFailure(KonyException e) {
        //save failed
    }

    @Override
```

```
    public void onSuccess(Categories arg0) {
        //save success

    }
});
```

Bulk Create

```
try {
    List < Categories > catList = new ArrayList < Categories > ();

    //create first category
    Categories cat = new Categories();
    cat.setCategoryName("Fruits");
    cat.setDescription("Apple");

    //create 2nd category
    Categories cat1 = new Categories();
    cat1.setCategoryName("Fruits1");
    cat1.setDescription("Apple1");

    //create 3rd category
    Categories cat2 = new Categories();
    cat2.setCategoryName("Fruits2");
    cat2.setDescription("Apple2");

    catList.add(cat);
    catList.add(cat);
    catList.add(cat2);

    SyncFactory.getSyncDataStoreInstance().bulkSave(catList);
    //bulk save success
} catch (KonyException e) {
```

```
    //bulk save failed
}
```

Bulk Create in Background

```
 List < Categories > catList = new ArrayList < Categories > ();

//create first category
Categories cat = new Categories();
cat.setCategoryName("Fruits");
cat.setDescription("Apple");

//create 2nd category
Categories cat1 = new Categories();
cat1.setCategoryName("Fruits1");
cat1.setDescription("Apple1");

//create 3rd category
Categories cat2 = new Categories();
cat2.setCategoryName("Fruits2");
cat2.setDescription("Apple2");

catList.add(cat);
catList.add(cat);
catList.add(cat2);

SyncClient.getSyncDataStoreInstance().bulkSaveInBackground(catList,
    new SyncObjectListCallback < Categories > () {

        @Override
        public void onFailure(KonyException arg0) {
            // TODO Auto-generated method stub

        }
```

```
        @Override
        public void onSuccess(List < Categories > arg0) {
            // TODO Auto-generated method stub

        }
    });
```

Bulk Update

```
 //bulkupdate
try {
    //First get records from db
    Query < Categories > query = SyncClient.getSyncDataStoreInstance
().createQuery(Categories.class);
    query.addWhereClause("CategoryName='Fruits'");
    List < Categories > categoriesList =
SyncClient.getSyncDataStoreInstance().executeQuery(query);

    //update categories list
    for (Categories cat: categoriesList) {
        cat.setCategoryName("Fruits Updated");
    }
    SyncClient.getSyncDataStoreInstance().bulkSave(categoriesList);
    //save success
} catch (KonyException e) {
    //save failed
}
```

Bulk Update in Background

```
 //bulkupdateinbackground
//First get records from db
Query < Categories > query = SyncClient.getSyncDataStoreInstance
```

```
().createQuery(Categories.class);
query.addWhereClause("CategoryName='Fruits'");
SyncClient.getSyncDataStoreInstance().executeQueryInBackground(query,
    new SyncObjectListCallback & lt; Categories & gt;
    () {

        @Override
        public void onFailure(KonyException arg0) {
            //get failure

        }

        @Override
        public void onSuccess(List < Categories > arg0) {
            //update categories list
            for (Categories cat: arg0) {
                cat.setCategoryName("Fruits Updated");
            }
            SyncClient.getSyncDataStoreInstance().bulkSaveInBackground
(arg0,
                new SyncObjectListCallback < Categories > () {

                    @Override
                    public void onFailure(KonyException arg0) {
                        // update failure

                    }

                    @Override
                    public void onSuccess(List < Categories > arg0) {
                        //update success

                    }
```

```
                });
        }
    });
```

39.7.8.5 Updating a Sync Object

Updating a Record

```
 try {
    //First get record from db
    PrimaryKey pk = SyncClient.getSyncDataStoreInstance
().createPrimaryKeyInstance();
    pk.setAttribute("CategoryID", 1);
    Categories cat = SyncClient.getSyncDataStoreInstance().getObject
(Categories.class, pk);

    //update the record
    cat.setCategoryName("Fruits Updated");
    cat.setDescription("Apple Updated");
    SyncClient.getSyncDataStoreInstance().save(cat);
    //save success
} catch (KonyException e) {
    //save failed
}
```

Updating a Record in Background

```
 //First get record from db
PrimaryKey pk = SyncClient.getSyncDataStoreInstance
().createPrimaryKeyInstance();
pk.setAttribute("CategoryID", 1);
SyncClient.getSyncDataStoreInstance().getObjectInBackground
(Categories.class, pk,
    new SyncObjectCallback & lt; Categories & gt;
    () {
```

```
        @Override
        public void onFailure(KonyException arg0) {
            //get record failed

        }

        @Override
        public void onSuccess(Categories c) {
            //update the record
            c.setCategoryName("Fruits Updated");
            c.setDescription("Apple Updated");
            SyncClient.getSyncDataStoreInstance().saveInBackground(c,
new SyncObjectCallback & lt; Categories & gt;
                () {
                    @Override
                    public void onFailure(KonyException e) {
                        //save failed
                    }

                    @Override
                    public void onSuccess(Categories arg0) {
                        //save success

                    }
                });

        }

    });
```

39.7.8.6 Retrieving an Object

Executing Queries

Query class can be used to define queries.

Execute Query

```
 SyncDataStore dataStore = SyncClient.getSyncDataStoreInstance();
Query < Categories > query = dataStore.createQuery(Categories.class);
query.addSelectColumn("CategoryId");
query.addWhereClause("
               try {
                       List<Categories> categoriesList = dataStore.executeQuery(q
                       //query success
               } catch (KonyException e) {
                       //query failed
               }
```

Execute Query in Background

```
 SyncDataStore dataStore = SyncClient.getSyncDataStoreInstance();
Query < Categories > query = dataStore.createQuery(Categories.class);
query.addSelectColumn("CategoryId");
query.addWhereClause("CategoryName='Fruits'");
dataStore.executeQueryInBackground(query, new SyncObjectListCallback <
Categories > () {

    @Override
    public void onFailure(KonyException arg0) {
        //query failed

    }

    @Override
    public void onSuccess(List<Categories> arg0) {
```

```
        //query success

    }
});
```

Getting an Object

To retrieve an object from database using its primary key.

getObject

```
 //getobject
try {
    //define pk
    PrimaryKey pk = SyncClient.getSyncDataStoreInstance
().createPrimaryKeyInstance();
    pk.setAttribute("CategoryID", 1);
    Categories cat = SyncClient.getSyncDataStoreInstance().getObject
(Categories.class, pk);
    //get success
} catch (KonyException e) {
    //get failed
}
```

getObject in background

```
 //getobject in background
//define pk
PrimaryKey pk = SyncClient.getSyncDataStoreInstance
().createPrimaryKeyInstance();
pk.setAttribute("CategoryID", 1);
SyncFactory.getSyncDataStoreInstance().getObjectInBackground
(Categories.class, pk,
    new SyncObjectCallback < Categories > () {

        @Override
```

```
        public void onFailure(KonyException arg0) {
            //get record failed
        }

        @Override
        public void onSuccess(Categories c) {
            //get success
        }

    });
```

Fetching an Object

To fetch all the details of a partially fetched object.

Fetching an Object

```
 //fetchDetails try{ //Get partial object - Only CategoryIds from
Category table
SyncDataStore dataStore = SyncClient.getSyncDataStoreInstance();
Query < Categories > query = dataStore.createQuery(Categories.class);
query.addSelectColumn("CategoryId");
query.addWhereClause("CategoryName='Fruits'");
List < Categories > categoriesList = dataStore.executeQuery(query);
//now fill all the partial objects
for (Categories c: categoriesList) {
    dataStore.fetchDetails(C);
}
//fetch success
} catch (KonyException e) {
    //fetch failed
}
```

Fetching an Object in Background

```
 //fetchDetails in background
try {
    //Get partial object - Only CategoryIds from Category table
    SyncDataStore dataStore = SyncClient.getSyncDataStoreInstance();
    Query < Categories > query = dataStore.createQuery
(Categories.class);
    query.addSelectColumn("CategoryId");
    query.addWhereClause("CategoryName='Fruits'");
    List < Categories > categoriesList = dataStore.executeQuery
(query);

    //fill first partial object
    Categories c = categoriesList.get(0);
    dataStore.fetchDetailsInBackground(c, new SyncObjectCallback <
Categories > () {
        @Override
        public void onFailure(KonyException arg0) {
            //fetch success
        }

        @Override
        public void onSuccess(Categories arg0) {
            //fetch failed

        }
    });

    //fetch success
} catch (KonyException e) {
    //fetch failed
}
```

39.7.8.7 Deleting an Object

Delete

```
 try {
    //First get record from db
    PrimaryKey pk = SyncClient.getSyncDataStoreInstance
().createPrimaryKeyInstance();
    pk.setAttribute("CategoryID", 1);
    Categories cat = SyncClient.getSyncDataStoreInstance().getObject
(Categories.class, pk);
    SyncClient.getSyncDataStoreInstance().delete(cat);
    //delete success
} catch (KonyException e) {
    //delete failed
}
```

Delete in Background

```
 //Define PK
PrimaryKey pk = SyncClient.getSyncDataStoreInstance
().createPrimaryKeyInstance();
pk.setAttribute("CategoryID", 1);
SyncClient.getSyncDataStoreInstance().deleteInBackground
(Categories.class, pk, new SyncCallback() {

    @
    Override
    public void onFailure(KonyException arg0) {
        //delete failed
    }

    @
    Override
```

```
    public void onSuccess() {
        //delete success
    }

});
```

Bulk Delete

```
 //bulkdelete
try {
    //First get records from db
    Query < Categories > query = SyncClient.getSyncDataStoreInstance
().createQuery(Categories.class);
    query.addWhereClause("CategoryName='Fruits'");
    List < Categories > categoriesList =
SyncClient.getSyncDataStoreInstance().executeQuery(query);

    SyncClient.getSyncDataStoreInstance().bulkDelete(categoriesList);
    //delete success
} catch (KonyException e) {
    //delete failed
}
```

Bulk Delete in Background

```
 //bulkdeleteinbackground
//First get records from db
Query < Categories > query = SyncFactory.getSyncDataStoreInstance
().createQuery(Categories.class);
query.addWhereClause("CategoryName='Fruits'");
SyncFactory.getSyncDataStoreInstance().executeQueryInBackground(query,
new SyncObjectListCallback < Categories > () {

    @Override
```

```
    public void onFailure(KonyException arg0) {
        //get failure
    }

    @Override
    public void onSuccess(List < Categories > arg0) {
        SyncClient.getSyncDataStoreInstance().bulkDeleteInBackground
(arg0, new SyncCallback() {

            @Override
            public void onFailure(KonyException arg0) {
                // delete failure
            }

            @Override
            public void onSuccess() {
                //delete success
            }
        });
    }
});
```

39.7.8.8 Pushing (or syncing) Changes to the Sync Server

The startSessionInBackgroundAPI can be used to sync (upload and download/push and pull) changes between device and sync server. This is purely asynchronous API. To get notifications during the API execution you can implement SyncListener interface and pass object of implementing class to synchronize.

```
 Sync s;
try {
    //get sync service
    s = new KonyClient().getSyncService();
```

```
    // create instance of sync listener. ListenerImpl which implements
SyncListener
    SyncListener listener = new ListenerImpl(this);

    //add sync listener
    s.addListener(listener);

    //start sync session
    s.startSessionInBackground();
} catch (KonyException e) {
    e.printStackTrace();
}
```

Configuring Various Sync Options

You can use sync options class to configure various sync options like:

- **Upload Error Policy**: This policy can be used to configure whether to continue or abort on getting upload error. Default value is continue on error.
- **Schema Upgrade Policy**: This policy is used to configure what action to be taken when schema upgrade is available. Default value is upload and upgrade.
- **Enabling or disabling download/upload for a scope**: Default value is continue on error.
- **Adding filter for syncing**: This can be used to set download filters. By default, all pass filters are used.
- **Configuring remove after upload policy**: this can be used to configure removal of records instance from device after successful upload. By default, this option is disabled for all scopes.

Configuring NetworkOptions

The NetworkOptions class can be used to configure various options for APIs that use the network.

For example, startSessionInBackground, performUpgradeInBackground, and isUpgradeRequiredInBackground.

Following options can be configured:

1. **Network Timeout**: the time for which device should wait for server to respond. If server does not respond in the specified time, an error is thrown. Default value is ten seconds.
2. **Retry Count**: Retry count specifies how many times a request can be sent to server in case of a server error. Default value is five.
3. **Retry Wait Time**: Time between two retries. Default value is five seconds.
4. **Retry Listener**: The listener which should be invoked on each retry.

39.7.9 Invoking an Object Service

Kony provides programmatic access to the backend data through Online Object Services. You must perform the following steps to gain the access:

1. Acquire a current instance of your object service.
2. Use the object service instance with data transfer objects to communicate as needed with your backend.

To acquire a current instance of your object service, refer to getObjectService Method documentation.

To know more about the methods that act on the Kony Fabric endpoint directly, refer to OnlineObjectService Class documentation.

39.7.9.1 getObjectService Method

The **getObjectService Method** gets the current instance of the object service. The getObjectService method is invoked on the SDK instance; **init** must run successfully before invoking this method.

Syntax

```
getObjectService(String serviceName)
```

Return Type

ObjectService Instance

Parameters

Input Parameter	Type	Description	Required
servicename	String	Name of the object service	Yes

Code

```
 KonyClient client = new KonyClient();
client.initAsync(getApplicationContext(), < appkey > , < appSecret > ,
< serviceUrl > , new InitCallback() {
    @Override
    public void onSuccess(JSONObject serviceDoc) {
        kony.print("Init Success");
        ObjectService objSVC = client.getObjectService( < servicename
> );
    }

    @Override
    public void onFailure(KonyException konyException) {
        kony.print("Init failed");
    }
});
```

39.7.9.2 OnlineObjectService Class

OnlineObjectService Class provides methods that perform operations acting on the Kony Fabric endpoint, including basic CRUD. An instance of OnlineObjectService is returned by the getObjectService Method.

Methods

The following methods are used by the OnlineObjectService class and its instantiations.

Create

Creates an object in the Kony Fabric endpoint.

39.7.9.3 Syntax

```
 create(HashMap<String,String> headers,HashMap<String,String>
queryParams,ObjectServiceCallback callback)
```

39.7.9.4 Return Type

None

39.7.9.5 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	HashMap<String,String>	Key/value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	HashMap<String,String>	Key/ value pairs of query parameters that are appended to the URL while making a network call	Optional

Input Parameter	Type	Description	Required
objectServiceCallback	ObjectServiceCallback	Interface with onSuccess and onFailure methods • **onSuccess**: Function invoked when the operation succeeds with the primary key of the created object. • **onFailure**: Function invoked when the operation fails with cause of failure.	Yes

39.7.9.6 Code

```
 ObjectService objSvc = konyClient.getObjectService
("<objectservicename>");
DataObject dataObject = new DataObject("product");
JSONObject bodyParams = new JSONObject();
bodyParams.put("name", "tv");
dataObject.setRecord(bodyParams);

try {
    objSvc.create(dataObject, new HashMap < String, String > (), new
HashMap < String, String > (),
        new ObjectServiceCallback() {
            @Override
            public void onSuccess(JSONObject response) throws
KonyException {
                kony.print("create success : " + response.toString());
            }
```

```
            @Override
            public void onFailure(ObjectServiceException
objectServiceException) {
                kony.print("create failed : " +
objectServiceException.getErrorMessage());
            }
        }
    );
} catch (Exception e) {
    kony.print("Exception while creating an object  : " +
e.printStackTrace());

}
```

Update

Updates an object in the Kony Fabric endpoint.

39.7.9.7 Syntax

```
 update(DataObject dataObject,HashMap<String,String>
headers,HashMap<String,String> queryParams,ObjectServiceCallback
callback)
```

39.7.9.8 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	HashMap<String,String>	Key/value pairs of httpHeaders that are sent along with the network call.	Optional

Input Parameter	Type	Description	Required
queryParams	HashMap<String,String>	Key/ value pairs of query parameters that are appended to the URL while making a network call	Optional
objectServiceCallback	ObjectServiceCallback	Interface with onSuccess and onFailure methods • **onSuccess**: Function invoked when the operation succeeds with the primary key of the created object. • **onFailure**: Function invoked when the operation fails with cause of failure.	Yes

39.7.9.9 Code

```
 ObjectService objSvc = konyClient.getObjectService
("<objectservicename>");
DataObject dataObject = new DataObject("product");
JSONObject params = new JSONObject();
params.put("name", ”playstation”);
dataObject.setRecord(params);
try {
    objSvc.update(dataObject, new HashMap < String, String > (), new
HashMap < String, String > (), new ObjectServiceCallback() {
        @verride
        public void onSuccess(final JSONObject object) {
            kony.print("update success : " + response.toString());
        }
```

```
        @Override
        public void onFailure(final ObjectServiceException
objectServiceException) {
            kony.print("update failed : " +
objectServiceException.getErrorMessage());
        }
    });
} catch (Exception e) {
    kony.print("Exception occurred while updating an object : " +
e.printStackTrace());
}
```

Delete

Deletes an object in the Kony Fabric endpoint.

39.7.9.10 Syntax

```
 delete(DataObject dataObject,HashMap<String,String>
headers,HashMap<String,String> queryParams,ObjectServiceCallback
callback)
```

39.7.9.11 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	HashMap<String,String>	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional

Input Parameter	Type	Description	Required
queryParams	HashMap<String,String>	Key/ value pairs of query parameters that are appended to the URL while making a network call	Optional
objectServiceCallback	ObjectServiceCallback	Interface with onSuccess and onFailure methods • **onSuccess**: Function invoked when the operation succeeds with the primary key of the created object. • **onFailure**: Function invoked when the operation fails with cause of failure.	Yes

39.7.9.12 Code

```
 DataObject dataObject = new DataObject("product");
JSONObject jobject = new JSONObject();
jobject.put("id", 1);
dataObject.setRecord(jobject);
try {
    objSvc.delete(dataObject, new HashMap < String, String > (), new
HashMap < String, String > (), new ObjectServiceCallback() {
            @Override
            public void onSuccess(JSONObject response) throws
KonyException {
                kony.print("delete success : " + response.toString());
            }
            @Override
```

```
            public void onFailure(ObjectServiceException
objectServiceException) {
                kony.print("delete failed" +
objectServiceException.getErrorMessage());
            }
            );
    }
```

Fetch

Fetches an object from the server.

39.7.9.13 Syntax

```
 fetch(DataObject dataObject,HashMap<String,String>
headers,HashMap<String,String> queryParams,ObjectServiceCallback
callback)
```

39.7.9.14 Parameters

Input Parameter	Type	Description	Required
dataObject	DataObject	An instance of the DataObject class that contains details about the object and its data	Yes
headers	HashMap<String,String>	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	HashMap<String,String>	Key/ value pairs of query parameters that are appended to the URL while making a network call.	Optional

Input Parameter	Type	Description	Required
objectServiceCallback	ObjectServiceCallback	Interface with onSuccess and onFailure methods • **onSuccess**: Function invoked when the operation succeeds with the primary key of the created object. • **onFailure**: Function invoked when the operation fails with cause of failure.	Yes

39.7.9.15 Code

```
DataObject dataObject = new DataObject("product");
dataObject.setODataUrl("$filter= " + "id" + "eq" + "1");
try {
    objectService.fetch(dataObject, new HashMap < String, String > (),
new HashMap < String, String > (), new ObjectServiceCallback() {
        @Override
        public void onSuccess(JSONObject response) throws
KonyException {
            kony.print("fetch success : " + response.toString());
        }
        @Override
        public void onFailure(ObjectServiceException
objectServiceException) {
            kony.print("fetch failed" +
objectServiceException.getErrorMessage());
        }
```

```
    });
} catch (Exception e) {
    kony.print("exception occurred while deleting an object " +
e.getMessage());
}
```

getMetaDataofAllObjects

Gets the metadata associated with the objects that are defined in the service from the server.

39.7.9.16 Syntax

```
 getMetadataOfAllObjects(boolean getFromServer,String
objectName,HashMap<String,String> headers,HashMap<String,String>
queryParams,ObjectServiceCallback callback)
```

39.7.9.17 Parameters

Input Parameter	Type	Description	Required
getFromServer	Boolean	Flag that retrieves the metadata of the object from the server or local store. • **True** - from server • **False** - from local store	Yes
objectName	String	Name of the object	Yes
headers	HashMap<String,String>	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional

Input Parameter	Type	Description	Required
queryParams	HashMap<String,String>	Key/ value pairs of query parameters that are appended to the URL while making a network call.	Optional
objectServiceCallback	ObjectServiceCallback	Interface with onSuccess and onFailure methods • **onSuccess**: Function invoked when the operation succeeds with the primary key of the created object. • **onFailure**: Function invoked when the operation fails with cause of failure.	Yes

39.7.9.18 Code

```
 try {
    objectService.getMetadataOfAllObjects(true,
dataObject.getObjectName(), null, null, new ObjectServiceCallback() {
            @Override
            public void onSuccess(final JSONObject object) {
                kony.print("get meta data of all objects success");
            }
        }
        @Override
        public void onFailure(final ObjectServiceException
objectServiceException) {
            kony.print("get meta data of all objects failed" +
objectServiceException.getErrorMessage());
```

```
        }
    }
});
```

getMetaDataofObject

Gets the metadata associated with an object that is defined in the service from the server or local store.

39.7.9.19 Syntax

```
 getMetadataOfObject(boolean getFromServer,String
objectName,HashMap<String,String> headers,HashMap<String,String>
queryParams, ObjectServiceCallback callback)
```

39.7.9.20 Parameters

Input Parameter	Type	Description	Required
getFromServer	Boolean	Flag that retrieves the metadata of the object from the server or local store. • **True** from server • **False** - from local store	Yes
objectName	String	Name of the object	Yes
headers	HashMap<String,String>	Key/ value pairs of httpHeaders that are sent along with the network call.	Optional
queryParams	HashMap<String,String>	Key/ value pairs of query parameters that are appended to the URL while making a network call.	Optional

Input Parameter	Type	Description	Required
objectServiceCallback	ObjectServiceCallback	Interface with onSuccess and onFailure methods • **onSuccess**: Function invoked when the operation succeeds with the primary key of the created object. • **onFailure**: Function invoked when the operation fails with cause of failure.	Yes

39.7.9.21 Code

```
 try {
    objectService.getMetadataOfObject(true, dataObject.getObjectName
(), null, null, new ObjectServiceCallback() {
        @Override
        public void onSuccess(final JSONObject object) {
            kony.print("get meta data of object success");
        }
        @Override
        public void onFailure(final ObjectServiceException
objectServiceException) {
            kony.print("get meta data of object failed" +
objectServiceException.getErrorMessage());
        }
    });
} catch (KonyException e) {
    e.printStackTrace();
}
```

39.7.10 Invoking a Metrics Service

When the Android SDK is initialized, it will automatically collect various standard metrics from a client and the standard metrics will be accessible using the Standard Reports within Kony Fabric Console.

The AndroidSDK also provides the ability for a developer to send additional custom metrics from a client app to Kony Fabric back-end to capture additional information. These custom data sets will be accessible using the Custom Reporting feature within Kony Fabric Console where a business analyst can design and share reports using a combination of standard and custom metrics.

Additionally, the Android SDK provides an Events API that allows an app to track user actions within the app to gain insight into the user journey.The developer can send various standard events such as form entry, touch events, service requests, gestures and errors. The developer can also send custom events to capture any app specific scenarios or transactions. These events can be analyzed within Kony Fabric Console by using the Standard Reports or user defined Custom Reports. For more details, refer to Custom Metrics and Reports Guide.

This section lists all `MetricsService` object APIs.

39.7.10.1 Create an instance of MetricsService

The **MetricsService** class sets the configuration for APM event reporting.

```
 //Sample code to create an instance of MetricService with variable
name "getMetricsService"
mKonyClient = new KonyClient();
MetricsService metricsClient = null;
try {
    metricsClient = mKonyClient.getMetricsService();
} catch (KonyCMSException e) {
    e.printStackTrace();
```

setUserID

The **setUserID** API sets the user ID for the data gathered from an application. The user ID allows the data to be tracked on a user basis for broad analysis like how many different users used the application. It also helps to track activities of a specific user, which can help in seeing what activities were done before a crash, or what events led to a transaction not passing through. The user ID allows the same user to be tracked across different devices as well.

```
 //Sample code to set up the user ID of application user
String mUserId = "<user-id>";
metricsClient.setUserId(mUserId);
```

> ***Note:*** The UserId related to metrics. The UserId length cannot be more than 100 characters.

sendEvent

The **sendEvent** API allows a developer to send event details from an application to server for analytics and reporting purposes. The event data is added to a buffer and sent to server as per configuration values set by the developer using setEventConfig API.

```
 //Sample code to send reports
String eventSubType = "<event-sub-type>";
String formID = "<form-id>";
String widgetID = "<widget-id>";
String flowTag = "<flow-tag>";
String metaData = "<meta-data>";
try {
    metricsClient.sendEvent(MetricsService.EventType.EVENT_TYPE_
CUSTOM, eventSubType, formID, widgetID, flowTag, metaData);
} catch (KonyCMSException e) {
    e.printStackTrace();
}
```

The following are the enums event types with values:

- EVENT_TYPE_FORM_ENTRY("FormEntry", 0)
- EVENT_TYPE_FORM_EXIT("FormExit", 1)
- EVENT_TYPE_TOUCH("Touch", 2)
- EVENT_TYPE_SERVICE_REQUEST("ServiceRequest", 3)
- EVENT_TYPE_SERVICE_RESPONSE("ServiceResponse", 4)
- EVENT_TYPE_GESTURE("Gesture", 5)
- EVENT_TYPE_ORIENTATION("Orientation", 6)
- EVENT_TYPE_ERROR("Error", 7)
- EVENT_TYPE_EXCEPTION("Exception", 8)
- EVENT_TYPE_CRASH("Crash", 9)
- EVENT_TYPE_CUSTOM("Custom", 10)
- EVENT_TYPE_SERVICECALL("ServiceCall", 11)
- EVENT_TYPE_APPTRANSITION("AppTransition", 12)
- EVENT_TYPE_APPLOAD("AppLoad", 13)

> ***Note:*** The EventType is an ENUM.
> The eventSubType, formId, widgetId, and flowTag fields can have max of 256 characters.

The following are the parameters for an event type:

- eventType - string literal for formID can be null
- eventSubType - string literal for eventSubType (max 256 characters)
- formID - string literal for formID (max 256 characters)
- widgetID - string literal for widgetID (max 256 characters)

- flowTag - string literal to override flow tag (max 256 characters)
- metaData - string literal that can be set by developer while setting a custom event for sending custom data as part of an event.

flushEvents

The **flushEvents** API allows a developer to force events to be sent to the server. The entire current event buffer is loaded and sent to the server for processing. The flushEvents API used as an override to send event data to server before the configured value or a service call that flushes the buffer.

```
 //Sample code to flushEvents

metricsClient.flushEvents();
```

getEventsInBuffer

The **getEventsInBuffer** returns a list of the buffered events.

```
 //Sample code to eventsInBuffer
Vector < Hashtable > bufferEvents = metricsClient.getEventsInBuffer();
```

setFlowTag

The **setFlowTag** API sets an event flow tag to be associated with all new events that are reported by using the sendEvent API.The flow tag is used to ease searching event data in terms of application flows like loginflow, searchflow. The setFlowTag also helps sorting and filtering data while building custom reports or running standard reports for the application events.

```
 //Sample code to setFlowTagString myFowTag = "<flow-tag>";
try {
    metricsClient.setFlowTag(myFowTag);
} catch (KonyCMSException e) {
    e.printStackTrace();
}
```

clearFlowTag

The **clearFlowTag** API clears the currently set event flow tag.

```
 //Sample code to clearFlowTag
metricsClient.clearFlowTag();
```

getFlowTag

The getFlowTag API gets the currently set event flow tag.

```
 //Sample code to getFlowTag
String mFlowTag = metricsClient.getFlowTag();
```

reportError

The **reportError** API enables an app to report an error event to metrics server.

```
 //Sample code to reportError
String errorcode = "<error-code>";
String errorType = "<error-type>";
String errorMessage = "<error-message>";
String errorDetails = "<error-details>";
try {
    metricsClient.reportError(errorcode, errorType, errorMessage,
errorDetails);
} catch (KonyCMSException e) {
    e.printStackTrace();
}
```

Parameters:

- errorCode - string errorCode can be nil if not applicable.
- errorType - string errorType can be nil if not applicable.
- errorMessage - string errorMessage can be nil if not applicable.

- errorDetails - string(JSON) errorDetails is a json string that can have key-value pairs for the following keys errfile, errmethod, errline, errstacktrace, errcustommsg, errcrashreport, formID, widgetID, and flowTag.

reportHandledException

The **reportHandledException** API enables apps to report a handled exception event.

```
 //Sample code to send exception to metrics server
String exceptioncode = "<exception-code>";
String exceptionType = "<exception-type>";
String exceptionMessage = "<exception-message>";
String exceptionDetails = "<exception-details>";
try {
    metricsClient.reportHandledException(exceptioncode, exceptionType,
exceptionMessage, exceptionDetails);
} catch (KonyCMSException e) {
    e.printStackTrace();
}
```

Parameters:

- exceptionCode - string exceptionCode can be nil if not applicable.
- exceptionType - string type of exception, such as Eval Error or syntax error. The exceptionType can be nil if not applicable.
- exceptionMessage - string exceptionMessage can be nil if not applicable.
- exceptionDetails - string(JSON) exceptionDetails is a JSON string that can have key-value pairs for the following keys exceptionfile, exceptionmethod, exceptionline, exceptionstacktrace, formID, widgetID, and flowTag.

setEventConfig

The **setEventConfig** API takes the required values to set the event configuration values. When eventConfigType is - CONF_TYPE_BUFFER event autoFlushCount and maxBufferCount are considered.

```
 //Sample code to setEventConfig
int autoFlushCount = 20;
int maxBufferCount = 800;
try {
    metricsClient.setEventConfig(MetricsService.EventConfigType.CONF_
TYPE_BUFFER, autoFlushCount, maxBufferCount);
} catch (KonyCMSException e) {
    e.printStackTrace();
}
```

Parameters:

- eventConfigType - sets the current configuration type

  > ***Note:*** Only buffer mode is supported for the eventConfigType currently.

- autoFlushCount - number of events to be buffered before a network call is triggered to post the event data to the server Possible values any positive integer. Default value is 15.
- maxBufferCount - Maximum event buffer count to store the events possible values any positive integer. Default value os 1000.

setBatchSize

The **setBatchSize** API allows a developer to specify the batch size to be set to flush events. Default batch size is 50.

```
 //Sample code to set batch size to flush eventsint batchSize = 20;
try {
    metricsClient.setBatchSize(batchSize);
```

```
} catch (KonyCMSException e) {
    e.printStackTrace();
}
```

sendCustomMetrics

The **sendCustomMetrics** API sends custom metrics event.

```
 //Sample code to send data to reporting service with group id as
"formID"
Hashtable mtable = new Hashtable();
mtable.put("<key>", "<Values>");
String key = "<metrics-key>";
metricsClient.sendCustomMetrics(formID, mtable);
```

For more details about custom metrics and reports, refer to Custom Metrics and Reports Guide.

Parameters:

- groupId - formID length cannot be more than 250 characters.
- data - data to be send

39.7.10.2 Event Details

For all event details, **ts** and **SID** values are automatically filled by MBaaS Client SDK, as part of the reportEvent, reportError and reportHandledException API calls. In case of automatically captured events, flowTag is also automatically filled with the currently set flowTag. Following are event specific details to be used while interfacing with MBaaS SDK.

FormEntry

- API to be used - sendEvent
- evtType - FormEntry
- FormID - value of the ID property of the form widget
- WidgetID - null

- evtSubType - Value of the ID property of the form widget
- metadata - null

FormExit

- API to be used - sendEvent
- evtType - FormExit
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - Dictionary (hash table) that contains the following key value pairs:
    - formdur - Duration spent in form in milliseconds. Optional.

Touch

- API to be used - sendEvent
- evtType - Touch
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - value of the ID property of the widget on which the touch happened
- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget)
- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget). Touch event is extended to the below widgets along with button onclick.
    - Flex Container/Scroll Container

      onClick

onTouchStart (if registered)

onTouchEnd (if registered)

- Segment

  onRowClick

- Button

  onClick (already in place, no new changes)

- Image

  onTouchStart (if registered)

  onTouchEnd (if registered)

- Switch

  onSlide

- metadata - null

ServiceRequest

- API to be used - sendEvent
- evtType - ServiceRequest (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata - null

ServiceResponse

- API to be used - sendEvent
- evtType - ServiceResponse (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata
  Dictionary (hash table) contains the following key value pairs:
  - opstatus - Optional
    returned by Kony servers
  - httpcode - HTTP status code
  - resptime - time taken to get the response.

Gesture

- API to be used - sendEvent
- evtType - Gesture (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget where the gesture happened
- WidgetID - value of the ID property of the widget on which the gesture happened
- evtSubType [String]
  `GESTURETYPE_NUMBEROFINPUTS_DIRECTION`
  For example, two finger left swipe - SWIPE_2_LEFT
- metadata - null

Orientation

- API to be used - sendEvent
- evtType - Orientation (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String] - any one of the below constants is used
    - PORTRAIT_TO_LANDSCAPE
    - LANDSCAPE_TO_PORTRAIT
- metadata - null

Error

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ErrorCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:
    - errcode - Optional
    - errmsg - Optional
    - errfile - Optional
    - errmethod - Optional

- errstacktrace - Optional
- errcustommsg - Optional

HandledException

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ExceptionCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:
  - exceptioncode - Optional
  - exceptionev - Optional
  - exceptionmsg - Optional
  - exceptionfile - Optional
  - exceptionmethod - Optional
  - exceptionstacktrace - Optional
  - exceptioncustommsg - Optional

Crash

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String]

- metadata
  Dictionary (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional
  - errline - Optional
  - errstacktrace - Optional
  - errcrashreport - Optional

Custom

- API to be used - sendEvent
- evtType - Custom
- FormID - any supplied form ID
- WidgetID - any supplied widget ID
- evtSubType - any supplied event subtype
- metadata - string or a dictionary

AppTransition

- API to be used - sendEvent
- evtType - AppTransition
- FormID - Value of the ID property of the form widget where the touch happened
- WidgetID - null

- evtSubType [String] - any one of the below string constants will obtain:
  - Background
  - Foreground
  - metadata
- Dictionary (hash table) that contains the following key value pairs:
  foredur - Duration spent in milliseconds.

AppLoad

- API to be used - sendEvent
- evtType - AppLoad
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - null
- evtSubType [String] – AppId which is the Application ID.
- metadata
- Dictionary (hash table) that contains the following key value pairs:
  loaddur - Duration spent in milliseconds.

39.8 .NET (Visual Studio) SDK

This section describes how to download the .NET (Visual Studio) SDK files and initialize the .NET client.

- Prerequisites
- Downloading the .NET (Visual Studio) SDK
- Initializing the .NET (Visual Studio) SDK
- Invoking an Identity Service

- Invoking an Integration Service
- Invoking a Metrics Service
- Invoking an Object Service
- API Reference

  To view the API Reference for Plain Windows, click Kony Windows docset.

39.8.1 Prerequisites

- .NET Framework 4.6.1 or higher
- NuGet Packages:
  - Acr.DeviceInfo 6.5.0 or higher
  - Newtonsoft.Json 12.0.1 or higher

> ***Note:*** Kony does not explicitly test or certify the .NET SDK with Xamarin, but it is designed to be compatible with Xamarin.

39.8.2 Downloading .NET (Visual Studio) SDK Files

To download the .NET SDK, follow these steps:

1. In the Kony Fabric console, navigate to **Apps** > **SDKs**, and click **.NET (Visual Studio)**. The system prompts you to save the zip file in your local system.

2. Save the `kony-windows-sdk.zip` file in your local system.
3. Extract `kony-windows-sdk.zip` file that you just downloaded.
   The `WindowsSDK` folder contains the following files
   - kony.SDK.dll
   - HELP
   - LICENSE.txt
   - Newtonsoft.Json.dll

39.8.3 Initializing the .NET SDK

Perform the following steps to initialize the .NET SDK.

1. Include the downloaded .dll files in your Visual Studio project.
2. Create an instance of the SDK.

```
Kony.SDK sdkObject = new Kony.SDK();
```

Use the methods and properties of the SDK class to initialize and access the .NET SDK.

When SDK is initialized, the Kony SDK registers a session and sends its information to the Kony Fabric Server. If the device is offline, or the server is not reachable, the session information persists on the device until it can successfully send the information to the Kony Fabric server.

For more information on application session, refer Standard Report Docs.

> ***Note:*** The .NET SDK does not support the Messaging Service or the Sync Service.

> ***Note:*** The sessions created by .NET Fabric SDK are interactive.

39.8.3.1 Init

To start using the services provided in Kony Fabric use the **Init()** method of the SDK class. This method may take several seconds to execute, so it is available as either a synchronous or asynchronous call.

This method fetches the app configuration from the Kony server and saves it in the cache. Later, the application uses the cached configuration. This method must be invoked before invoking any other SDK method.

Signature

Synchronous

```
JObject Init(string appKey, string appSecret, string serviceUrl)
```

Asynchronous

```
async Task<JObject> InitAsync(string appKey, string appSecret, string
serviceUrl)
```

Input Parameters

appKey

App key of the Kony application.

appSecret

App secret of the Kony application.

serviceUrl

URL of the Kony server.

Return Values

Returns Service doc in the form of a **JObject**.

Exceptions

System.Exception

The **Init** method throws this exception when an error occurs while fetching the data or instantiating a service.

Example

Synchronous

```
 //Sample code to initialize Kony Fabric Client synchronously.
string appkey = < your - app - key >
    string appsecret = < your - app - secret >
    string serviceURL = < your - service - url >

Kony.SDK sdkObject = new Kony.sdk();
try {
    JObject serviceDoc = sdkObject.Init(appkey, appsecret,
serviceURL);
    Console.WriteLine("Init Success");
} catch (Exception e) {
    Console.WriteLine("Init Failure");
}
```

Asynchronous

```
 //Sample code to initialize Kony Fabric Client asynchronously.
string appkey = < your - app - key >
    string appsecret = < your - app - secret >
    string serviceURL = < your - service - url >

Kony.SDK sdkObject = new Kony.sdk();
try {
    JObject serviceObj = await sdkObject.InitAsync(appkey, appsecret,
serviceURL);
    Console.WriteLine("Init Success");
} catch (Exception e) {
    Console.WriteLine("Init Failure");
}
```

39.8.4 Invoking an Identity Service

Perform the following steps to invoke an identity service.

- Login with provider type as Basic
- Get Backend Token
- User Profile
- Get Provider Name
- Get Provider Type
- Logout

> ***Note:*** The .NET SDK does not support the Custom Auth Identity Service.
> The .NET SDK does not support the OAuth/SAML Identity Service.

39.8.4.1 Login with provider type as Basic

To login, first use the SDK class **GetIdentityService()** method to retrieve the IdentityService. Then log in with the **login()** method of the IdentityService class.

GetIdentityService

The **GetIdentityService** method retrieves an IdentityService object with the specified provider name.

Signature

```
IdentityService GetIdentityService(string providerName)
```

Input Parameters

providerName

The name of the provider.

Return Values

Returns an IdentityService instance.

Exceptions

System.Exception is thrown when not initialized or the service is not defined.

Example

```
// Sample code to retrieve the IdentityService
instance.Kony.IdentityService identity;
string providerName = < your - provider - name > ;
try {
    identity = sdkObject.GetIdentityService(providerName);
} catch (Exception e) {
    Console.WriteLine("Invalid Provider");
}
```

When you have an IdentityService object, use the Login method of the IdentityService class to login.

Login

The **Login** method will login the user with the given credentials. This method may take several seconds to execute, so it is available as either a synchronous and asynchronous call.

> ***Note:*** The .NET SDK does not support multilogin, offfline login, or SSO.

Signature

Synchronous

```
 string Login(string userId, string password, Dictionary<string,
string> headers, Dictionary<string, string> parameters)
```

Asynchronous

```
async Task < string > LoginAsync(string userId, string password,
Dictionary < string, string > headers, Dictionary string,string>
parameters)
```

Input Parameters

userId

The user name. This value cannot be null.

password

The user's password. This value cannot be null.

headers

HTTP Request header key value pairs. (Optional)

parameters

HTTP Request parameter key value pairs. (Optional)

Return Values

Returns the claims token value if the login is successful.

Remarks

If the provider is Kony, parameters **userID** and **password** must not be null or empty.

Exceptions

A Kony FabricException is thrown if the provider is Kony and **userID** or **password** are empty or null.

Example

Synchronous

```
 // Sample code to login synchronously.
// sdkObject is the SDK object.
Kony.IdentityService identity;
string providerName = < your - provider - name > ;
string username = < username -
for -your - provider > ;
string password = < password -
for -your - provider > ;

try {
    identity = sdkObject.GetIdentityService(providerName);
    string claimsToken = identity.Login(username, password);
    Console.WriteLine("Login Success");
} catch (Exception e) {
    if (identity == null) {
        Console.WriteLine("Invalid provider");
    } else {
        Console.WriteLine("Login Failure");
    }
}
```

Asynchronous

```
 // Sample code to login asynchronously.// sdkObject is the SDK
object.
Kony.IdentityService identity;
string providerName = < your - provider - name > ;
string username = < username -
for -your - provider > ;
string password = < password -
for -your - provider > ;

try {
    identity = sdkObject.GetIdentityService(providerName);
    string claimsToken = await identity.LoginAsync(username,
password);
    Console.WriteLine("Login Success");
} catch (Exception e) {
    if (identity == null) {
        Console.WriteLine("Invalid provider");
    } else {
        Console.WriteLine("Login Failure");
    }
}
```

39.8.4.2 Get Backend Token

Use the IdentityService class method **GetBackendToken()** to get the provider datasource token.

GetBackendToken

The **GetBackendToken** method fetches the provider datasource token. This method may take several seconds to execute so it is available as both a synchronous and asynchronous call.

Signature

Synchronous

```
 ProviderToken GetBackendToken(bool fromServer, Dictionary<string,
string> parameters)
```

Asynchronous

```
 async Task<ProviderToken> GetBackendTokenAsync(bool fromServer,
Dictionary<string, string> parameters)
```

Input Parameters

fromServer

Flag to force fetch from server only.

parameters

HTTP request parameters.

Return Values

Returns a provider token object.

Remarks

If fromServer is **true**, the SDK fetches the token from the server. If fromServer is **false**, the SDK returns the token present in local storage.

Exceptions

Kony FabricException is thrown if the provider claims token is invalid.

Example

Synchronous

```
// Sample code to get backend token for provider synchronously.
string username = < username - for -logged - in -provider > ;
string password = < password - for -logged - in -provider > ;
bool fromServer = < true / false > ;
```

```
Dictionary < string, string > parameters = new Dictionary < string,
string > ();
parameters.userid = username;
parameters.password = password;
Kony.ProviderToken backendToken;

try {
    backendToken = userstore.GetBackendToken(fromServer, parameters);
    Console.WriteLine("Backend Token fetched");
} catch (Exception e) {
    Console.WriteLine("Failed to get backend token");
}
```

Asynchronous

```
// Sample code to get backend token for provider asynchronously.
string username = < username - for -logged - in -provider > ;
string password = < password - or -logged - in -provider > ;
bool fromServer = < true / false > ;
Dictionary < string, string > parameters = new Dictionary < string,
string > ();
parameters.userid = username;
parameters.password = password;
Kony.ProviderToken backendToken;

try {
    backendToken = await userstore.GetBackendTokenAsync(fromServer,
parameters);
    Console.WriteLine("Backend Token fetched");
} catch (Exception e) {
    Console.WriteLine("Failed to get backend token");
```

```
}
```

39.8.4.3 Get User Profile

Use the IdentityService class method **GetProfile()** to get the user profile.

GetProfile

The **GetProfile** method fetches the user profile. This method may take a few seconds to execute, so it is available as either a synchronous or asynchronous call.

Signature

Synchronous

```
Profile GetProfile(bool fromServer)
```

Asynchronous

```
async Task<Profile> GetProfileAsync(bool fromServer)
```

Input Parameters

fromServer

Flag to force fetch from server only. Default value is false.

Return Values

Returns a user profile object.

Remarks

If fromServer is **true**, the SDK fetches the profile from the server. If fromServer is **false**, the SDK returns the profile present in local storage.

Exceptions

Kony Fabric Exception is thrown if the provider claims token is invalid.

Example

Synchronous

```
 // Sample code to get user profile details synchronously.
// identity is the IdentityService object.
bool fromServer = < true / false > ;
Kony.Profile profile = identity.GetProfile(fromServer);
if (profile == null) {
    Console.WriteLine("Unable to get profile");
}
```

Asynchronous

```
 // Sample code to get user profile details asynchronously.
// identity is the IdentityService object.
bool fromServer = < true / false > ;
Kony.Profile profile = await identity.GetProfileAsync(fromServer);
if (profile == null) {
    Console.WriteLine("Unable to get profile");
}
```

39.8.4.4 Get Provider Name

Use the IdentityService class **Name** property to get the authentication provider name.

Name

```
 // Sample code to get provider name
// identity is the IdentityService object.
string providerName = identity.Name;
```

39.8.4.5 Get Provider Type

Use the IdentityService class **Type** property to get the authentication provider type.

Type

```
 // Sample code to get provider type
// identity is the IdentityService object.
string providerType = identity.Type;
```

39.8.4.6 Logout

Use the **Logout()** method of the IdentityService class to logout from the service.

Logout

The **Logout** method is used to logout of the service. This method may take a few seconds to execute, so it is available as both a synchronous and asynchronous call.

Signature

Synchronous

```
bool Logout()
```

Asynchronous

```
async Task<bool> LogoutAsync()
```

Input Parameters

None.

Return Values

Returns true if the logout is successful.

Exceptions

Kony FabricException is thrown when a web services call exception occurs.

Example

Synchronous

```
 // Sample code to logout from auth service synchronously.
// identity is the IdentityService object.
bool isLoggedOut;
try {
    isLoggedOut = identity.Logout();
} catch (Exception e) {
    Console.WriteLine("Unable to logout");
}
```

Asynchronous

```
 // Sample code to logout from auth service asynchronously.
// identity is the IdentityService object.
bool isLoggedOut;
try {
    isLoggedOut = await identity.LogoutAsync();
} catch (Exception e) {
    Console.WriteLine("Unable to logout");
}
```

39.8.5 Invoking an Integration Service

To invoke an Integration service, use the SDK class **GetIntegrationService()** method to get the service object. To invoke an operation, use the returned IntegrationService object.

39.8.5.1 GetIntegrationService

This API retrieves an IntegrationService instance with the specified service name.

Signature

```
 IntegrationService GetIntegrationService(string serviceName)
```

Input Parameters

serviceName

The name of the service.

Return Values

Returns an IntegrationService object.

Exceptions

System.Exception

The API throws this exception when not initialized, the claim token is undefined, or the service is undefined.

Example

```
 // Sample code to fetch the integration service details.
Kony.KonyService integrationSvc;
var serviceName = < your - service - name > ;
try {
    integrationSvc = sdkObject.GetIntegrationService(serviceName);
} catch (Exception e) {
    Console.WriteLine("Invalid Service");
}
```

39.8.5.2 InvokeOperation

The **InvokeOperation** method invokes the specified operation. This method may take a few seconds, so it is available as either a synchronous or asynchronous call.

Signature

Synchronous

```
 JObject InvokeOperation(string operationName, Dictionary<string,
string> headers, Dictionary<string, string> parameters)
```

Asynchronous

```
 async Task<JObject> InvokeOperationAsync(string operationName,
Dictionary<string, string> headers, Dictionary<string, string>
parameters)
```

Input Parameters

operationName

The name of the operation.

headers

HTTP request header key value pairs.

parameters

HTTP request parameter key value pairs.

Return Values

Returns an HTTP Response as a JSON object.

Exceptions

Kony FabricException

The **InvokeOperation** method throws this exception when either Name or URL is either empty or null.

Example

Asynchronous

```
 // Sample code to invoke the specified operation asynchronously.//
sdk.Object is the SDK object.
string serviceName = < your - service - name > ;
string operationName = < your - operation - name > ;
Dictionary < string, string > params = new Dictionary < string, string
> ();
Dictionary < string, string > headers = new Dictionary < string,
string > ();
JObject result;

try {
    integrationSvc = sdkObject.GetIntegrationService(serviceName);
    result = await integrationSvc.InvokeOperationAsync(operationName,
headers, params);
} catch (Exception e) {
    if (integrationSvc == null) {
        Console.WriteLine("Invalid Service");
    } else {
        Console.WriteLine("Invoke operation failed");
    }
}
```

39.8.6 Invoking a Metrics Service Object

When the .NET SDK is initialized, it automatically collects various standard metrics from a client and the standard metrics will be accessible using the Standard Reports within Kony Fabric Console.

The .NET SDK also provides the ability for a developer to send additional custom metrics from a client app to Kony Fabric back-end to capture additional information. These custom data sets will be accessible using the Custom Reporting feature within Kony Fabric Console where a business analyst can design and share reports using a combination of standard and custom metrics.

Additionally, the .NET SDK provides an Events API that allows an app to track user actions within the app to gain insight into the user journey. The developer can send various standard events such as form entry, touch events, service requests, gestures and errors. The developer can also send custom events to capture any app specific scenarios or transactions. These events can be analyzed within Kony Fabric Console by using the Standard Reports or user defined Custom Reports. For more details, refer to Custom Metrics and Reports Guide.

39.8.6.1 Create an instance of MetricsService

You must initialize the MetricsService class before using any functionality related to analytics of the app built in .NET SDK.

```
 // Sample code to fetch the metrics service object.
Kony.MetricsService metricsSvc;

try {
    metricsSvc = sdkObject.GetMetricsService();
} catch (Exception e) {
    Console.WriteLine("Invalid Service");
}
```

39.8.6.2 Metrics Service Object Methods

This section describes the methods of the Metrics Service.

ClearFlowTag

The ClearFlowTag method clears the set flow tag.

Syntax

```
 ClearFlowTag();
```

Parameters

None

Returns

None

Example

```
 // Sample code for ClearFlowTag

var metricsServiceObj = servicesOjb.GetMetricsService();
metricsSericeObj.ClearFlowTag();
```

FlushEvents

The FlushEvents method allows a developer to send events collected in buffer form from the device to the server. The entire current event buffer is loaded and sent to the server for processing. The FlushEvents method is used as an override to send event data to the server before the configured value or a service call that flushes the buffer.

Syntax

```
 FlushEvents()
```

Parameters

None

Returns

None

Example

```
 //Sample code for FlushEvents
var metricsServiceObj = servicesObj.GetMetricsService();
metricsServiceObj.FlushEvents();
```

GetFlowTag

The GetFlowTag method gets the set flow tag.

Syntax:

```
 GetFlowTag(flowTag);
```

Parameters

None

Returns

None

Example

```
// Sample code for GetFlowTag

string getflowtag;
var metricsServiceObj = servicesOjb.GetMetricsService();
getflowtag = metricsSericeObj.GetFlowTag();
```

ReportError

The ReportError method enables an app to report an error event to metrics server.

Syntax

```
 ReportError(string errorCode, stsring errorType, string errorMessage,
string errorDetails)
```

Parameters

errorCode - string. Can be null if not applicable.

errorType - string. Can be null if not applicable.

errorMessage - string. Can be null if not applicable.

errorDetails - json string. This param is a json string that can have key-value pairs for the following keys: errfile, errmethod, errline, errstacktrace, errcustommsg, errcrashreport, formID, widgetID, and flowTag.

Example

```
 //Sample code for ReportError
var metricsServiceObj = servicesObj.GetMetricsService()
metricsServiceObj.ReportError("1234", "SpecificError", "custom error
message", "{errfile:file.js}");
```

ReportHandledException

The ReportHandledException method enables apps to report a handled exception event.

Syntax

```
 ReportHandledException(string exceptionCode, string exceptionType,
string ExceptionMessage, string exceptionDetails)
```

Example

```
 //Sample code to send exception to metrics server
var metricsServiceObj = servicesObj.GetMetricsService()
metricsServiceObj.ReportHandledException("1234", "SpecificException",
"custom exception message", "{errfile:file.js}");
```

SendCustomMetrics

The SendCustomMetrics method allows the developer to send custom metrics from the application. The custom metrics should already be registered in Kony Fabric Console for the application before data is sent from the application.

Syntax

```
 SendCustomMetrics(formId, reportingData);
```

Parameters

formId - string. Specifies the name of the form to which the custom metrics is to be sent. Cannot be more than 250 characters.

reportingData- Dictionary<string, string> of key/value pairs. Key is the custom metric name and value is the value given for that specific instant. For example: Rating, Excellent.

Example

```
 //Sample code for SendCustomMetrics
var metricsServiceObj = servicesObj.GetMetricsService();
Dictionary < object, object > reportingData = new Dictionary < object,
object > ();
reportingData.Add("Rating", "Excellent");
metricsServiceObj.SendCustomMetrics(formId, reportingData);
```

SendEvent

The SendEvent method allows a developer to send event details from an application to the server for analytics and reporting purposes. The event data is added to a buffer and sent to the server as per configuration values set by the developer using the SetEventConfig method.

Example

```
 //Sample code for SendCustomMetrics
var metricsServiceObj = servicesObj.GetMetricsService();
Dictionary < object, object > reportingData = new Dictionary < object,
object > ();
reportingData.Add("Rating", "Excellent");
metricsServiceObj.SendCustomMetrics(formId, reportingData);//Sample
code to send reports
var metricsServiceObj = servicesObj.GetMetricsService()
string eventSubType = "subTypeSample";
string formID = "sampleFormID";
string widgetID = "sampleWidgetID";
string flowTag = "sampleFlowTag";
JObject metaData = new JObject();
metaData.Add("formdur", "100");
try {
    metricsServiceObj.SendEvent(EventType.Custom, eventSubType,
```

```
formID, widgetID, flowTag, metaData);
} catch () {}
```

SetEventConfig

The SetEventConfig method sets the event config param. Based on the **autoFlushCount** param, there will be an automatic flush from the device to the server.

Syntax

```
 SetEventConfig(eventConfigType, autoFlushCount, maxBufferCount);
```

Parameters

eventConfigType - sets the current configuration type. Only buffer mode is supported.

autoFlushCount - the number of events to be buffered before a network call is triggered to post the event data to the server. Possible value is any positive integer. Default value is 15.

maxBufferCount - the maximum numnber of event buffers to store the events. Possible value is any positive integer. Default value is 1000.

Example

```
 // Sample code for SetEventConfig

int autoFlushCount = 20;
int maxBufferCount = 800;
try {
    metricsClient.SetEventConfig(EventConfigType.BUFFER,
autoFlashCount, maxBufferCount);
} catch (Exception e) {}
```

SetFlowTag

The SetFlowTag method sets the flow tag.

Syntax

```
SetFlowTag(flowTag);
```

Parameters

flowTag - string that specifies the flow tag.

Example

```
// Sample code for SetFlowTag

string flowtag = "SampleFlowTag";
var metricsServiceObj = servicesOjb.GetMetricsService();
metricsSericeObj.SetFlowTag(flowtag);
```

SetUserId

The SetUserId method sets the user ID.

Syntax

```
SetUserId(userid);
```

Parameters

userid - string that specifies the user ID.

Example

```
// Sample code for SetUserId

string userid = "SampleUserID";
var metricsServiceObj = servicesObj.GetMetricsService();
metricsServiceObj.SetUserId(userid);
```

For more details about custom metrics and reports, refer to Custom Metrics and Reports Guide.

39.8.6.3 Event Details

For all event details, ts and SID values are automatically filled by MBaaS Client SDK, as part of the reportEvent, reportError and reportHandledException API calls. In case of automatically captured events, flowTag is also automatically filled with the currently set flowTag. Following are event specific details to be used while interfacing with MBaaS SDK.

FormEntry

- API to be used - sendEvent
- evtType - FormEntry
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - null

FormExit

- API to be used - sendEvent
- evtType - FormExit
- FormID - value of the ID property of the form widget
- WidgetID - null
- evtSubType - Value of the ID property of the form widget
- metadata - Dictionary (hash table) that contains the following key value pairs:
    - formdur - Duration spent in form in milliseconds. Optional.

Touch

- API to be used - sendEvent
- evtType - Touch
- FormID - value of the ID property of the form widget where the touch happened
- WidgetID - value of the ID property of the widget on which the touch happened
- evtSubType - value of this attribute depends upon where the touch happened. Button_Click should be used when touch happens to be a click event on button widget)
- metadata - null

ServiceRequest

- API to be used - sendEvent
- evtType - ServiceRequest (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata - null

ServiceResponse

- API to be used - sendEvent
- evtType - ServiceResponse (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType
  Service ID - in case of service invoking Kony middle ware
  URL - in case of other requests
- metadata
  Dictionary (hash table) containing following key value pairs:
  - opstatus - Optional
    returned by Kony servers
  - httpcode - HTTP status code
  - resptime - time taken to get the reponse.

Gesture

- API to be used - sendEvent
- evtType - Gesture (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget where the gesture happened
- WidgetID - value of the ID property of the widget on which the gesture happened
- evtSubType [String]
  `GESTURETYPE_NUMBEROFINPUTS_DIRECTION`
  For example, two finger left swipe - SWIPE_2_LEFT
- metadata - null

Orientation

- API to be used - sendEvent
- evtType - Orientation (constant, exposed by MBaaS SDK)
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType [String] - any one of the below constants is used
    - PORTRAIT_TO_LANDSCAPE
    - LANDSCAPE_TO_PORTRAIT
- metadata - null

Error

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType
  ErrorCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:
    - errcode - Optional
    - errmsg - Optional
    - errfile - Optional
    - errmethod - Optional
    - errstacktrace - Optional
    - errcustommsg - Optional

HandledException

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null

- evtSubType
  ExceptionCode - Optional
- metadata
  Dictionary (hash table) containing following key value pairs:
  - exceptioncode - Optional
  - exceptionev - Optional
  - exceptionmsg - Optional
  - exceptionfile - Optional
  - exceptionmethod - Optional
  - exceptionstacktrace - Optional
  - exceptioncustommsg - Optional

Crash

- API to be used - sendEvent
- FormID - value of the ID property of the form widget currently displayed on the screen
- WidgetID - null
- evtSubType [String]
- metadata
  Dictionary (hash table) containing following key value pairs:
  - errcode - Optional
  - errmsg - Optional
  - errfile - Optional
  - errmethod - Optional

- errline - Optional
- errstacktrace - Optional
- errcrashreport - Optional

Custom

- API to be used - sendEvent
- evtType - Custom
- FormID - any supplied form ID
- WidgetID - any supplied widget ID
- evtSubType - any supplied event subtype
- metadata - string or a dictionary

39.8.7 Invoking an Object Service

Kony provides you with programmatic access to backend data. Access is gained in two steps:

1. Acquire a current instance of your object service.
2. Use the object service instance together with data objects to communicate as needed with your backend.

To know more about how to acquire a current instance of your object service, refer to Creating an ObjectService Object documentation.

To know more about the methods that act on the Kony Fabric endpoint directly, refer to ObjectService Class documentation.

To know more about the usage of the data transfer objects, refer to DataObject Class documentation.

> ***Note:*** The .NET SDK does not support Offline Object Services.

39.8.7.1 Creating an ObjectService Object

The following example shows how to create an instance of the ObjectService class. This example first creates an instance of the SDK. It then calls the **Init** method before creating the ObjectService instance.

Example

```
public void GetObjectServiceHandle() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc = sdkObject.ObjectServices
("sapobj");
    } catch (Exception e) {
        sdkObject.Log("GetObjectService Failed");
    }
}
```

39.8.7.2 ObjectService Class

The ObjectService class provides methods that act on the Kony Fabric endpoint, including basic CRUD and metadata functions .

Methods

The following methods are used by the ObjectService class.

Create Method

Creates an object in the Kony Fabric endpoint.

39.8.7.3 Syntax

```
Create(dataObject, headers, queryParams);
```

39.8.7.4 Parameters

Parameter	Type	Description
dataObject	DataObject	An instance of the DataObject class which contains data about the object and its data.
headers	Dictionary <string, string>	Key/value pairs of httpHeaders that are sent along with the network call.
queryParams	Dictionary <string, string>	Key/value pairs of query params that are appended to the url while making a network call.

39.8.7.5 Example

```
public void CreateObject() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc = sdkObject.GetObjectService
("<service_name>");
        Kony.DataObject dataObject = new Kony.DataObject("<object_
name>");
        dataObject.AddField("field1", "value1");
        JObject result = objectsvc.Create(dataObject);
    } catch (Exception e) {
        sdkObject.Log("Create Object Failed");
    }
}
```

DeleteRecord Method

Deletes an object in the Kony Fabric endpoint.

39.8.7.6 Syntax

```
DeleteRecord(dataobject, headers, queryParams);
```

39.8.7.7 Parameters

Parameter	Type	Description
dataObject	DataObject	An instance of the DataObject class which contains data about the object and its data.
headers (Optional)	Dictionary <string, string>	Key/value pairs of httpHeaders that are sent along with the network call.
queryParams (Optional)	Dictionary <string, string>	Key/value pairs of query params that are appended to the url while making a network call.

39.8.7.8 Example

```
 public void DeleteObject() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc = sdkObject.ObjectServices
("<service_name>");
        Kony.DataObject dataObject = new Kony.DataObject("<object_
name>");
        //Primary Key is Required to perform the DeleteRecord
operation.
        dataObject.DeleteRecord("Primary_Key", "value");
        JObject result = objectsvc.DeleteRecord(dataObject);
    } catch (Exception e) {
        sdkObject.Log("Delete Object Failed");
```

```
    }
}
```

GetMetadataOfAllObjects Method

Gets the metadata associated with the objects defined in the service from the server or local store.

39.8.7.9 Syntax

```
GetMetadataOfAllObjects(headers, queryParams);
```

39.8.7.10 Parameters

Parameter	Type	Description
headers	Dictionary <string, string>	Key/value pairs of httpHeaders that are sent along with the network call.
queryParams	Dictionary <string, string>	Key/value pairs of query params that are appended to the url while making a network call. To query data, the Odata Url is sent in this parameter.

39.8.7.11 Example

```
public void GetMetadataForAllObjects() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc = sdkObject.ObjectServices
("<service_name>");
        Kony.Metadata objectsMetadata = null;
        objectsMetadata = objectsvc.GetMetadataOfAllObjects();
    } catch (Exception e) {
        sdkObject.Log("Get Metadata Of All Objects Failed");
```

```
    }
}
```

GetMetadataOfObject Method

Gets the metadata associated with an object defined in the service from the server or local store.

39.8.7.12 Syntax

```
GetMetadataOfObject(objectName, headers, queryParams);
```

39.8.7.13 Parameters

Parameter	Type	Description
objectName	string	Object name in Kony Fabric
headers	Dictionary <string, string>	Key/value pairs of httpHeaders that are sent along with the network call.
queryParams	Dictionary <string, string>	Key/value pairs of query params that are appended to the url while making a network call.

39.8.7.14 Example

```
public void GetMetadataForObject() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc = sdkObject.ObjectServices
("<service_name>");
        Kony.Metadata objectsMetadata = null;
        objectsMetadata = objectsvc.GetMetadataOfObject("<object_
Name>");
    } catch (Exception e) {
```

```
        sdkObject.Log("Get Metadata Of Object Failed");
    }
}
```

PartialUpdate Method

Partially updates an object in the Kony Fabric endpoint.

39.8.7.15 Syntax

```
PartialUpdate(dataObject);
```

39.8.7.16 Parameters

Parameter	Type	Description
dataObject	DataObject	An instance of the DataObject class which contains data about the object and its data.
headers	Dictionary <string, string>	Key/value pairs of httpHeaders that are sent along with the network call.
queryParams	Dictionary <string, string>	Key/value pairs of query params that are appended to the url while making a network call.

39.8.7.17 Example

```
public void PartialUpdateSampleObject() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc =
sdkObject.ObjectServices.<service_name>;
        Kony.DataObject dataObject = new Kony.DataObject("<object_
name>");
```

```
        dataObject.AddField("Updatedfield1", "Updatedvalue1");
        //Primary Key is required to perform the update operation.
        dataObject.AddField("Primary_Key", "value");
        JObject result = objectsvc.PartialUpdate(dataObject);
    } catch (Exception e) {
        sdkObject.Log("Update Object Failed");
    }
}
```

Update Method

Updates an object in the Kony Fabric endpoint.

39.8.7.18 Syntax

```
Update(dataobject, headers, queryParams);
```

39.8.7.19 Parameters

Parameter	Type	Description
dataObject	DataObject	An instance of the DataObject class which contains data about the object and its data.
headers	Dictionary <string, string>	Key/value pairs of httpHeaders that are sent along with the network call.
queryParams	Dictionary <string, string>	Key/value pairs of query params that are appended to the url while making a network call.

39.8.7.20 Example

```
public void UpdateSampleObject() {
    Kony.SDK sdkObject = new Kony.SDK();
    try {
```

```
        sdkObject.Init("<app_key>", "<app_secret>", "<service_url>");
        Kony.ObjectService objectsvc = sdkObject.ObjectServices
("<service_name>");
        Kony.DataObject dataObject = new Kony.DataObject("<object_
name>");
        dataObject.AddField("Updatedfield1", "Updatedvalue1");
        //Primary Key is required to perform the update operation.
        dataObject.AddField("Primary_Key", "value");
        JObject result = objectsvc.Update(dataObject);
    } catch (Exception e) {
        sdkObject.Log("Update Object Failed");
    }
}
```

39.8.7.21 DataObject Class

The DataObject class represents a data object in the Object Service. An instance of this class is required as a parameter in many methods of the ObjectService class.

Constructors

The DataObject class has one constructor.

DataObject

Creates an instance of the DataObject class.

39.8.7.22 Syntax

```
DataObject(objectName, recordObject);
```

39.8.7.23 Parameters

Parameter	Type	Description
objectName	string	An instance of the DataObject class which contains data about the object and its data.

Parameter	Type	Description
recordObject	Dictionary <string, object>	An Dictionary object through which the record is sent.

39.8.7.24 Example

```
Kony.DataObject dataObject = new Kony.DataObject("objectName");
dataObject.AddField("field", "value");
```

40. Settings - Kony Cloud

Using **Settings** in Kony Cloud, you can manage users and accounts based on your role, for example, inviting users, assigning roles to users, managing reports and environments permissions, deleting users, and exporting users.

40.1 Roles Permissions to Access Fabric Console

Roles help to users to access Fabric Console and perform certain operations. There are four types of roles Fabric including the owner, admin, member, and Dev portal.

- **Owner**: An Owner role has access to the complete functionality of Fabric Console. An Owner has access to advanced configurations and operations such as **external auth configuration**, **MFA for account level**, and so on. An Owner is a super-set of the Admin role.
- **Admin**: An Admin role has access to the functionality of the Fabric Console.
- **Member**: A Member role has access to the limited functionality of Fabric Console.
- **Dev Portal**: Lets you create a Portal for exposing APIs created using Kony Fabric. Developers from internal and external partner teams can access the portal created to explore and test the APIs.

Refer to the following screen-shot for more details.

PERMISSIONS	OWNER	ADMIN	MEMBER
Clouds	✓	✓	✓
Manage Clouds	✓	✓	✓
Add a new Cloud	✓	✓	
VPN	✓	✓	
Manage VPN	✓	✓	
Add a new VPN Connection	✓	✓	
Users	✓	✓	
Manage Users	✓	✓	
Invite Users	✓	✓	
Account information	✓	✓	✓
Profile	✓	✓	✓
Reporting	✓	✓	✓
Standard Reports	✓	✓	✓
Custom Metrics	✓	✓	Configurable
Custom Reports	✓	✓	Configurable

Note: The following features can be granted by using the **Custom Access** functionality:
- Build Client App
- Engagement Services
- App Services
- Logic Services
- Logging Console

Note: The Feature Level Access Permissions (**Custom Access**) functionality supported only for Fabric Clouds.

40.2 Environment Permissions - Fabric Clouds

An environment can have 3 types of access permissions to users.

- **No Access**: Users cannot access a Kony environment.
- **Full Access**: Users can access a Kony environment, which helps to build apps, publish apps, and Admin Console.
- **Custom Access**: The **Feature Level Access** permission is supported only for Fabric clouds. Based on these permissions set for a specific Fabric Cloud, users are privileged to access a specific set of features of Fabric accordingly.

> ***Important:*** By default, all the roles have **Full Access** to Visualizer, Fabric Starter, and Cloud Build environments. For the remaining environments, all the roles have **No Access** by default.

> ***Note:*** The **Custom Access** permission is supported only for Fabric Clouds.

> ***Note:*** By default, all users have access to Fabric Console to create apps.

The following table details you about the features with access level permissions of an environment:

Environment Permissions - Custom Access		
Features in an Environment	Feature Level Access Permissions	
	Disable Access	Enable Access
Build Client App - If this feature is enabled, users can view the environment configuration from Visualizer and build a client app that connects to the environment. It allows a user to only have Build Client App access, if the user is only allowed to build binaries pointing to an environment, but not publish or make changes to the server from Visualizer or from Fabric console. This feature is very useful when developers need to build binaries pointing to a production server. However, only admins are allowed to modify the server runtime by allowing the publishing capabilities. ***Important:*** You can build apps using Visualizer if you have access permissions to the **Build Client App** feature in the Cloud. To build and publish apps, you must have access to the App Services feature in your cloud.		
App Services: If this feature is enabled, users can build apps using Visualizer and publish apps as well. This is a superset of Build Client App access and provides both the build and publish capabilities. • For example, If a developer has been given access to App Services, the developer can build and publish to the server from Visualizer or from Fabric console.		
Engagement Services - Access to the Engagement Server's admin console associated with this environment.		

Environment Permissions - Custom Access		
Features in an Environment	Feature Level Access Permissions	
	Disable Access	Enable Access
Logic Services - Access to the Integrate Node.js Services to Fabric Applications. • Refer to How to Integrate Node.js Services into Fabric Apps.	⊗	✓
Logging Console - Access to the logs for this environment from the Log Console. • Refer to Log Services.	⊗	✓

You can configure the **Environment Permissions**, as follows:

- **While inviting users to Cloud:** **Settings** > **Users** > **Manage Users** > **INVITE** > **Environment Permissions** > **FEATURE LEVEL ACCESS**. Refer to Invite User to an Account
- **While updating User Environments Access**: **Settings** > **Users** > **Manage Users** > **More Options** > **Manage Cloud Access**. This is applicable to the existing users. Refer to Managing Cloud Access of Existing User
- **While updating the environment users access: Environments** > **Clouds** > **More Options** > **Manage Access**. Managing Cloud Features in Environments

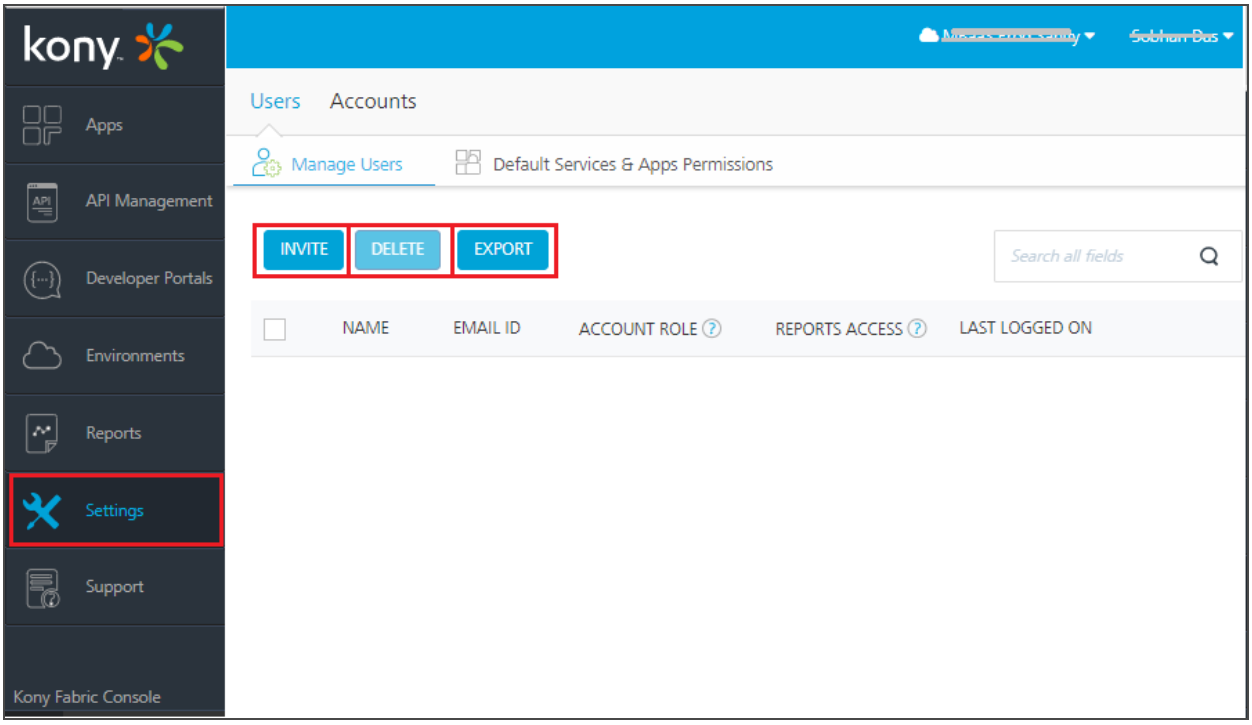
40.3 Users - Cloud

The **Users** tab in Kony Cloud helps you manage users in your account. Only users having an **owner/admin** role in the account can access this **Users** tab.

40.3.1 Manage Users

To manage users of an account, follow these steps:

1. In the Kony Fabric Console, click **Settings** in the left pane. By default, the **Users** > **Manage Users** page is displayed.

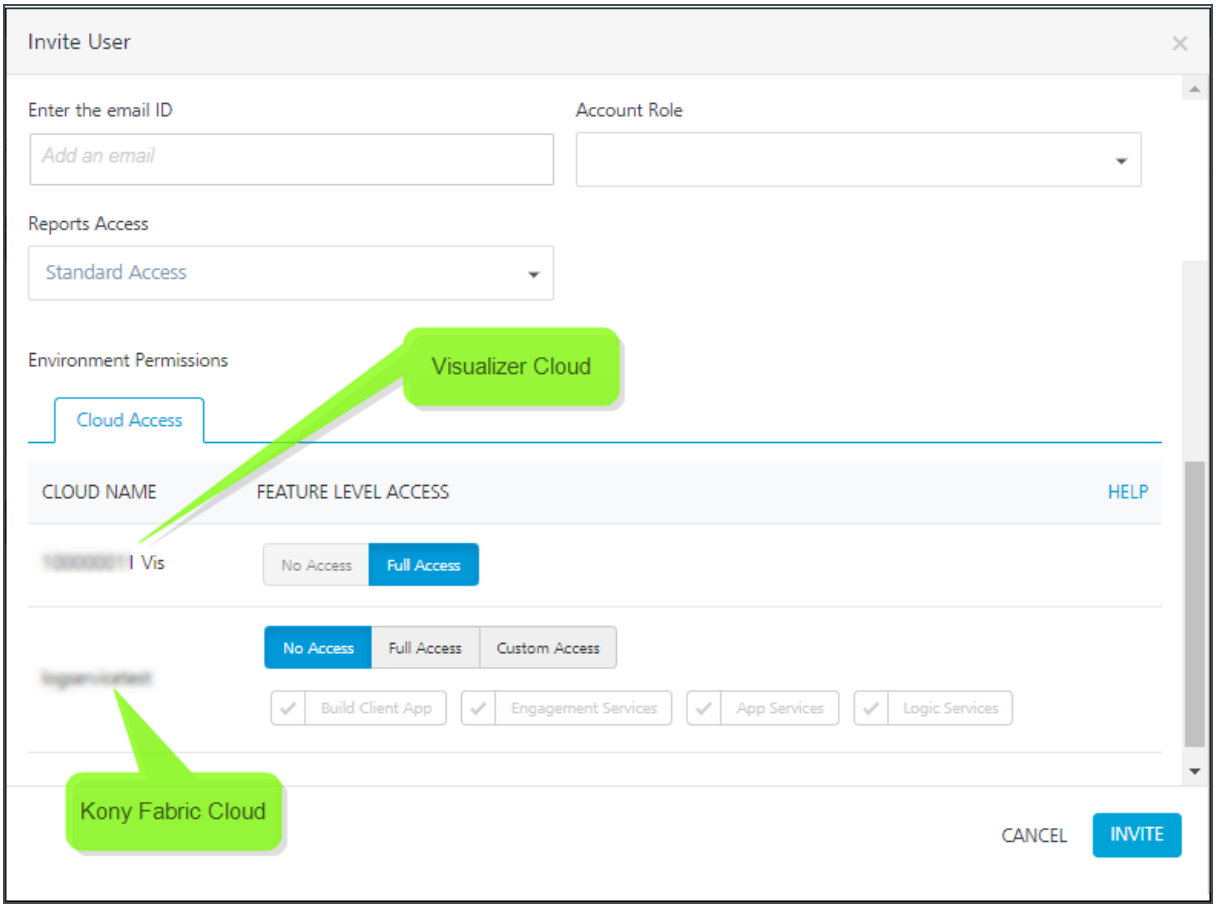


2. In the **Manage Users** page, you can perform various actions such as inviting a user, deleting a user(s), exporting users, and controlling user access.

40.3.1.1 Invite User to an Account

A user invited to an account can access reports and clouds in that account based on the role. Follow these steps to invite a user.

1. Go to Fabric Console, click **Settings** > **Users** > **Manage Users**.
2. Click **Invite**. The **Invite User** dialog window appears. You need to fill the required user details such as email, account role, Reports Access, and environment permissions.

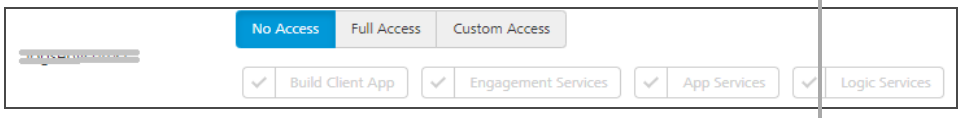
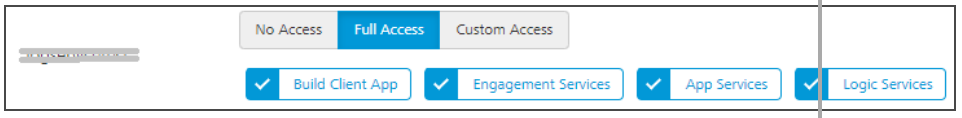


> ***Note:*** This is a sample screen shot, which has **Full Access** for a Visualizer Cloud. The Full Access is also applicable for the following clouds: **Licensing Cloud** and **AppFactory**.

3. Enter the following details in the **Invite User** dialog window:

Step	Field	Action
a	Enter the email ID	Enter the user email address
b	Account Role	Select the role available from the **Account Role** list. • **Admin/Owner:** By default, a user with an **Admin/Owner** role has **Full Access** to reports. For example, **Full Access** (Standard Reports, Dashboard, Funnel Reports, Custom Reports, Custom Metrics, Custom Dashboards) > ***Note:*** An invited user cannot modify the **Reports Access** field of an **Admin/Owner**. • **Member**: By default, a user with the **Member** role has **Standard Access** to reports. > ***Note:*** An **Owner/Admin** can change the **Report Access** permissions while inviting a Member role. • **Developer Portal Only**: By default, a user with the **Developer Portal Only** role has **Standard Access** permission to reports. > ***Note:*** A user with account role **Developer Portal Only** will have the Reports Access as Standard Access. You cannot modify the **Reports Access** field of the **Developer Portal Only** user.

Step	Field	Action
c	Reports Access	If you have selected **Member** as Account Role, select one of the following Report access permissions: • **Standard Access** (Standard Reports, Dashboard). This is selected by default. • **Custom Access** (Standard Reports, Dashboard, Custom Reports, Custom Dashboards) • **Full Access** (Standard Reports, Dashboard, Custom Reports, Custom Metrics, Custom Dashboards) ***Important:*** A user with Reports Access permissions as **Full Access** and **Custom Reports Access** can access these reports even if the user does not have access to any of the environments.

Step	Field	Action
d	Environment Permissions	Displays the clouds under the **CLOUD NAME** column created for a specific account. i. Under the **FEATURE LEVEL ACCESS** permissions column, click the required tab, for example, **No Access**, **Full Access**, and **Custom Access**. ***Note:*** For more information on the environment permissions, refer to Feature Level Access Permissions - Fabric Cloud. ***Note:*** The **Custom Access** permission is supported only for Kony Fabric clouds. ii. Select the required permissions for the Fabric Clouds: • **No Access** All features of this cloud are disabled for the selected user.  • **Full Access** Access to all features of this cloud is enabled for the selected user. Each of these features is enabled with a tick mark.  • **Custom Access -** Supports from Kony Fabric V9 onwards only for Kony Fabric Clouds. You can select all the features or specific ones of a cloud by using the **Custom Access** permission. For example: When you select **Custom Access**, all features of this cloud are enabled to the selected user by default. Each of these

4. Click **INVITE**. An invitation will be sent to the user. After the user accepts the invitation, the user is added to the users list.

40.3.1.2 Delete User from Account

Only users with the **Admin** role in the account can access the **Users** tab.

1. Go to Kony Fabric and click **Settings** > **Users** > **Manage Users**. The existing users are listed on the **Manage Users** page.
2. Select the check box for each user.
3. Click the **DELETE** button. The **Delete User(s)** dialog appears for your confirmation.

   > ***Note:*** Click the **More Options** button next to the user and select **Delete**.

4. Click **DELETE**.

40.3.1.3 Export Users of Account to Excel File

You can export the existing users of the account data to an Excel (.xls) file. The excel format contains users with record-level data in a tabular form such as the User Name, User Email, Account Role, Reports Access, and Last Logged ON.

1. Go to Kony Fabric, click **Settings** > **Users** > **Manage Users**. The existing users are listed on the **Manage Users** page.
2. To export all the existing users, click **EXPORT**.

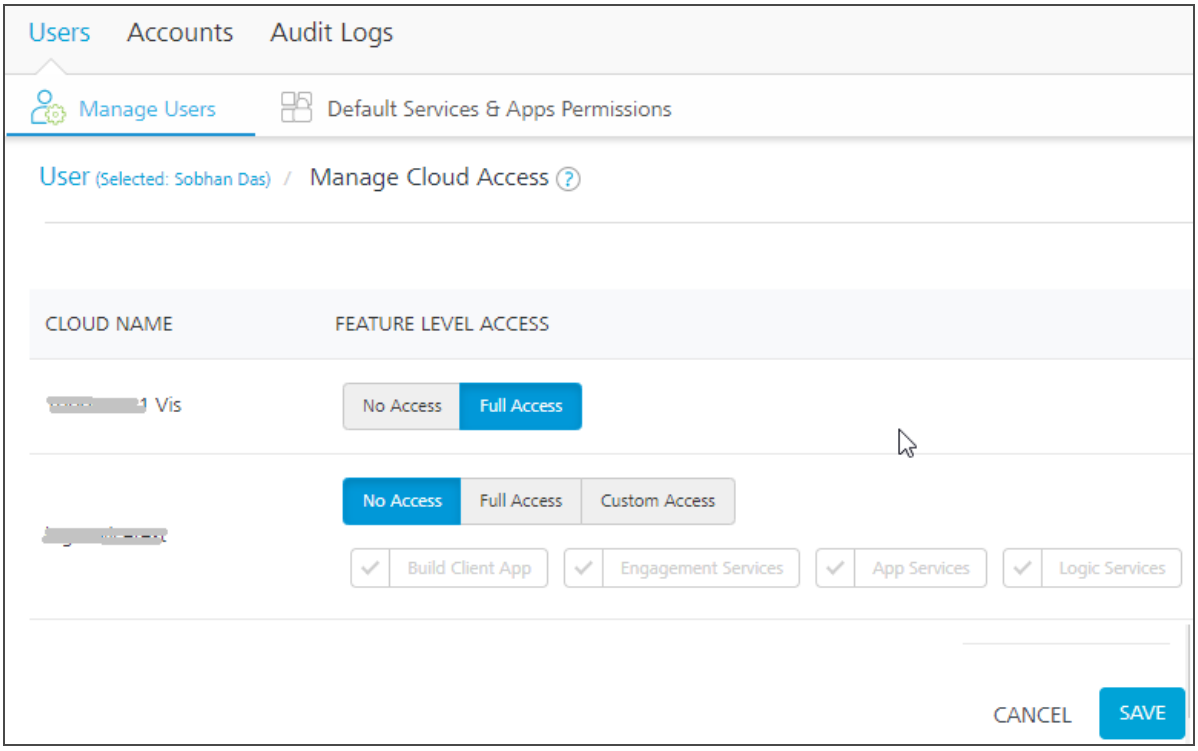
   > ***Note:*** To export one or more users, select the required check boxes and then click **EXPORT**.

40.3.1.4 Managing Cloud Access of Existing Users

You can modify the existing cloud access permissions of a user, if required.

1. Go to Kony Fabric, click **Settings** > **Users** > **Manage Users**. The existing users are listed on the **Manage Users** page.
2. Click the **More Options** button next to the user. The options available are as follows:
   - Delete
   - Managing Cloud Access
   - View User Activity
3. Select **Manage Cloud Access**. The **Manage Cloud Access** dialog box appears and displays the available clouds for that user.

   Managing Cloud Access of Existing User



4. Under the **Feature Level Access** column, select the feature permissions of the Clouds.
5. Under the **FEATURE LEVEL ACCESS** permissions column, click the required tab and select the features, for example, **No Access**, **Full Access**, and **Custom Access**.

> ***Note:*** For more information on the environment permissions, refer to Feature Level Access Permissions - Fabric Cloud.

6. Click **SAVE** to save the changes.

40.3.2 Default Services & Apps Permissions

As an Admin/Owner, you can configure the default access control for the new apps and services. So, when a new user is invited to an account or automatically added to an account using the master account setting, these default access control settings are applied to users. Refer to Default Services & Apps Permissions.

40.4 Accounts

The **Accounts** tab in the Settings menu helps you view the details of the account such as account name, location, company information, phone number, address and so on. It also allows you to configure **external-user authentication** and **multi-factor authentication** for the account.

40.4.1 Profile

Only users with the Owner role in the account can modify the account profile details.

All other users in the account can view account profile details in read-only mode.

40.4.2 External User Authentication

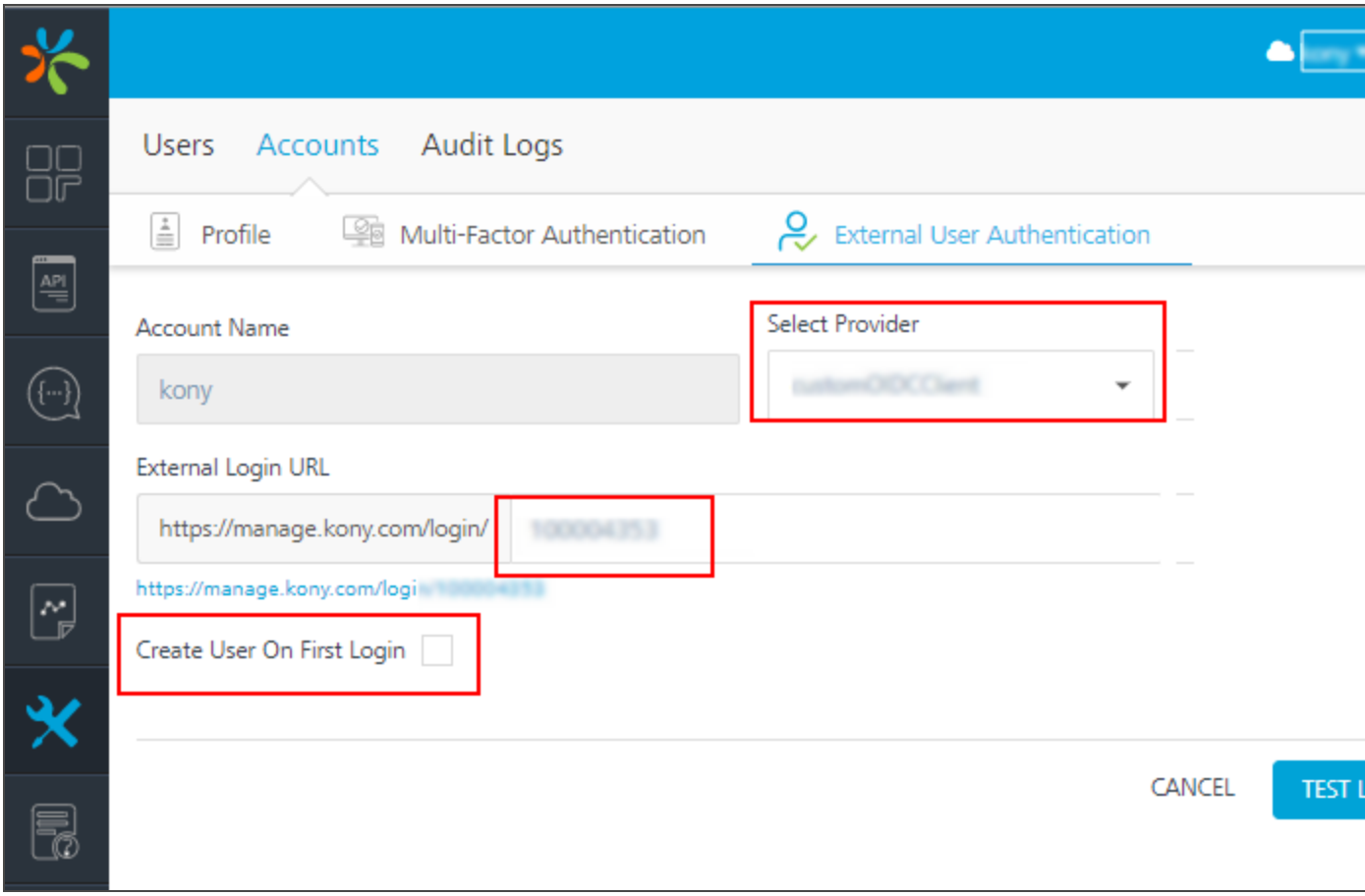
Only users with the Owner role in the account can access the **External User Authentication** tab. You can enable the external authentication with users to access the Kony Fabric console. OAuth compatible identity services are supported for external authentication.

To enable the external user authentication, follow these steps:

1. Create an OAuth based identity service in Kony Fabric.
2. As an owner, go to Kony Fabric > **Settings**.

3. In the **Accounts** tab, click the **External User Authentication**.
4. Click **ENABLE EXTERNAL USER AUTHENTICATION**.
5. In the Select Provider list, select an existing identity service that you have created and configure it.

> ***Note:*** OAuth compatible identity services will be displayed in the Select Provider list of the external authentication page.



6. Configure custom login URL, if required.
7. Select the **Create User on First Login** check box if you want the users to be created on the fly

upon the first login.

> ***Note:*** If you choose not to create the users on the fly, you will need to create them in Kony Fabric and in the IdP separately.

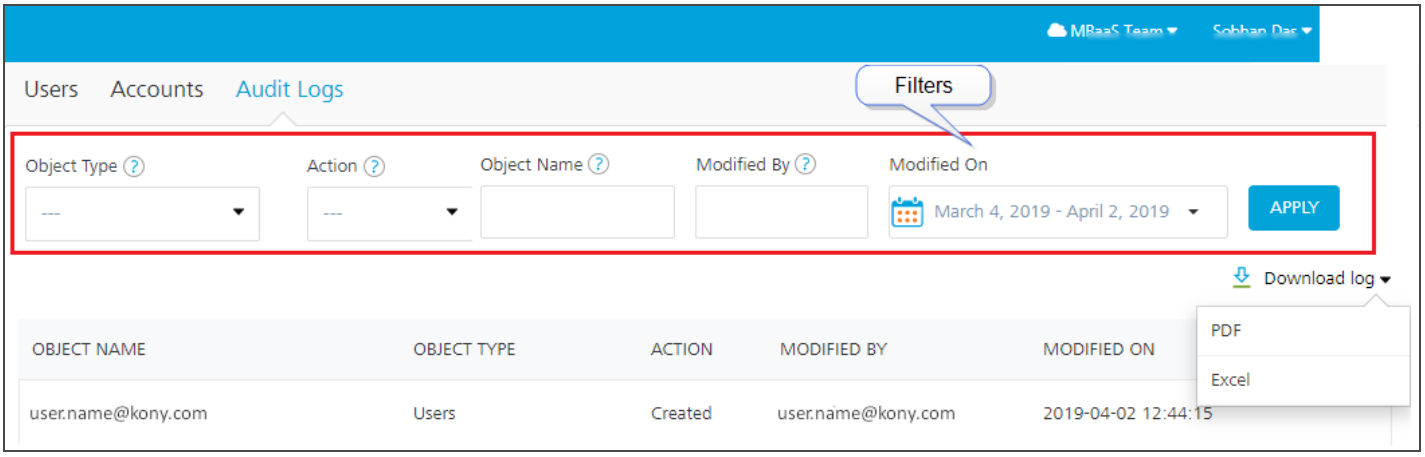
8. Click **SAVE**.

40.4.3 Multi-factor authentication

Only users with the Owner role in the account can access this tab. For more information on multi-factor authentication, refer to How to Enable Multi-Factor Authentication.

40.5 Audit Logs - Cloud

The **Audit Logs** tab in the **Settings** menu helps you to capture all the user activities performed in a Kony Fabric Account. These activities are displayed as Audit Logs. Only users having an **owner/admin** role in the account can access the **Audit Logs** tab.



To view the user activities of an account in the Audit Logs page, follow these steps.

1. In the Kony Fabric Console, navigate to **Settings** > **Audit Logs**. The Audit Logs page displays the existing logs by default.

These logs are displayed based on the object categories such as apps, services, and users. The logs are displayed with the following details:

- **OBJECT NAME**: Displays the name of the object, such as: app name, service, and email ID of the user.
- **OBJECT TYPE**: Displays the type of the object, such as: Apps, Services, and Users.
- **ACTION**: Displays the CRUD operations performed for the object type, such as: created, deleted, modified, published, and unpublished.
- **MODIFIED BY**: Displays the email ID of the user that performed the action.
- **MODIFIED ON**: Displays the time-stamp of when the action was performed.

> ***Note:*** By default, the logs are displayed for the last week.

To apply filters and view specific user activities of an account, follow these steps.

1. In the Kony Fabric Console, navigate to **Settings** > **Audit Logs**. The Audit Logs page displays the existing logs by default.
2. You can apply the following filters for the Audit Logs:
   - **Object Type**: From the drop-down list, select the options for which you want to view the audit logs.

     The drop-down list contains the following options:

     - Apps
     - Services
     - Users

- **Action**: From the drop-down list, select the options for which you want to view the audit logs.

  The drop-down list contains the following options:

  - Create
  - Modify
  - Delete
  - Publish
  - Unpublish

- **Object Name**: Enter the name of the object, such as the app name, service name or the email ID of the user.
- **Modified By**: Enter the email ID of the user that performed the actions.
- **Modified On**: From the drop-down list, select the date range for which you want to view the activities.

  The drop-down list contains the following options:

  - Today
  - Yesterday
  - Last 7 Days
  - Last 30 Days
  - This Month
  - Last Month
  - Custom Range

    For the **Custom Range** option, select the start date and the end date from the date range picker.

3. Click Apply to display the logs based on the filters.

To download the audit logs, click the **Download log** button and select the required option, such as PDF and Excel.

41. Settings - On-Premises

Using **Settings**, a superuser can manage tasks by configuring identity providers, adding users, importing new users and groups, assigning roles to users, deleting users and groups, configuring a proxy server, and configuring reports server.

The following are included under Settings:

- Users
- Proxy
- Reports

41.1 Users

Companies maintain a store of users and their details. Kony Fabric provides a mechanism to create a store of users in Kony Fabric Console (locally) or import from Active Directory.

Users listed in **Users** tab can access Kony Fabric Console to create apps. With the user information, an admin provides users access to Kony Fabric Console. Users are stored in accounts and are unique for each source or domain.

The Users tab contains the following sections:

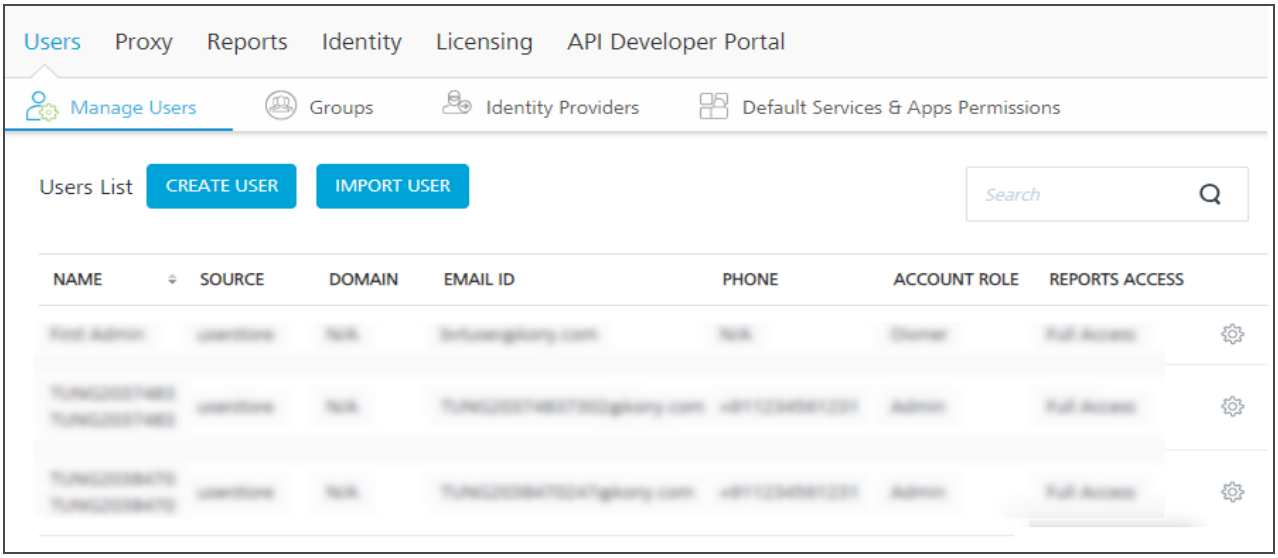
- Manage Users
- Groups
- Identity Providers
- Default Services & Apps Permissions

41.1.1 Manage Users

A user is an individual person. Each user needs an account to access Kony Fabric Console. A superuser creates user accounts for owners, admins, and members who use Kony Fabric Console.

The user module deals with creating users within Kony Fabric Console, importing users from external sources, editing user profiles, and assigning or unassigning environment access to users. Only a user with administrative privileges can access the user module. The user module also allows you to activate or deactivate user roles based on the requirements.

From the **Users** tab, click **Manage Users**. The **Manage Users** screen appears with the list of users. The list view displays a list of all the users along with other details. You can search the users based on each column and sort on each column.



The Users List view displays the following columns:

Column	Description
NAME	First and last name of the user.
SOURCE	Source that belongs to a user. If the user is a local user, the system displays as userstore.
DOMAIN	If the user is imported from Active Directory or created locally.
EMAIL ID	Email ID as received from Active Directory or as specified by an admin.
PHONE	Phone number as received from Active Directory or as specified by an admin.

Column	Description
ACCOUNT ROLE	Role as received from Active Directory or as specified by an admin.
REPORTS ACCESS	Access to reports is provided as follows: • **Standard Access** option provides access to Standard Reports and Dashboard. • **Custom Access** option provides access to Standard Reports, Dashboard, Custom Reports, Funnel Reports, and Custom Dashboard. • **Full Access** option provides access to Standard Reports, Dashboard, Custom Reports, Funnel Reports, and Custom Dashboard, and Custom Metrics.
Setting button	Allows you to edit and delete users, and change environment access to users.

You can navigate the list view through the **Previous** and **Next** buttons.

The **Manage Users** module in **Users** tab helps you search for a specific identity across Kony Fabric Console, and helps you assign and evaluate access controls against the identity. This module helps for editing role and profile information of users.

> ***Note:*** Special characters are supported in the import process. Names like O'Leary or Fernandez-Gutierrez are allowed.

> ***Important:*** As a user, you must be an admin or owner to access the **Manage Users** page and perform different tasks based on the role.

You can do the following tasks from the Users page:

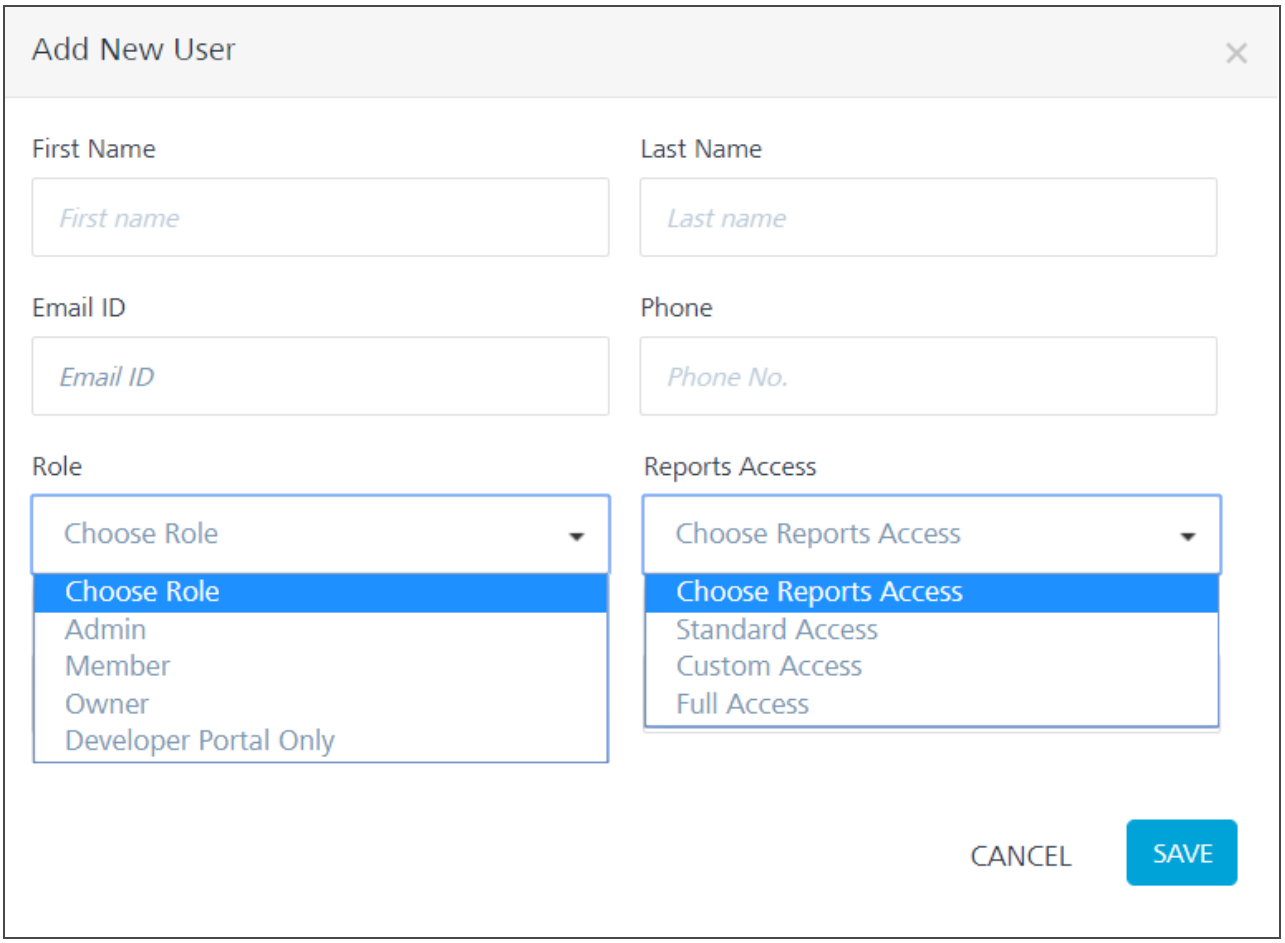
- Create a User in Kony Fabric Console
- Import Users from Active Directory
- Edit a User
- Change Environment Access to a User
- Delete a User

41.1.1.1 How to Create a User in Kony Fabric Console

To create a user, follow these steps:

1. On Kony Fabric, click **Settings**. By default, the **Manage Users** page appears. The **Manage Users** tab is only visible to users who are owners or admins. The page lists all owners, admins, and members of the account.

2. In the **Manage Users** page, click **Create User** button. The **Add New User** page appears.



3. Enter the following details.

> ***Note:*** The following fields, except Phone, are mandatory:

   a. **First Name**: First name of the user.

   b. **Last Name**: Last name of the user.

   c. **Email ID**: Email address of the user. The address should include alphanumeric and special characters a,nd should follow the standard email address form.

d. **Phone**: Phone number of the user. It should be numeric. This information is optional.

e. **Role**: Select the role from the list.

- **Owner**: An owner has the most privileges and can do the following:
  - Add, modify, and delete an environment.
  - Add, modify, and delete other owners, admins, and members.
- **Admin:** An admin has fewer privileges than an owner and can do the following:
  - Add other admins and members.
  - Modify and delete other admins and members.
  - Grant and deny environment access to other admins and members.
- **Member:** A member has the fewest privileges, including read-only access to the cloud given by other owners or admins.
- **Developer Portal Only**: By default, a **Developer Portal Only** role has **No Access** permission for reports access. For example, **No Access** (Standard Reports, Dashboard).

> ***Note:*** A user with account role **Developer Portal Only** will have the Reports Access as No Access. You cannot modify the reports access to the **Developer Portal Only** user.

f. If you have selected the **Member** as Account Role, select one of the following Report access permissions:

- **No Access** (Standard Reports, Dashboard)
- **Custom Reports Access** (Standard Reports, Dashboard, Custom Reports, Custom Dashboards)

- **Full Access** (Standard Reports, Dashboard, Custom Reports, Custom Metrics, Custom Dashboards)

> *Important:* A user with Reports Access permissions as **Full Access** and **Custom Reports Access** can access these reports even if the user does not have access to any of the environments.

g. **Password**: Enter the password for the user.

h. **Confirm Password**: Retype the password to ensure it is accurate.

4. Click **Save** to save the user details. The system will add the new user in the grid.

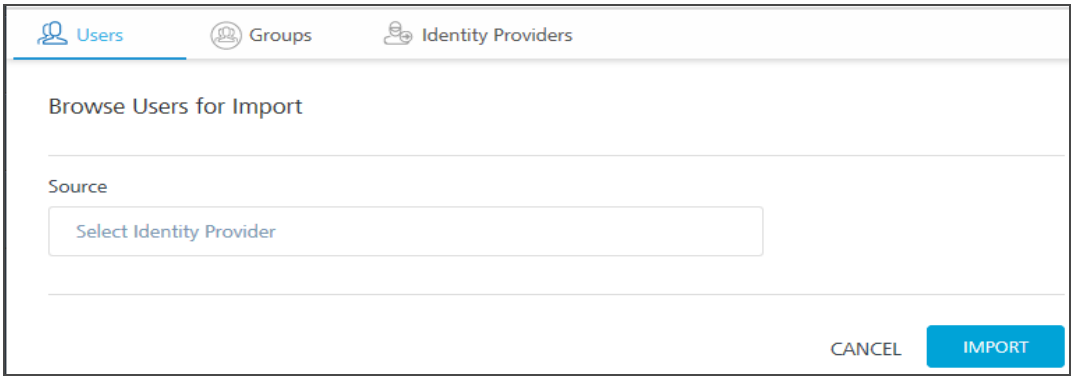
41.1.1.2 How to Import Users From Active Directory

You can add users to Kony Fabric Console database by importing them from Active Directories by using **Import Users** window. For more to configure Active Directory, refer to How to Configure Active Directory.

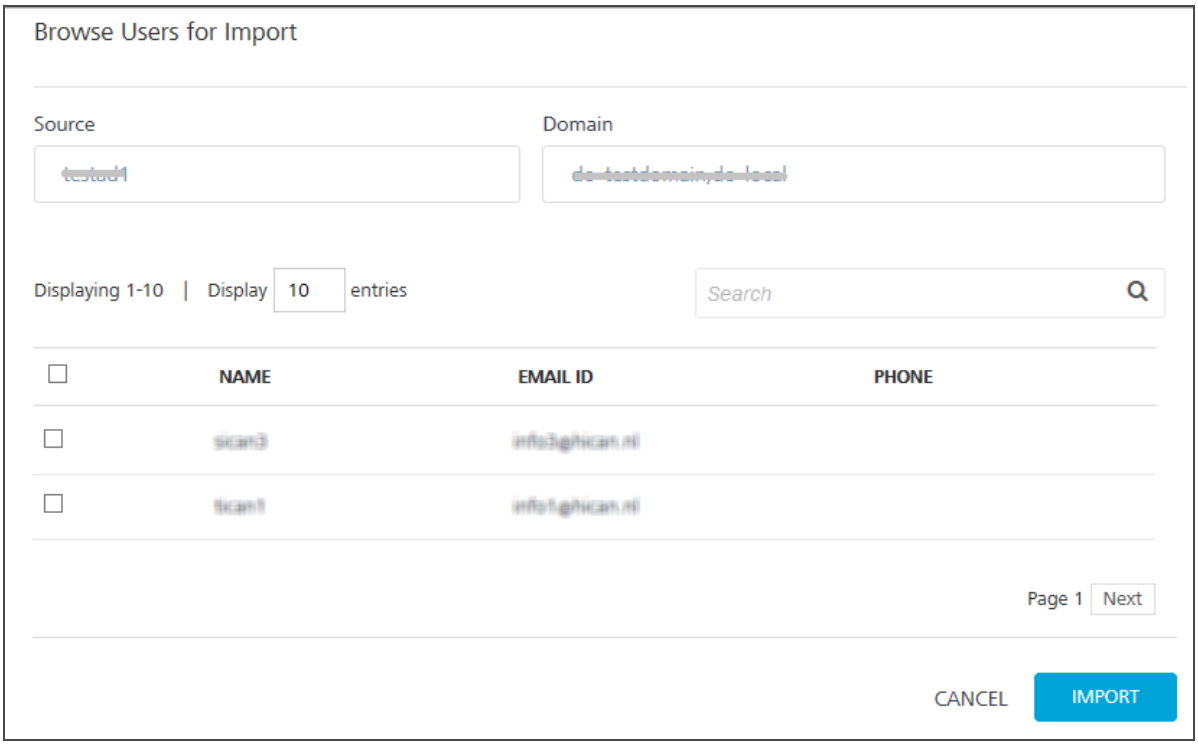
To import a user from Microsoft Active Directory, follow these steps:

1. To import a new user from Active Directory, click the **IMPORT USER** next to the Users List label at the top of the page.

   The **Browse Users from Import** window appears with the **Source** drop-down list. The Source drop-down list has all Active Directories configured in the Identity Providers.



2. Click in the **Source** list, and choose appropriate Active Directory. The system displays the **Domain** list includes a list of domains associated with Active Directory that you selected.



3. From the **Domain** list, choose the domain name. The following details of users for the selected domain appears:

- NAME
- EMAIL ID
- PHONE

You can search for users through the available search filters. Apply a single filter or combination of filters to define the search criteria and get the results.

Based on search criteria, the list view is updated with respective user details.You can navigate the list view using the **Previous** and **Next** buttons.

4. Select the required user or users through the check box next to the **NAME** listing. You can select the complete user list by selecting the check box next to the **NAME** column name.

5. Click the **IMPORT** button to import the users from Active Directory. The system displays the **Success** Window with a list of the updated users in the Users List view. The imported users from a source are identified under the **SOURCE** column.

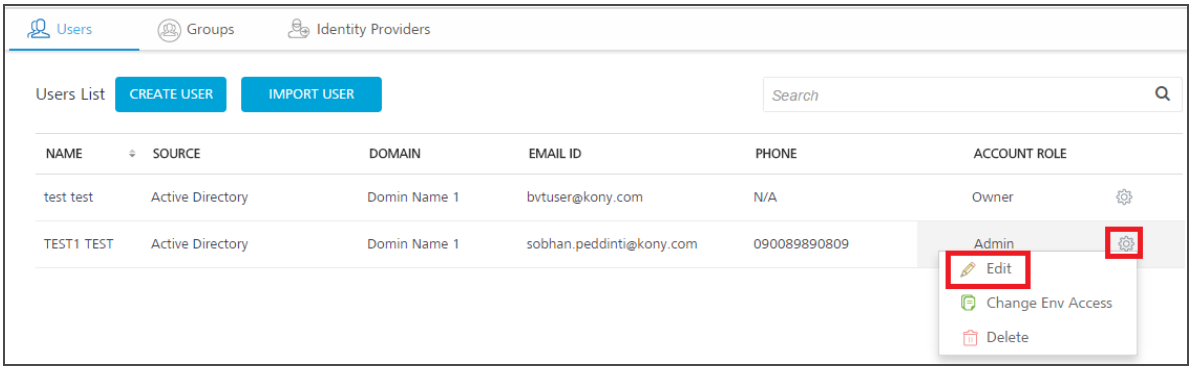
41.1.1.3 How to Edit a User

Based on role of the logged-in user, the user can make changes to admins, users, or owners.

> *Important:* An admin or owner can access the **Manage Users** page and perform different tasks.

To edit a user, follow these steps:

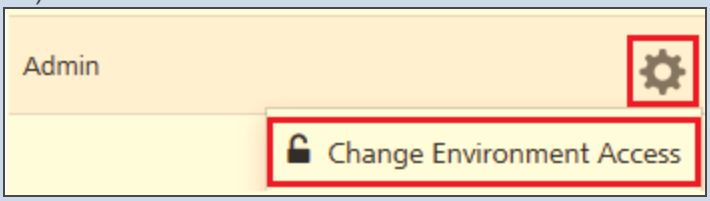
1. In **Settings > Users > Manage Users** page, hover your cursor over the required user from the list. Click the **Settings** button, and then click **Edit**.



Important: Based on user roles, the system allows users access to different settings. For example:

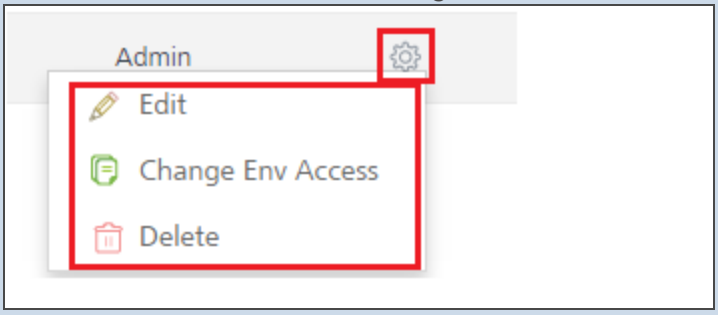
- If you log in as an admin or owner, you can change access to environments in the Change Environment Access page.

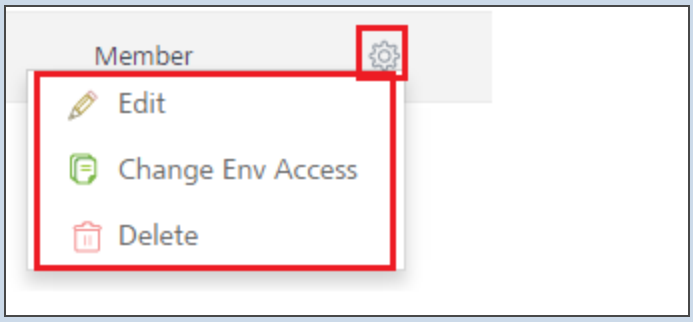
  In this case, an admin or owner makes changes for the logged-in admin or owner (self-user).



- If you log in as an admin, you can edit, change environment access, and delete other admins and members.

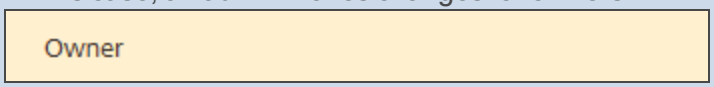
  In this case, an admin makes changes for other admins and members.



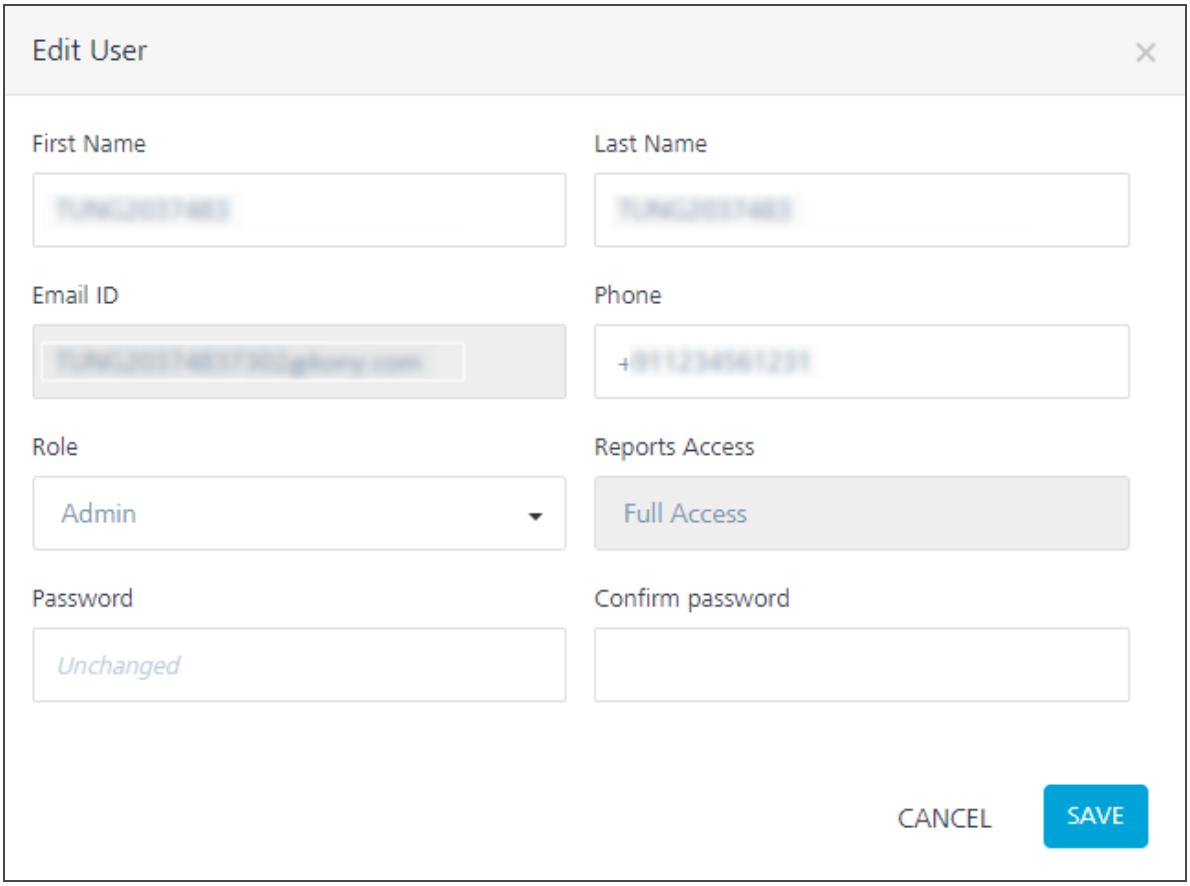


- If you log in as an admin, you cannot edit owners. The **Settings** button is not available in this case.

  In this case, an admin makes changes for owners.



The **Edit User** dialog appears.



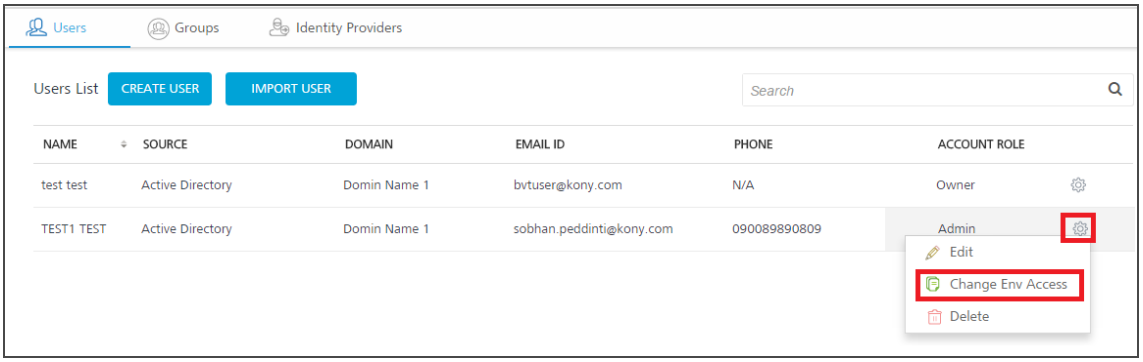
2. Make the necessary changes, if required. For example, you can change first name and last name, change email ID, change password, change phone number, change role of the user, and change reports access to the user.
3. Click **SAVE** to save the changes and close the dialog.

41.1.1.4 How to Change Environment Access to a User

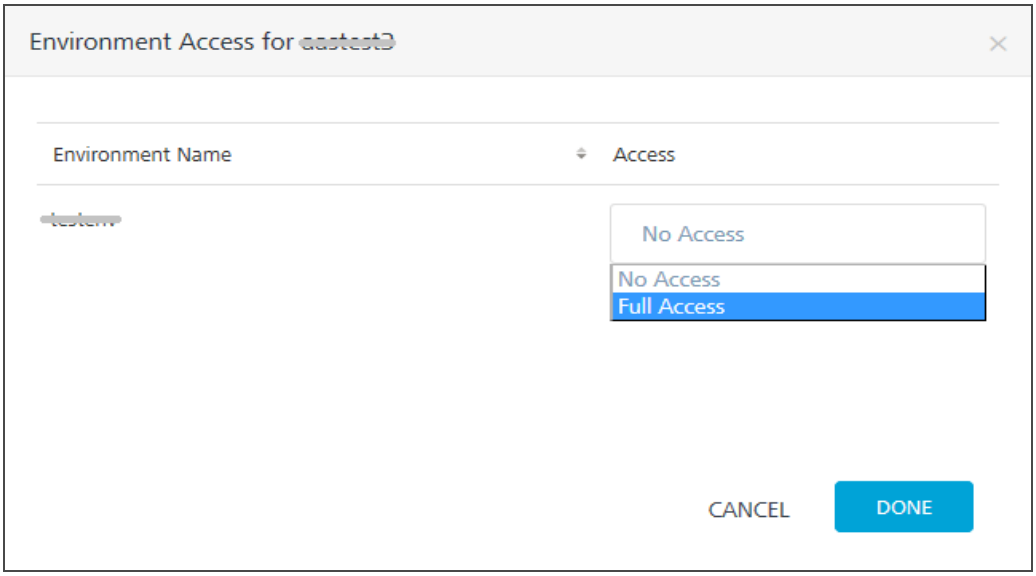
Users can be given full access or can be denied access to configured environments. An environment can contain all four servers such as server, messaging, sync, and management together or in different combinations. You can change the access for each user separately.

To change an environment access, follow these steps:

1. In **Settings > Users > Manage Users** page, hover your cursor over the required user from the list. Click the **Settings** button, and then click **Change Environment Access**.



The **Environment Access** page appears with all configured environments.



2. For an environment from the **Access** drop-down list, select the option.
   - **No Access**: indicates that users cannot access an environment.
   - **Full Access**: indicates that users can access an environment.
3. Click **Done** to close the page. The changes are applied to the user.

41.1.1.5 How to Delete a User

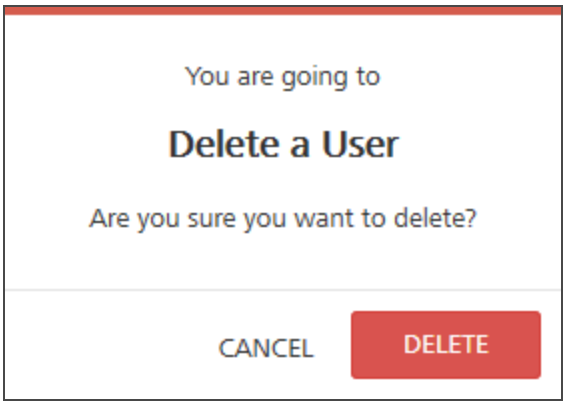
If you are a user (owner, admin, or member), you cannot delete yourself. A user with an admin or owner role can delete other admins and members. An admin cannot delete an owner.

> *Important:* As a user, you must be an admin or owner to get access to the Users page and perform different tasks based on the role.

To delete a user, follow these steps:

1. In **Settings > Users > Manage Users** page, hover your cursor over the required user from the list, click the **Settings** button, and then click **Delete**.

   The **Delete** confirmation page appears, shown below:

2. Click **Delete**. The system deletes the user from the Users grid.

41.1.1.6 How to Change your Kony Fabric User Password

If you are a user with member privileges (not an admin or owner,) you cannot change your password in the User List. However, you can change your password through your user profile.

To change your Kony Fabric user account password, follow these steps:

1. In your Kony Fabric account, click on your name in the upper-right corner of the screen.
2. From the drop-down menu, select **Profile**.

3. Click the **Set Password** tab.
4. In **Old Password**, type your old password.
5. In **New Password**, type your new password.
6. In **Confirm Password**, re-type your new password.
7. Click **Save**.

41.1.2 Groups

Groups are a set of similar users that provide a flexible way to define compound access rights. For example, a group combines entities such as roles and users. An administrator manages different types of user groups within an organization. Groups are useful when you have two entities to work on a common issue. For example, app developers and their subordinates can collaborate on an app issue by forming a group.

When a group is imported, all users of that group will also be imported. Users of subgroups within the groups will also be imported. For example, group Hyderabad consists of subgroups Hyd-Product and Hyd-Finance. If user James King is a part of Hyd-Finance only, he will get imported.

> ***Important:*** Active Directory groups cannot be edited. No local user can be added as part of Active Directory group.

From the **Users** tab, click **Groups**. The Groups screen appears with the list of groups. The Group list view displays a list of all the groups along with other details. You can search the groups based on each column and sort on each column.

You can navigate the list view through the **Previous** and **Next** buttons.

The Groups list view displays the following columns:

Columns	Description
GROUP NAME	Displays the name of the group.

Columns	Description
SOURCE	Displays if the Group is imported from Active Directory or created Locally.
DOMAIN	Displays domain that belongs to Group.
DESCRIPTION	Description of the Group detailing features and functionality.

You can perform the following activities from the **Groups** page:

- How to Import a Group from a Source

41.1.2.1 How to Import a Group from a Source

You can also add groups to the Kony Fabric database by importing them from Active Directory, using the **Browse Groups for Import** window.

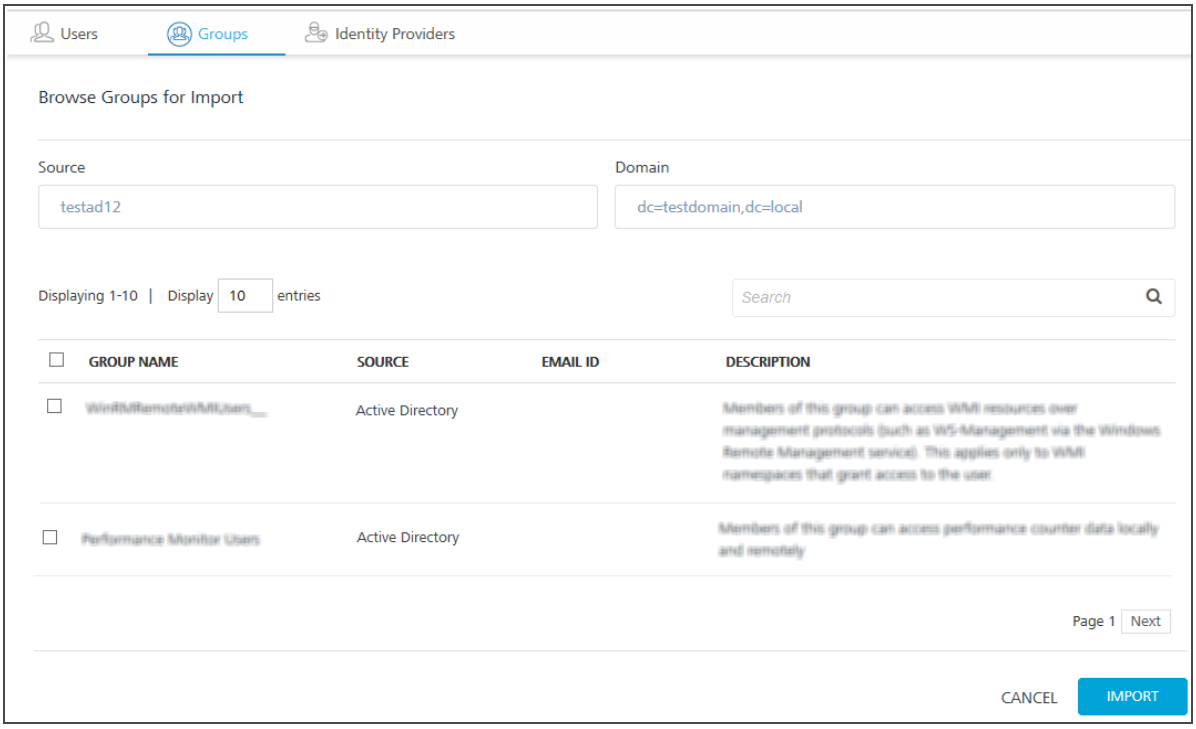
To import a group from Active Directory, follow these steps:

1. In the Groups tab, click the **IMPORT GROUP** next to the Group List label at the top of the page.

   The **Browse Groups for Import** screen appears with the **Source** drop-down list. The Source drop-down list has all Active Directories configured in the Identity Providers.

2. From the **Source** list, choose the appropriate Active Directory. The system displays the **Domain** list, a list of domains associated with the selected Active Directory.

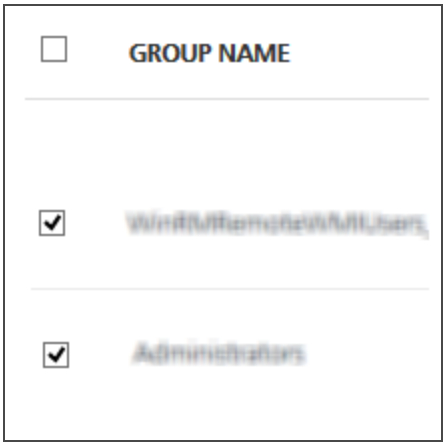
3. From the **Domain** list, choose the domain name. The Group details from the selected domain appear in the grid.



You can search for the groups through the available search filters. Apply a single filter or combination of search filters to define the search criteria and get the result. For example, enter the partial or complete name of the group in the **Search Groups** field.

Based on the search criteria, the list view is updated with the respective group details. You can navigate the list view using the **Previous** and **Next** buttons.

4. Select the required group or groups through the check box next to the **Group Name** listing. You can select the complete user list by selecting the check box next to the **Group Name** column name.
   When you select the group or groups, the **IMPORT** button becomes active.



5. Click the **IMPORT** button to import the groups from Active Directory. The system displays the **Success** window with a list of updated users. The selected groups are copied to Kony Fabric database and displayed in the **Groups List** page.

   The following table provides details about the groups:

Properties	Description
GROUP NAME	Along with the groups, all users that are part of a group and part of any subgroups are individually imported into Kony Fabric database. • The subgroup itself is not imported and its details are not captured as a group. • For example, Group X includes a subgroup named Y. Group X has users: A, B, C, D. • Sub Group Y has users: A, F, G. • When Group X is imported, all six Users {A, B, C, D, F, G} are imported to Kony Fabric. Subgroup Y is not imported. • If Subgroup Y is imported, then only users A, F, and G are imported because Group Y is a subgroup. • Once the groups are added, any apps can be targeted to them.

41.1.3 Identity Providers

You can configure multiple identity providers and import users from different domains into Kony Fabric Console.

Once you have logged into Kony Fabric Console, from the left pane, click the Settings. In the **Users** page, click the **Identity Providers** tab. The Identity Providers page appears with a list of Active Directories and SiteMinder configured within Kony Fabric Console. You can search and sort Active Directories based on each column.

The Identity Providers list view displays the following columns:

Column	Description
Domain	List of Active Directory domains.
Provider Type	Directory type of Active Directory.
Host or IP Address	List of host names or IP addresses.
Port	Port numbers of Active Directory Servers.
Created By	Name of the administrator who configured Active Directory Servers.
Created On	Date and time details of when Active Directory Servers are configured.
Information Icon 	Number of users and groups imported from Active Directory when you click on the information icon. If no users or groups are imported from a directory, the information icon turns into a check box. To delete Active Directory, select the desired check box and then click the **Delete** button.

Column	Description
Delete Button	Deletes selected Active Directory from the database.The **Delete** button dims because it is not available until a check boxes is selected.

You can navigate the list view through the **Previous** and **Next** buttons.

The following external identity provider types are supported:

- Microsoft Active Directory
- SiteMinder

Microsoft Active Directory

Kony Fabric supports importing users from Active Directory. Active Directory is a centralized and standardized system that automates network management of user data.

> ***Important:*** As an administrator, you must have the appropriate permissions to configure multiple Active Directory instances.

These upcoming sections will help you learn more about managing your network resources:

- How to Configure Active Directory
- How to Update Active Directory Configuration
- How to View Users of Active Directory
- How to View Groups of Active Directory
- How to Delete Active Directory Configuration

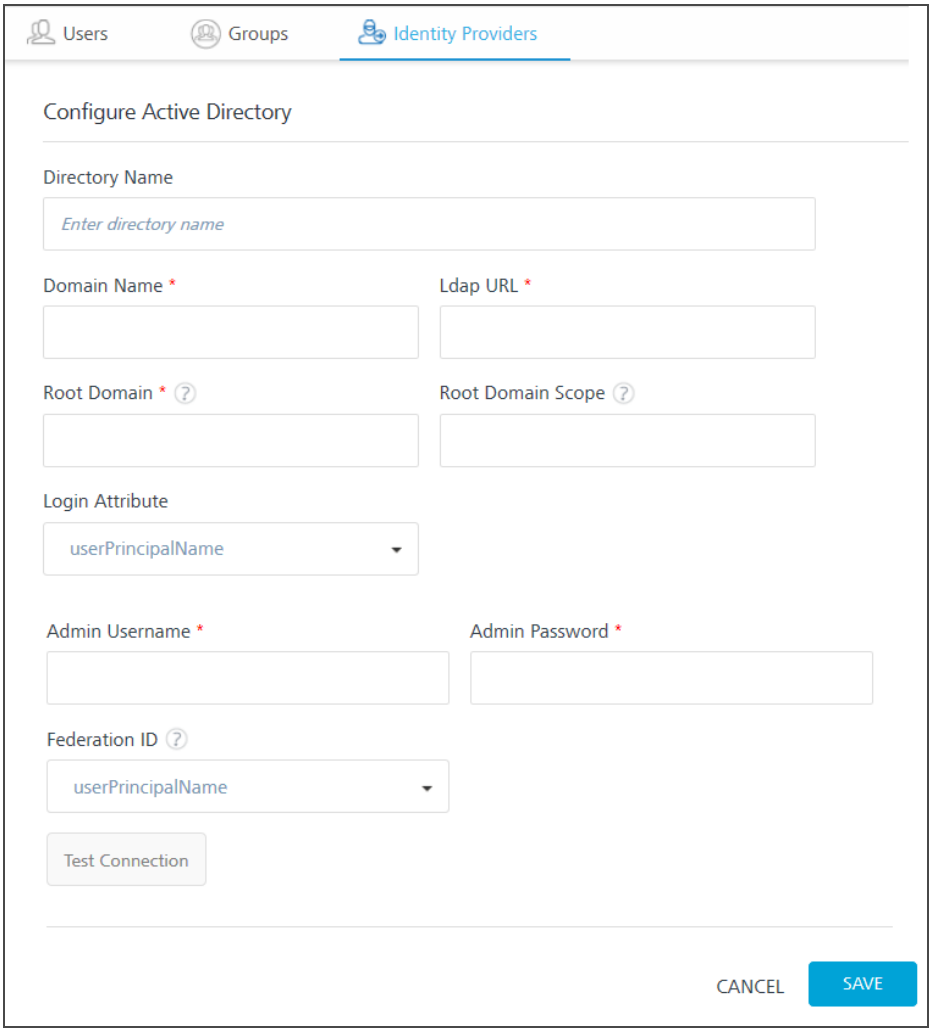
41.1.3.1 How to Configure Active Directory

The Configure Directory page helps configure communication between Kony Fabric Console database and Active Directory. Kony Fabric Console uses a database to fetch employee details, to provide user authentication, and to update and synchronize users.

To configure Active Directory, follow these steps:

1. After you sign in to the Kony Fabric Console, from the left pane, click **Settings**, and click **Users > Identity Providers**. The **Identity Providers** page appears.
2. Click the **CONFIGURE** drop-down list, and choose **Active Directory**.

   The **Configure Active Directory** page appears.



3. In the **Directory Name** text box, enter the appropriate name for Active Directory.
4. In the **Domain Name** text box, enter the domain name of Active Directory.

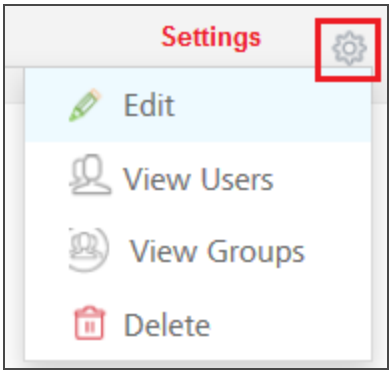
5. In the **Ldap URL** text box, enter the domain name of Active Directory.
6. In the **Root Domain** text box, enter the root domain name of Active Directory.
7. In the **Root Domain Scope** text box, enter the domain name of Active Directory. If root domain scope is not defined, the **Root Domain Scope** field will be defaulted to root domain.
8. In the **Login Attribute** text box, select one of the attributes from the drop-down list to search Active Directory.
9. In the **Admin Username** text box, enter the admin name that is used to access Active Directory Server.
10. In the **Admin Password** text box, enter the admin password that is used to access Active Directory Server.
11. In the **Federation ID** text box, enter the unique identifier of Active Directory.
12. Click the **Test Connection** button. If the connection is established, a confirmation message appears.
13. Click the **SAVE** button. The following save status message appears.
14. Click **OK** to return to the main page.

41.1.3.2 How to Update Active Directory Configuration

You may need to update Active Directory settings for specific reasons. For example, you may need to update a port number.

To Update active directory configuration, follow these steps:

1. From the **Identity Providers** page, hover your cursor over the required Active Directory service from the list, click the **Settings** button, and then click **Edit**.

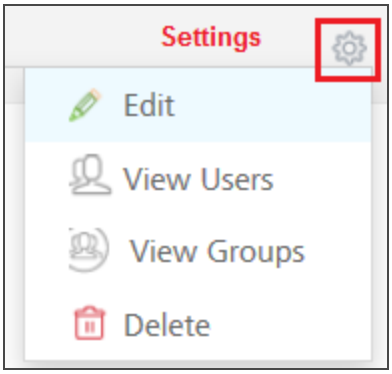


2. The **Edit Directory** page appears. The desired fields can be updated. There are no restrictions. Click **Save** to save changes to Active Directory.

41.1.3.3 How to View Users of Active Directory

To view users of Active Directory, follow these steps:

1. From the **Identity Providers** page, hover your cursor over the required Active Directory service from the list, click the **Settings** button.
2. Click **View Users** to display the **Users List** page.



41.1.3.4 How to View Groups of Active Directory

To view groups of Active Directory, follow these steps:

1. From the **Identity Providers** page, hover your cursor over the required Active Directory service from the list, click the **Settings** button.
2. Click **View Groups** to display the **Groups List** page.

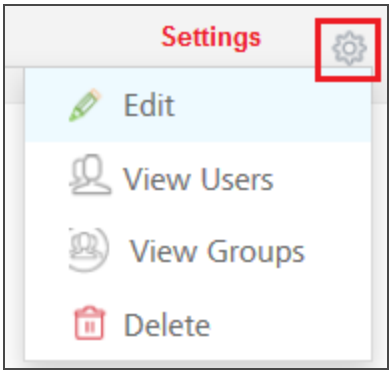
41.1.3.5 How to Delete Active Directory Configuration

To delete a directory, users and groups of the directory should be deleted. An admin can select and delete one or more entries in the Directory List. Once the directories are deleted, the entries do not appear in the Directory List.

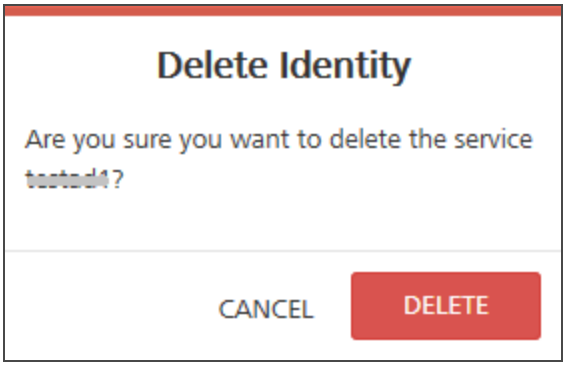
> ***Note:*** When no users and groups are imported from selected Active Directory, the Information icon turns into a check box.

To delete a directory, follow these steps:

1. From the **Identity Providers** page, hover your cursor over required Active Directory service from the list, click the **Settings** button, and then click **Delete**.



The system displays Delete Directory Confirmation Message: *"Are you sure you want to delete the service <AD name>?"*



2. Click **DELETE** to confirm the deletion.The directory is removed from the **Identity Providers** list.

SiteMinder

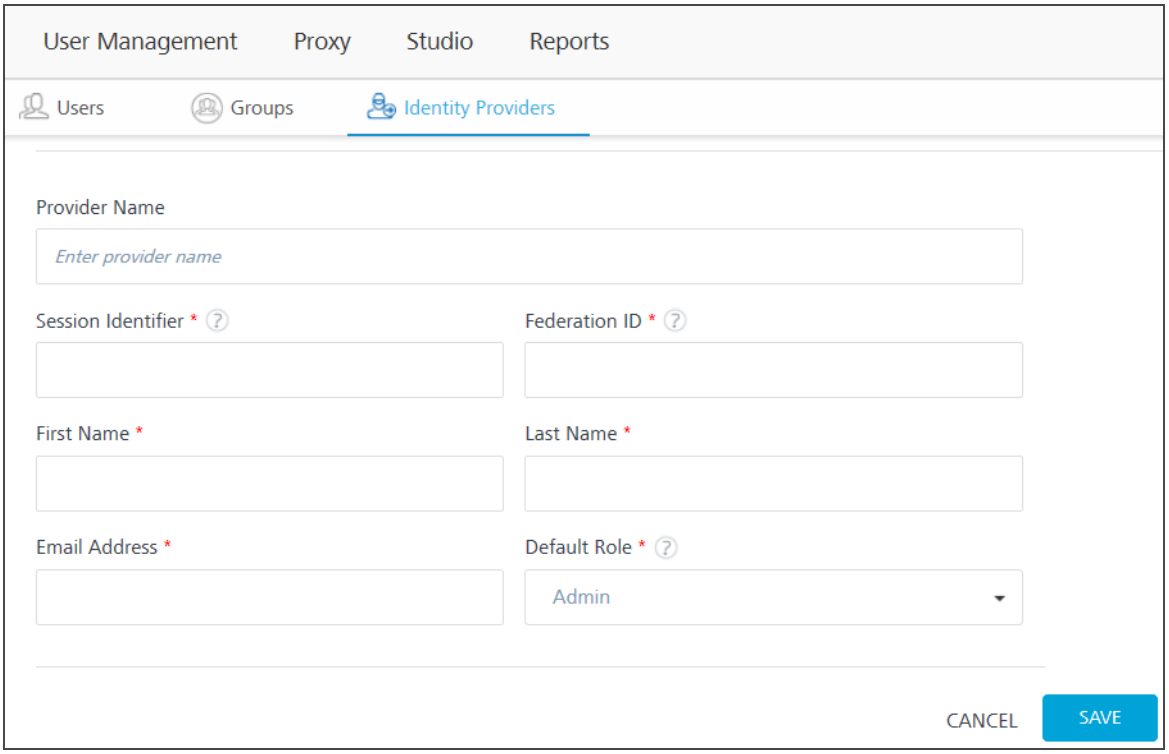
A centralized Web access management system, SiteMinder enables user authentication and single sign-on, policy-based authorization, and identity federation. SiteMinder also audits access to Web applications and portals.

After a developer configures a SiteMinder in Kony Fabric Console, the SiteMinder instance handles authentication and authorization of users. When a SiteMinder user logs in Kony Fabric Console, the system directs the user to the SiteMinder log-in page. After a SiteMinder user logs in to the SiteMinder instance, the user will be redirected to Kony Fabric Console. Kony Fabric auth service will verify whether SiteMinder headers and cookies exist before giving access to Kony Fabric design time components. Kony Fabric components such as Kony Fabric Console, Kony Fabric Engagement, Kony Fabric Sync, and Kony Fabric Integration Server.

To configure SiteMinder in the console, follow these steps:

1. Sign in to the Kony Fabric Console. From the left pane, in **Settings > Users**, click **Identity Providers**. The **Identity Providers** page appears.
2. Click the **CONFIGURE** drop-down list, and choose **SiteMinder**.

The **SiteMinder configuration** page appears.



3. In the **Provider Name** text box, enter the appropriate name for the SiteMinder.
4. In the **Session Identifier** text box, enter the cookie name.
   SiteMinder uses the cookie name to send the session ID that identifies the logged-in user session.
5. In the **Federation ID** text box, enter the federation ID.
   SiteMinder uses the identifier to uniquely identify a user independent of a session.
6. In the **First Name** text box, enter the appropriate first name of the SiteMinder user.
7. In the **Last Name** text box, enter the appropriate last name of the SiteMinder user.
8. In the **Email Address** text box, enter the email address of the SiteMinder user.
9. In the **Default Role** text box, select the role from the drop-down list.

By default, any authenticated SiteMinder user is mapped to the default role.

10. Click the **SAVE** button. The configured SiteMinder appears in the Identity Providers list view.

41.1.4 Default Services & Apps Permissions

As an Admin/Owner, you can configure the default access control for the new apps and services. So, when a new user is invited to an account or automatically added to an account using the master account setting, these default access control settings are applied to users. Refer to Default Services & Apps Permissions.

41.2 Proxy

With proxy, you can enable more security to your apps. Typically, you use the proxy server to filter web content, and monitor uploads and downloads on the Internet. When connecting to the Internet through proxies, the IP address of the proxy server will be shown instead of your machine's.

> ***Important:*** As a user, you must be an admin or owner to access the **Proxy** page and perform different tasks based on the role.

You can do the following tasks from the Proxy page:

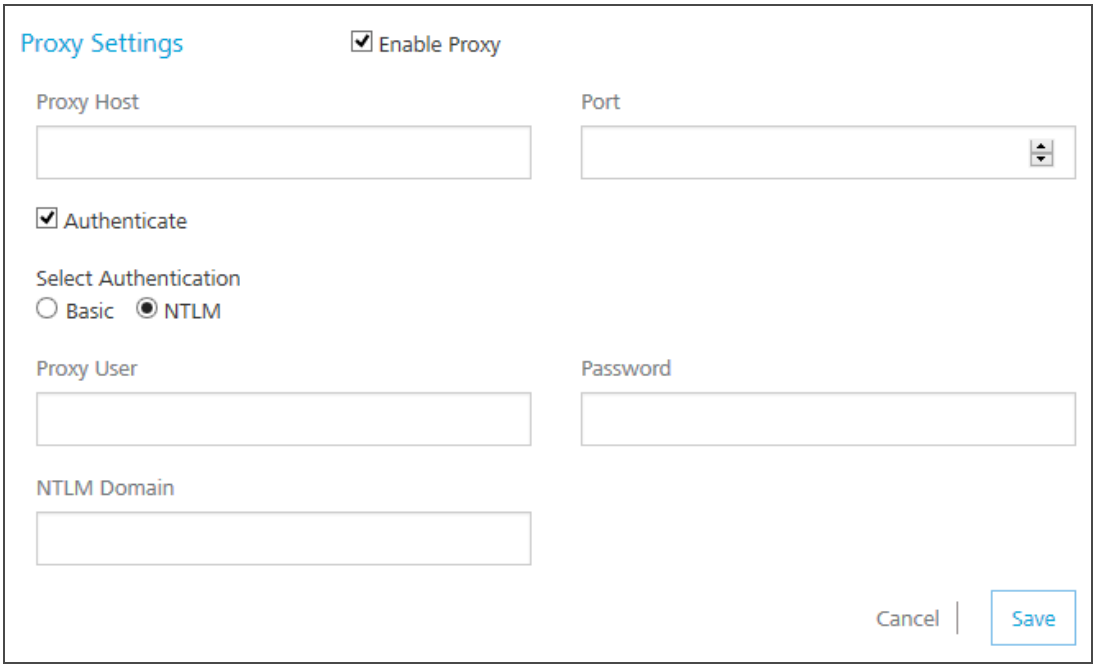
- How to Configure a Proxy
- How to Enable a Proxy to an Integration Service
- How to Delete a Proxy

41.2.1 How to Configure a Proxy

You can configure only one proxy server. A proxy server can require basic or NT LAN Manager (NTLM) authentication. If you have NTLM authentication configured as your proxy, you need to provide additional information such as proxy user, password, and NTLM domain.

To configure a basic or NTLM proxy, follow these steps:

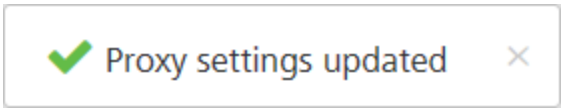
1. On Kony Fabric, click **Settings**. By default, the **Manage Users** page appears.
2. Select **Proxy** to open the **Proxy Settings**.
3. Select the **Enable Proxy** check box.
4. In the **Proxy Host** text box, enter the IP of the server.



5. From the **Port** text box, enter the port number, which can range from 1 to 65535.
6. To enable authentication for your proxy, select the **Authenticate** check box, and follow these steps. Otherwise skip to Step 6.
    a. Under **Select Authentication**, select **Basic** or **NTLM**.
    b. For Basic authentication, do the following:
        - In the **Proxy User** text box, enter the user for the proxy.
        - In the **Password** text box, enter the password for the proxy.

a. For NTLM authentication, follow these steps to add the required configurations to Kony Studio:
   - In the **Proxy User** text box, enter the user for the proxy.
   - In the **Password** text box, enter the password for the proxy.
   - In the **NTLM Domain** text box, enter the domain for the proxy.

7. Click **Save** to the save the proxy. The confirmation message appears.



41.2.2 How to Enable a Proxy to an Integration Service

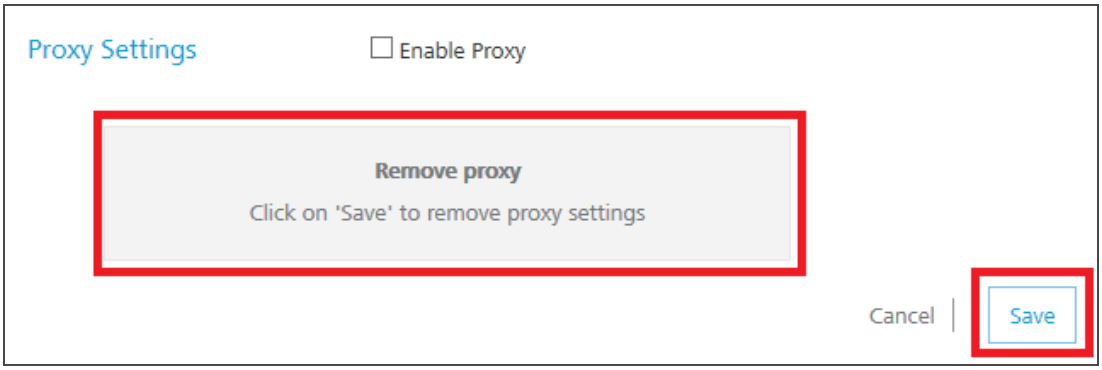
Once a proxy is configured, you can enable the proxy for an integration service. For more details, refer to Integration Services.

41.2.3 How to Delete a Proxy

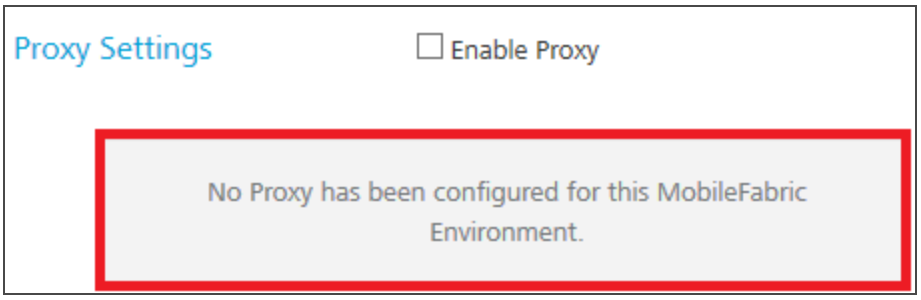
To delete a proxy server, follow these steps:

1. From the **Proxy Settings** list, clear the **Enable Proxy** check box.

The system displays the confirmation message `Remove proxy: Click on Save to remove proxy settings`, shown below:



2. Click **Save**. The system deletes the proxy and displays the confirmation message, shown below:



41.3 Reports

In the **Reports** tab, you can configure the JasperReports Server. After you complete JasperReports Server configuration, the following **Reports** page displays data (reports) from the JasperReports Server.

41.3.1 How to Configure the JasperReports Server

To configure the JasperReports Server, follow these steps:

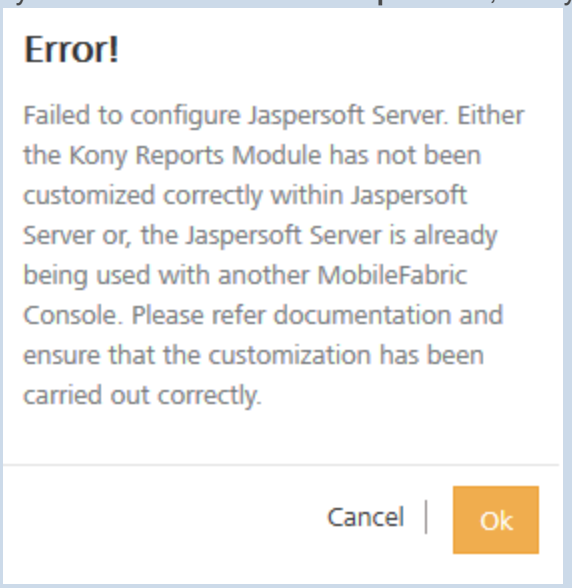
> ***Important:*** Before configuring the JasperReports Server in the Reports tab, ensure that you have installed the JasperReports Server and configured Kony Fabric Console in the JasperReports Server.
>
> For more details about how to set up the JasperReports Server, refer to Kony Analytics and Reporting Installation Guide.

1. From the **Settings** page, select **Reports**. The **Reports** tab appears.
2. In the **Jasper URL** text box, enter the JasperReports Server URL.
3. In the **Username** text box, type `jasperadmin`.

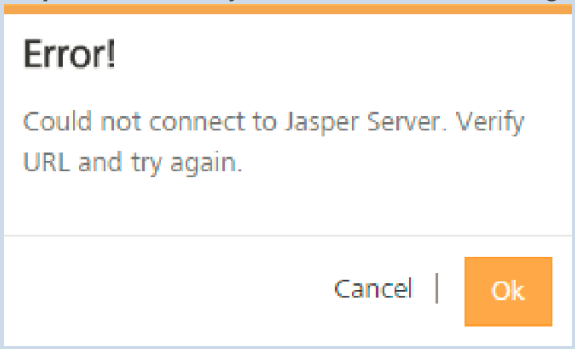
   > ***Note:*** Enter credentials for jasperadmin. The default credentials for jasper admin are:
   > username = jasperadmin
   > password = jasperadmin

4. In the **Password** box, type `jasperadmin`.
5. Click **Save** to the save the JasperReports Server. The following confirmation message appears.

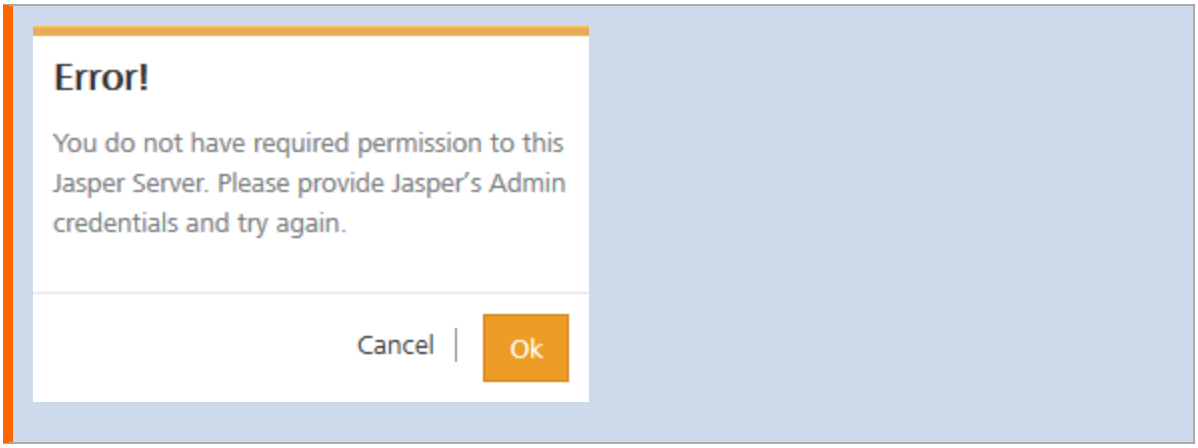
Important: If Kony Fabric Console is not configured in the JasperReports Server, and you try to save the details in the **Reports** tab, the system throws the following error:



If you enter a wrong URL for JasperReports Server, and you try to save the details in the **Reports** tab, the system throws the following error:



When you save the details in the **Reports** tab by providing wrong admin username and password of the JasperReports Server, the system throws the following error:



After you configured JasperReports Server successfully, you can access the reports from **Reports** page.

42. Reports

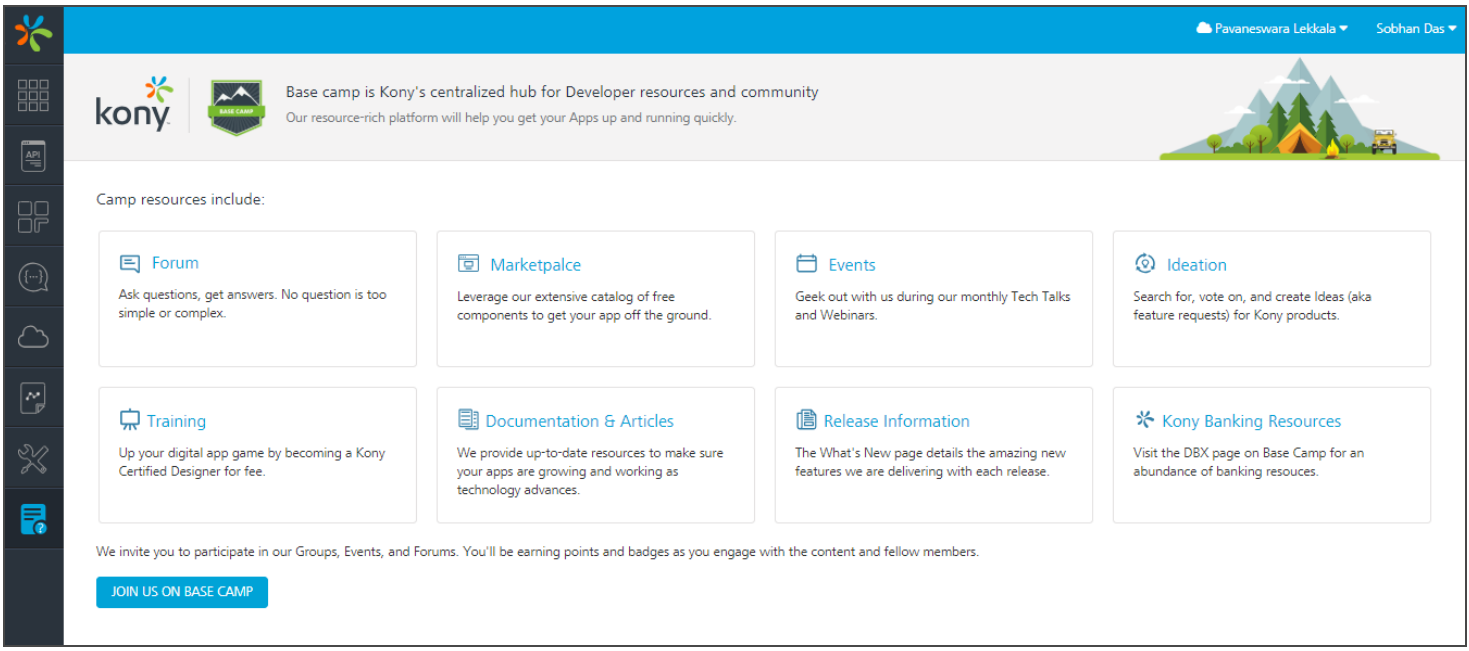
The **Reports** page displays standard reports, dashboard, funnel reports, custom reports, custom metrics, and custom dashboards. To view reports, click **Reports** from the left pane.

> *Important:* **For Kony On-Premise only**: To access the reports, you must configure the JasperReporting Server by clicking the **Configure Reporting Server** button. For more details on how to configure the reporting server, click here.

- For information on standard reports and dashboard, refer to Standard Reports and Dashboard documentation.
- For information on funnel reports, refer to Funnel Reports documentation.
- For information on custom reporting, refer to Custom Reporting - Metrics, Reports, and Dashboard documentation.

43. Kony Support

The **Support** page displays links to the latest tutorials and articles and Developer resources from Kony Base Camp Library. Base Camp is Kony's centralized hub for developer resources and community. Our resource-rich platform will help you get your Apps up running quickly.



The following are the Developer resources from Kony Base Camp Library:

- Forum - Ask questions, get answers.
- Marketplace - Free! Yes, we said free components to pop into your apps.
- Events - Geek out with us during our monthly Tech Talks and Webinars.
- Ideation - Search for, vote on, and create Ideas (feature requests) for Kony products.
- Training - Up your game by becoming a Kony Certified Designer for free.
- Documentation & Articles & Announcements - Resources the way you want them.

- Release Information - Do not miss a beat when we release new features into the wild.
- Kony Banking Customers - Visit the DBX page on Base Camp for banking resources.

You can join the Base Camp community to participate in Groups, Events, and Forums. You'll be earning points and badges as you engage with the content and fellow members. To join Kony Base Camp, click **JOIN US ON BASE CAMP**.

44. Telemetry

Telemetry provides a mechanism to the administrators of the Kony Fabric portal to easily upload information about their setup while filing a support ticket. The feature aids a quick resolution of an issue.

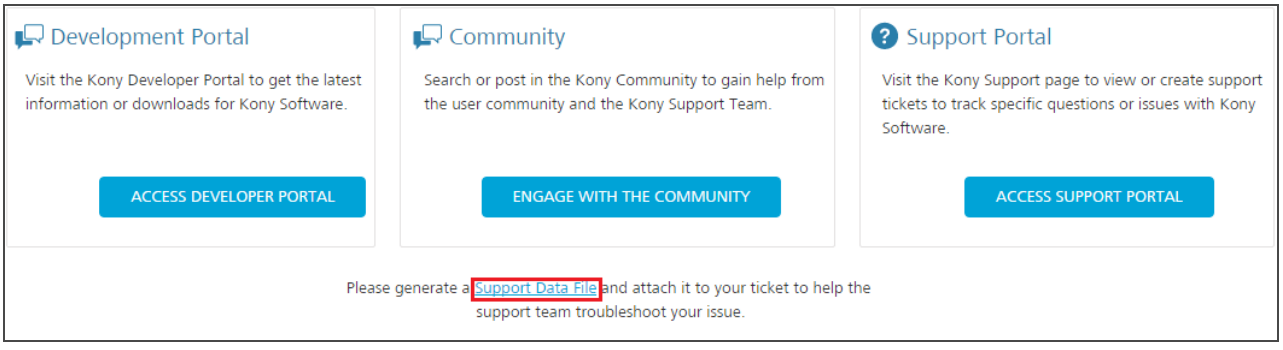
Another objective of the feature is to gather information regarding the Kony Fabric portal usage. The information enables Kony to analyze the important areas of the Kony Fabric portal which helps in developing user-centric features.

When a Kony Fabric user encounters an issue with the deployment, they file a support ticket with Kony Support and continue to work with an assigned Support Engineer to provide information about the environment and the issues faced. The process involves a lot of back and forth around gathering information about the system to identify the root cause of the problem.

To ease the process of troubleshooting an issue, Kony integrates telemetry tool which gathers background information about the user's deployment and also provides further information about the deployment state at the time the issue occurred. Now a user just needs to attach a diagnostics report, which contains all the necessary information, while filing a support ticket.

To generate a diagnostics report, follow these steps:

1. From Kony Fabric Console, click **Support**.



2. In the Support page, click **Support Data File**.

   A **diagnostics.json** file will be downloaded.

While filing a support ticket, attach the **diagnostics.json** file.

45. Tutorials

Through the following tutorial, you can review the steps of setting up Kony Fabric, configure a Salesforce account, configure Kony Fabric for your application, and build applications using Kony Fabric.

- Kony Fabric How To Tutorial

46. Appendix - Sync Strategy

During offline operation, the mobile application on the device is completely disconnected from the server. Instances of the same application that may run on other mobile devices at the same time, either connected to or disconnected from the server. Data stored locally on the mobile device running in offline mode may be modified or deleted or new data may be created. Potentially, the same data entity may be modified by multiple mobile clients concurrently, running either in online or offline operation, with potential effects on the server data store. These modifications that may be in conflict with others, have to be synchronized with the data on the server when a mobile application switches back to online operation. During this process, conflicts need to be reconciled and updates have to be made available to other application clients connected.

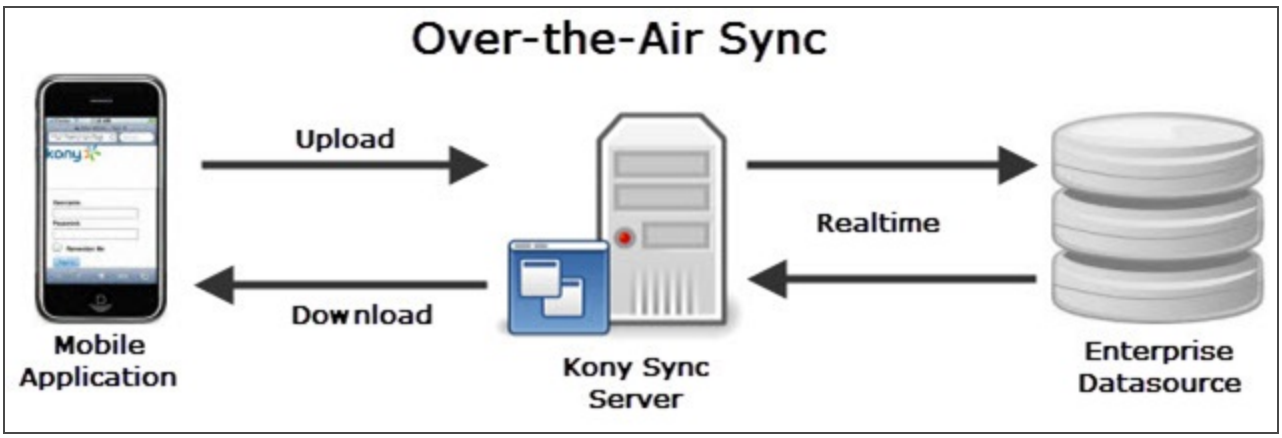
To enable the above synchronization needs the server may choose to store the data before merging it with the Enterprise Datasource. Though this is not a mandatory requirement, it is very desirable when the Enterprise Datasource has not been designed for handling additional mobile users or is only occasionally available.

Kony Fabric Sync Framework provides two flexible sync strategies that help the application developer design the application that best suites the enterprise needs.

46.1 Over The Air Sync (OTAsync)

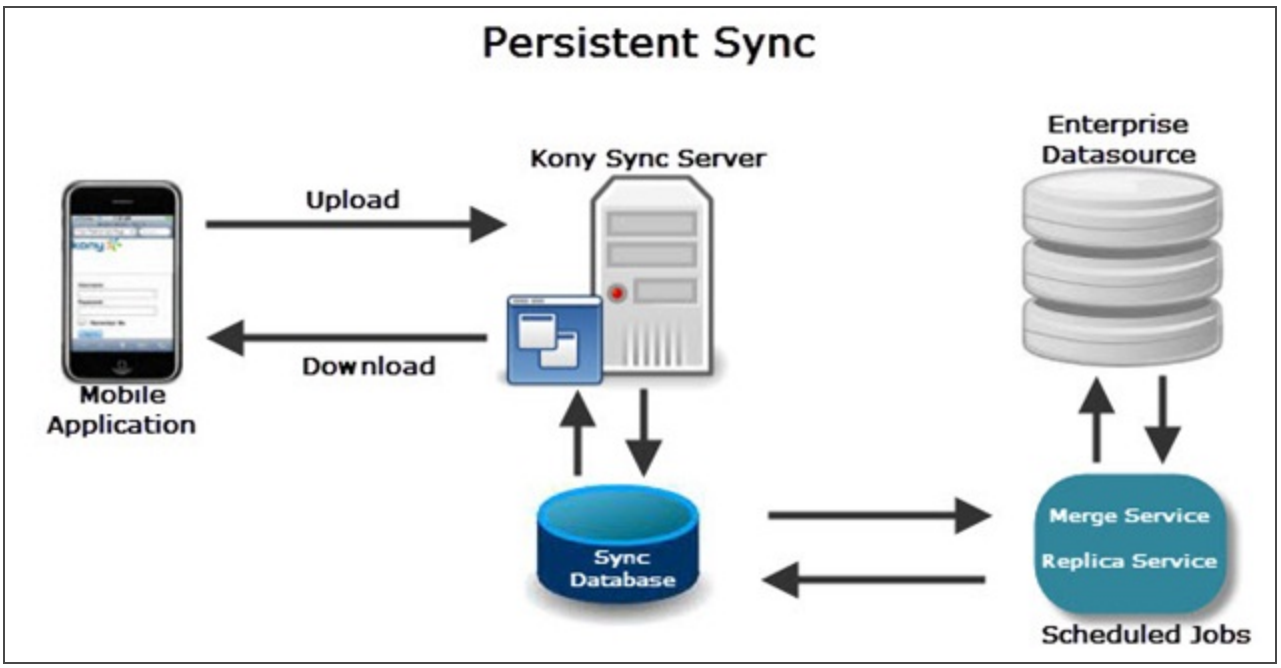
In this strategy whenever the mobile application invokes sync API, the Kony Fabric Sync server immediately merges upload packets with the enterprise server. Similarly the Kony Fabric Sync server queries the enterprise server for "delta" changes real-time and sends the same to the mobile application.

In this strategy, Enterprise Datasource is assumed to be available when the device starts the sync session.



46.2 Persistent Sync

In this strategy the application data is first persisted on Kony Fabric Sync server and later merged with the Enterprise Datasource as part of the offline process. Though this results in some data latency but ensures that the system is still functional even though the Enterprise Datasource may not be available.



In this strategy, the Sync Framework buffers / persists the backend data; during synchronization of a mobile client, the delta for that client is calculated based on the existing data in the Kony Fabric Sync Server. You can schedule replication jobs to occur at predetermined intervals to update the Kony Fabric Sync Server data. These jobs compare backend data with the data that is persisted in the Kony Fabric Sync Server and decide whether an object needs to be updated.

46.3 When to Use which Sync Strategy?

Deciding upon to use a Sync Strategy is one of the key architectural / designs decisions that you need to make when developing any enterprise grade offline application. This is not a simple decision but is based on number of input parameters and system constraints. Below are some of the recommendations that can help you determine to choose the appropriate strategy:

46.3.1 OTASync Strategy is recommended solution when:

- The Enterprise backend is highly available to the Kony Fabric Sync Server. It can be assumed that the availability of connectivity between the Device and Kony Fabric Sync server is the same as the availability of connectivity between the Kony Fabric Sync Server and the Enterprise backend. So whenever the device has network availability it can directly post the changes to the Enterprise backend.
- When there is a need to keep system and operational costs down by not having to replicate the enterprise backend on the Kony Fabric Sync server. Data replication does added to the amount of disk space needed to maintain the data and also resources to monitor the data.
- The Enterprise backend is “Provisioned”. Provisioned is the term used to signify that typical database design patterns are followed like tracking deletes through a soft delete flag and tracking lastupdated timestamp for each data item (or a table row). These elements are essential for OTASync as they are needed for sending appropriate updates back to the client database and keep the processing as low as possible on the Kony Fabric Sync server.
- The Enterprise backend can provide delta changes based on the last updated timestamp as that the device sends. Before we get into the details to understand the need for this, it is important to understand that the essence of OTASync strategy to is to avoid “persisting” a copy of the Enterprise backend data on the Kony Fabric Sync server. During OTASync, the device waits for

the client changes to merge with enterprise backend and at the same time, enterprise updates to been sent to the device. Now, if the enterprise backend does not provide the ability to query “deltas”, it means that you have to do the real time delta determination. This means that current state of device data is available on the Kony Fabric Sync server to compare with the current state of the data on the Enterprise backend (so that you can determine the deltas between two sources). Since we don’t persist data on the Kony Fabric Sync server (for OTASync strategy), this means sending a snapshot of the client data with every upload request may increase the pay load of the upload request to an unmanageable level. SalesForce WebServices offers a very good design pattern on how you should design services to suite the OTASync strategy.

- Every time the users perform a sync, they get access to the Latest updates on the Enterprise backend.

46.3.2 PersistentSync is recommended solution when:

- The Enterprise backend is not highly available. For example: the Order Processing system is busy serving desktop users during the peak hours (say 9:00 AM to 5:00 PM) and cannot take additional load of the “new” mobile users (employees accessing the system using mobile devices). So, in order to make sure that the mobile users are still able to submit requests, you have to persist these requests on Kony Fabric Sync server and merge with the Enterprise backend offline (during off peak hours).
- A scheduled system down time does not allow users to access the (typically due to time zone differences)
- The Enterprise backend is “UnProvisioned”. Provisioned is the term used to signify you have to follow typical database design patterns such as tracking deletes through a soft delete flag and tracking lastupdated timestamp for each data item (or a table row). An unprovisioned backend does not follow these characteristics and hence it means that Kony Fabric Sync Server has to do delta determination (which dataitems /rows are added new, deleted and updated). When we deal with row items in thousands, this can be a processor intensive and time consuming exercise. hence this is usually done offline say few times during a day for the entire dataset (data for all users in the system)

- It is acceptable that users are productive even with information set that is “outdated” or “stale” by a few hours. Merging of the data is done usually at periodic intervals, it means that data in the Replica Database can be outdated with respect to the Enterprise backend.

46.3.3 What are the prerequisites for OTASync strategy ?

In order to select the OTASync strategy for a particular SyncScope, ensure that the follow prerequisites are met:

- Enterprise backend is Provisioned. This means that it provides a way to query updated records based on a timestamp and also query the deleted records based on a timestamp.
- It should provide a way to get data items (rows) in batches. This is important as the device has limited memory and can process limited data at any given point in time.

46.3.4 What are the prerequisites for PersistentSync strategy?

In order to select the PersistentSync strategy for a particular SyncScope, ensure that the follow prerequisites are met:

- The system provides access to data for all users with a single login / authentication. This is essential since the replica database stores the data for all users and should be able to query the same from the Enterprise backend. The replica service does not refresh data for a particular user. It does across users. For example: it invokes getAll operation on Account objects. This means that it gets all the accounts across users and then tags the one that is updated and that is deleted. The replica service does not have any context of the users that are defined in the systems. The replica service only job is to refresh data periodically from Enterprise backend.
- Additional storage capacity to be allocated on Kony Fabric Sync Server as it now also stores a copy of the Enterprise backend

46.4 ChangeTracking

Applications requiring data synchronization capabilities require that changes are tracked in the server database so that these changes can be delivered to clients during a subsequent synchronization session (and vice versa). Below, we discuss different strategies that you can apply / configure to ensure that you can share only incremental updates between client and server.

Kony Fabric Sync framework provides the ability to either utilize change tracking that the datasource provides or in absence of such capability at the datasource, the Sync framework itself can change tracking in the server.

46.5 Conflict Resolution

During synchronizing two data stores (local device client and enterprise server), conflicts may occur if the same data object is modified in both participating stores independently. You can detect the conflicts by comparing state information of the two objects. The resolution of a conflict is called reconciliation and is always performed on the enterprise schema.

47. Appendix - App Services

App Services console provides you with the following details:

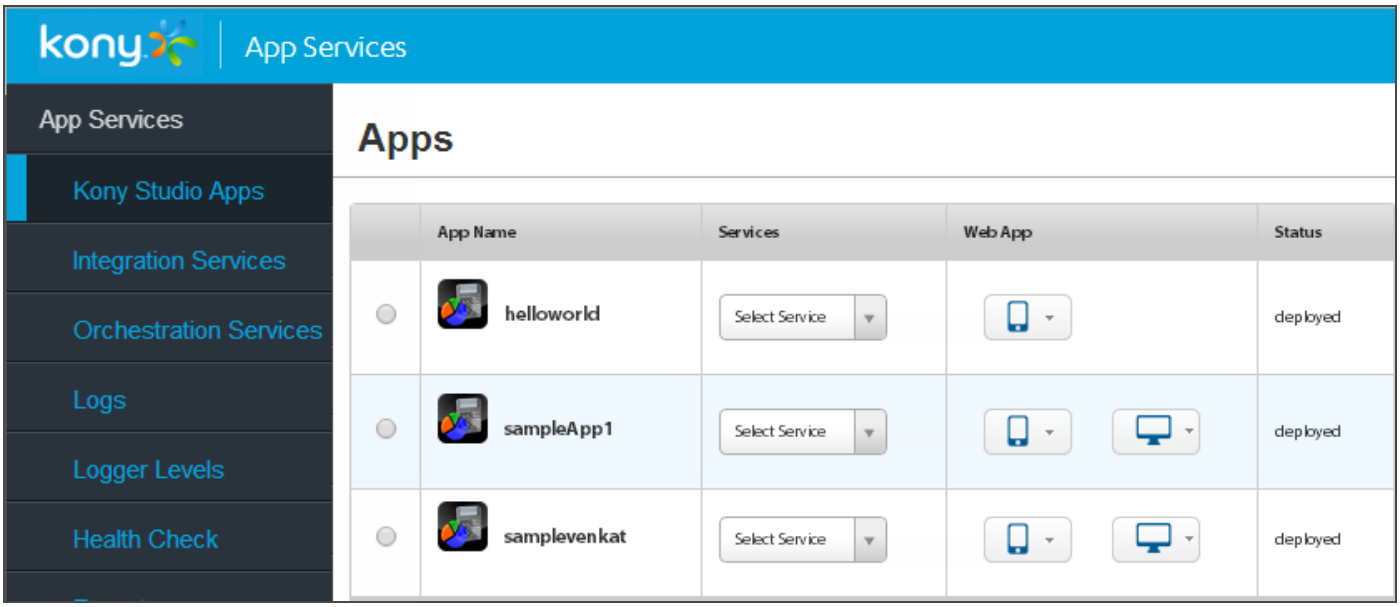
- List of Kony Studio applications in your kony cloud environment.
- List of Integration services and Orchestration services defined across your Kony Fabric environment.

App Services console consist of the following tabs:

- Kony Studio Apps: Displays a list of all the Kony Studio Apps.
- Integration Services: Displays a list of Integration services that are added across your Kony Fabric apps.
- Orchestration Services: Displays a list of Orchestration services that are added across your Kony Fabric apps.
- Logs: These are the system generated reports containing a list of activities that are performed based on the Logger Levels.
- Logger Levels: Allows you to specify the type of logs recoded for Integration Services and Orchestration Services.
- Health Check: The Health check view denotes the connection properties for App Server in a cloud environment
- Reports: Enables you to create Standard Reports, and Custom Reports.
- Downloads: Allows you to download the `middleware-system.jar` for writing preprocessor and postprocessor code.

47.1 Kony Studio Apps

Click **Kony Studio Apps** from the left pane of the console to view a list of Kony Studio applications available in your cloud environment.



For each of these Apps, following details are available:

1. **App Name**: Name of the Kony Studio application.
2. **Services**: List of all the services that are added to the app.
3. **Web App**: List the channels (Desktop, Mobile, and Tablet) on which the app is available.
4. **Status**: Specifies if the deployment of the app is successfully.

For each of the apps, you can perform following actions:

- Invoke an operation
- Launch an app
- Delete an app

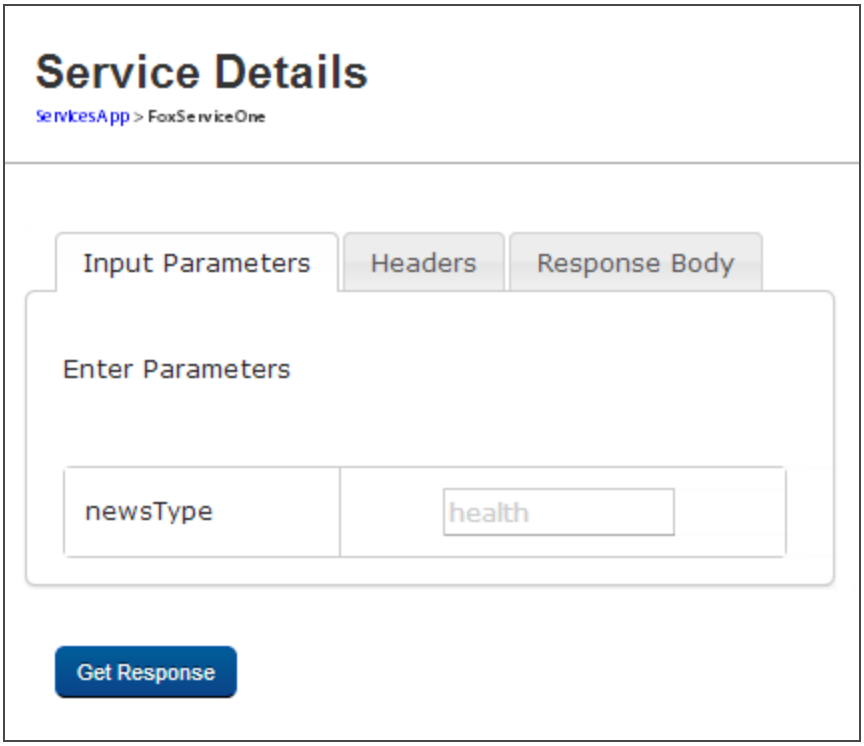
47.1.1 Invoking an operation

To invoke an operation, follow these steps:

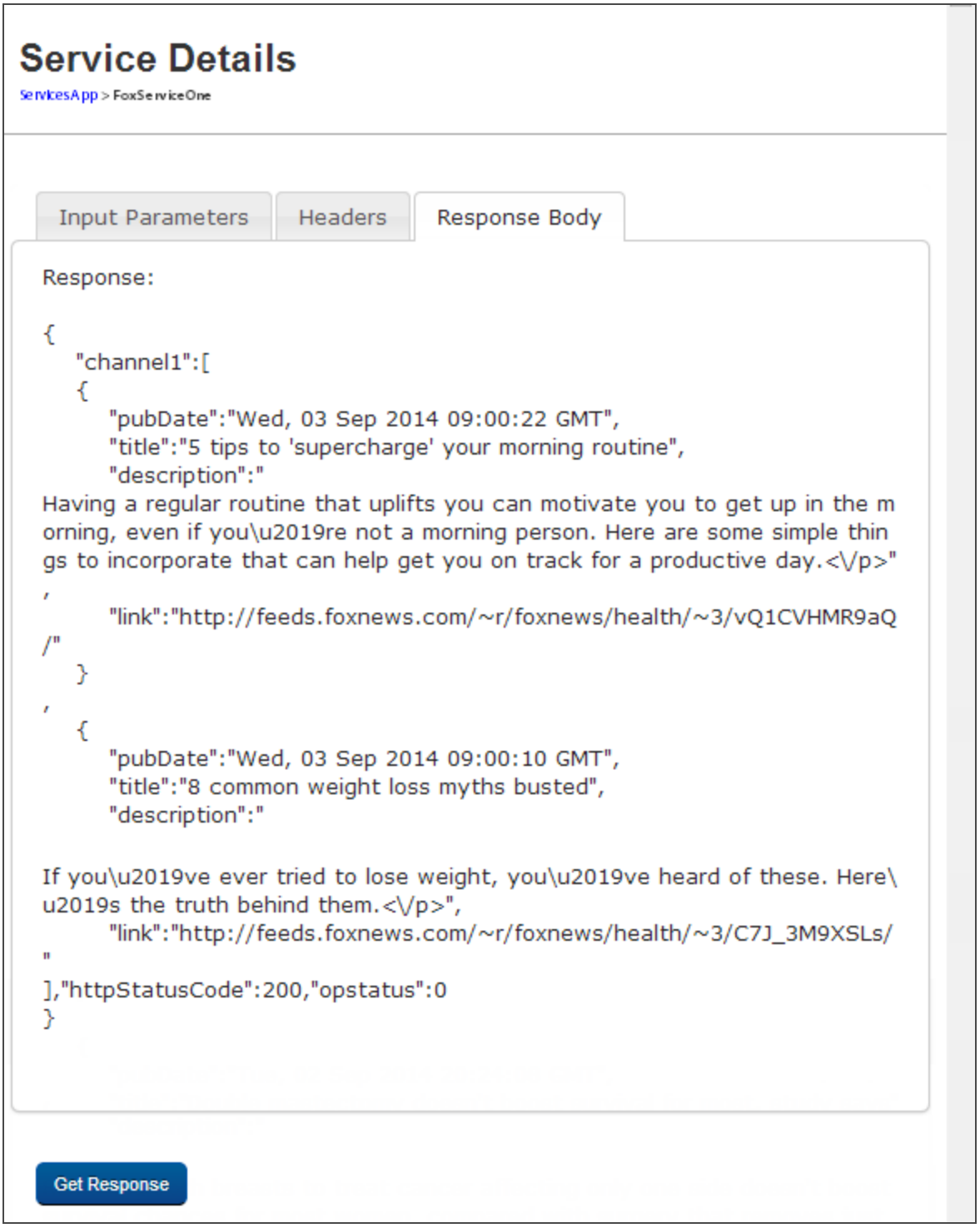
1. In the Kony Studio Apps page, select an operation from the **Services** list of an application.

The **Service Details** page is displayed.

2. Enter the input parameters,if required.



3. Click **Get Response**. The **Response Body** tab appears.

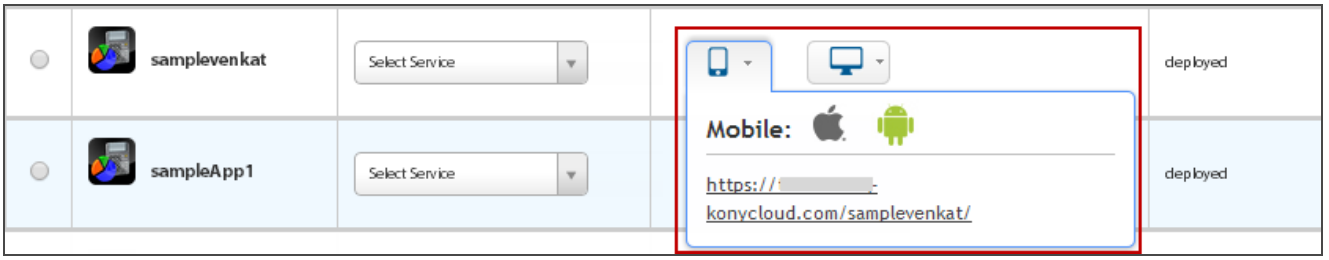


4. Click **Headers** to view the headers response.

47.1.2 Launching an App

To launch an application, follow these steps:

1. In the Kony Studio Apps page, click the down-arrow next to the channel (Desktop, Mobile, and Tablet).



   Following information is available:

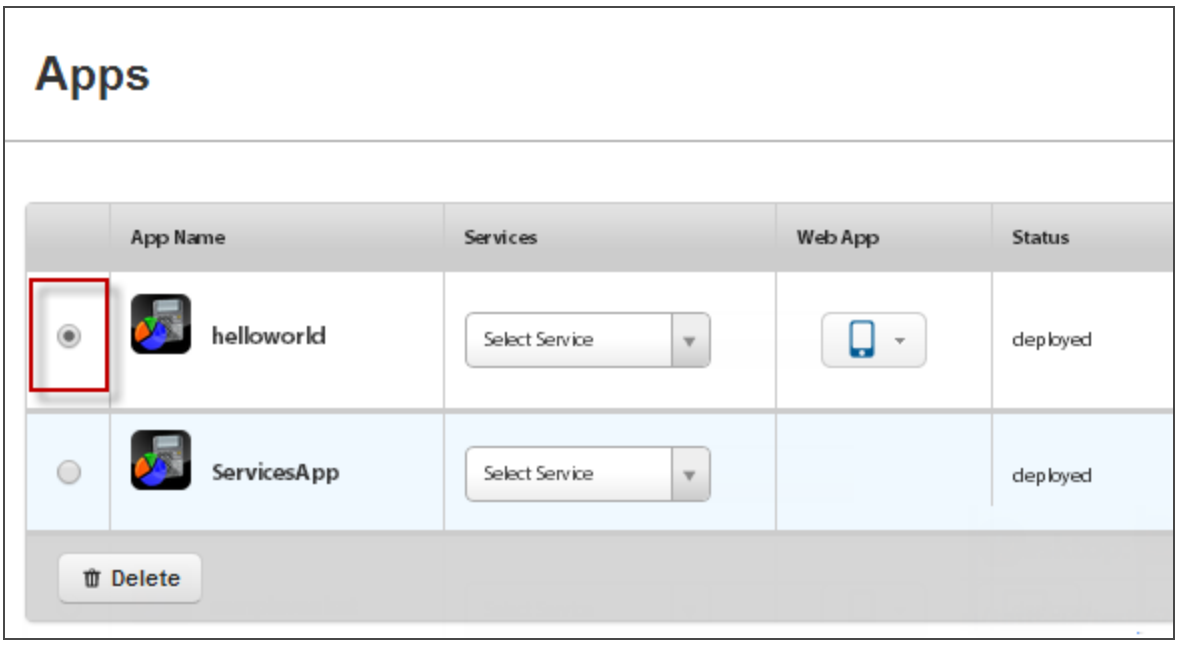
   - Platforms for which the app is built. For example, iOS and Android.
   - URL of the application.

2. Click the URL to launch the application.

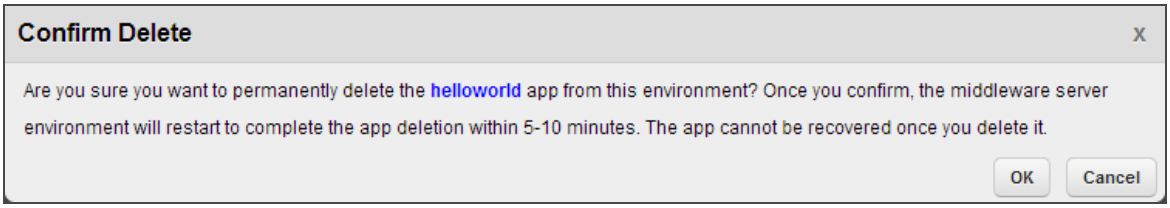
47.1.3 Deleting an app

To delete an app from the cloud, follow these steps:

1. In the **Kony Studio Apps** page, select an application.



2. Click **Delete**.

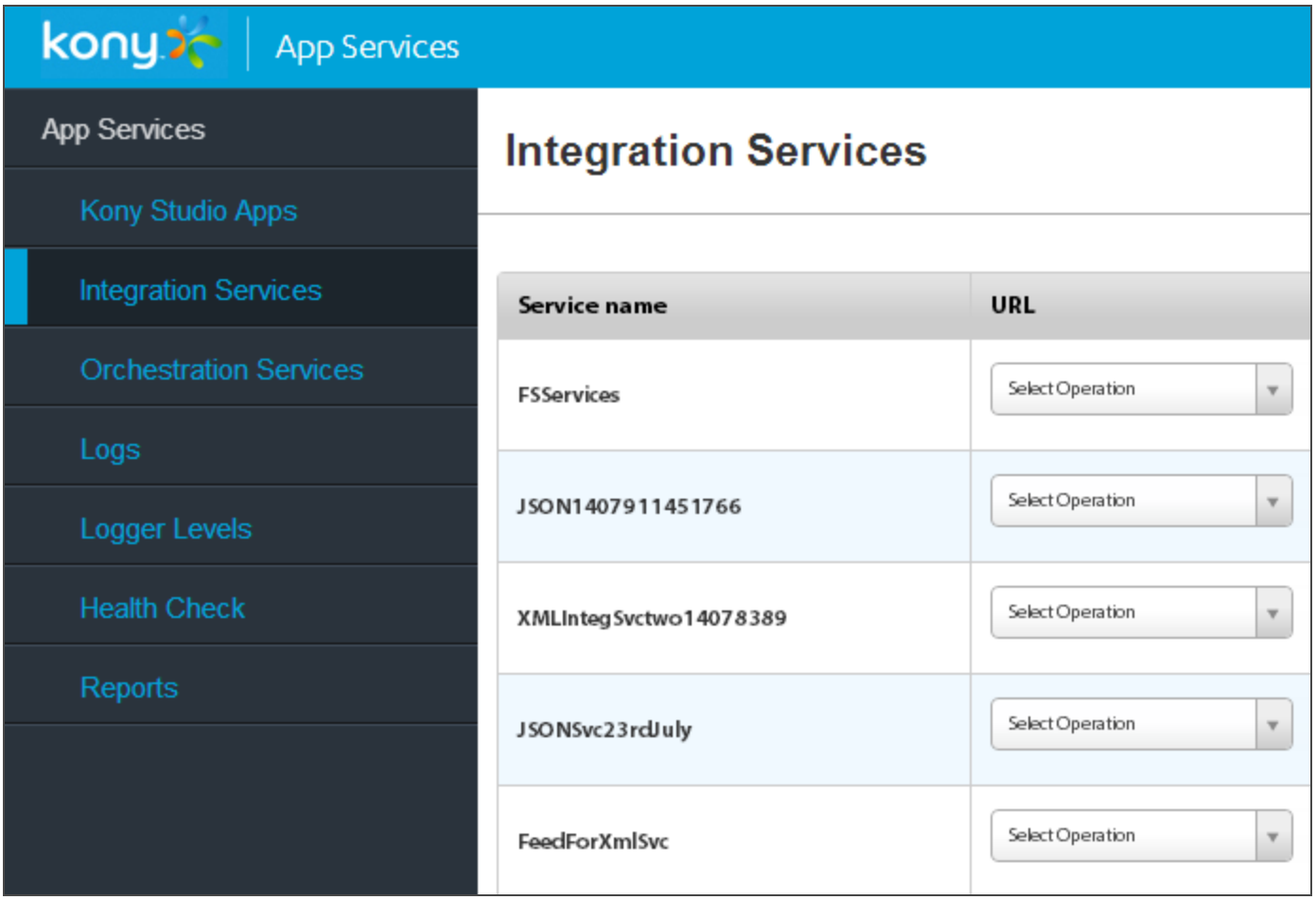


   A **Confirm Delete** window is displayed.

3. Click **OK** to delete the application.

47.2 Integration Services

Click **Integration Services** from the left pane of the console to view a list of Integration services that are available across the applications within your Kony Fabric environment.



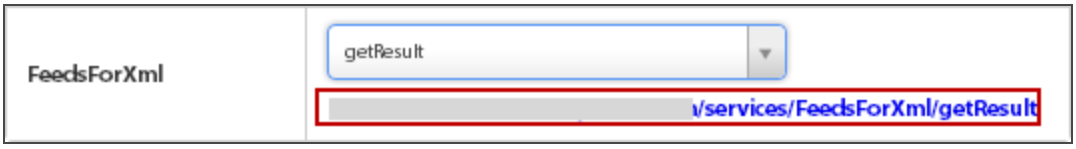
For each of the services, following details are available:

1. **Service name**: Name of the Integration service.
2. **URL**: Select an operation from the URL list to display the URL link of the operation. This link is used for invoking the corresponding operation.

47.2.1 Invoking an operation

To invoke an operation, follow these steps:

1. In the Integration page, select an operation from the URL list of an application.

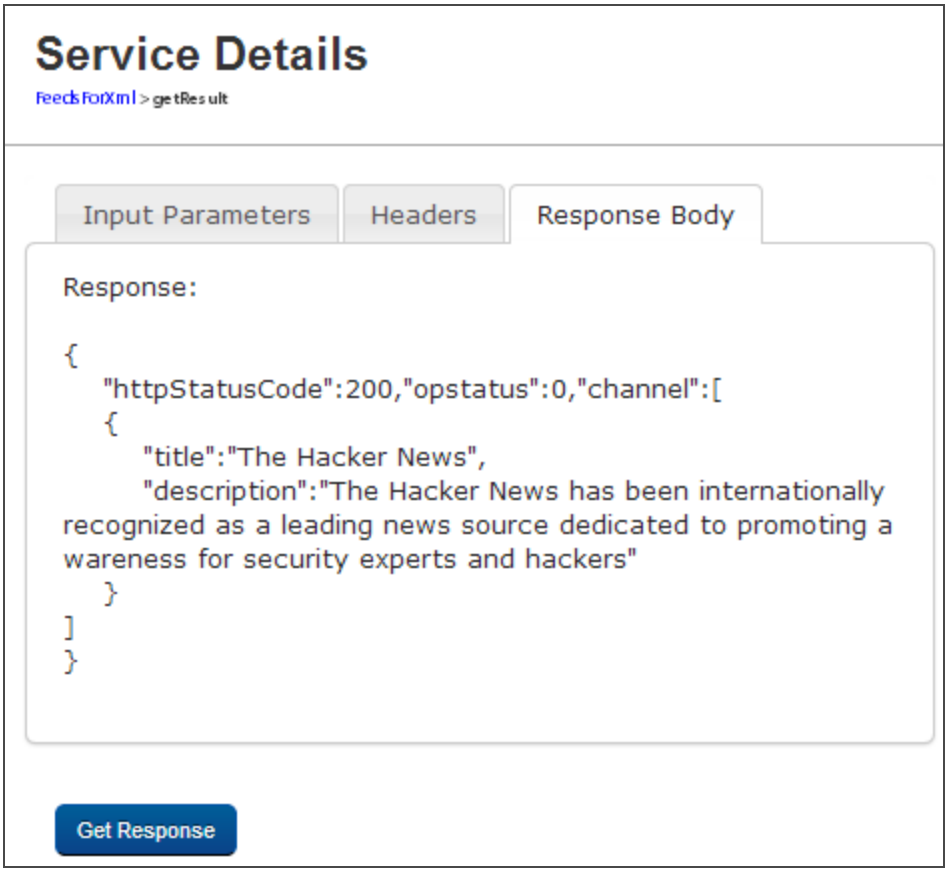


The URL of the operation is displayed.

2. Click the URL to open the **Service Details** page.

3. Enter the input parameters,if required.

4. Click **Get Response**. The **Response Body** tab is displayed.



5. Click **Headers** to view the headers response.

Service Details

FeedsForXml > getResult

Input Parameters | Headers | Response Body

Request Headers

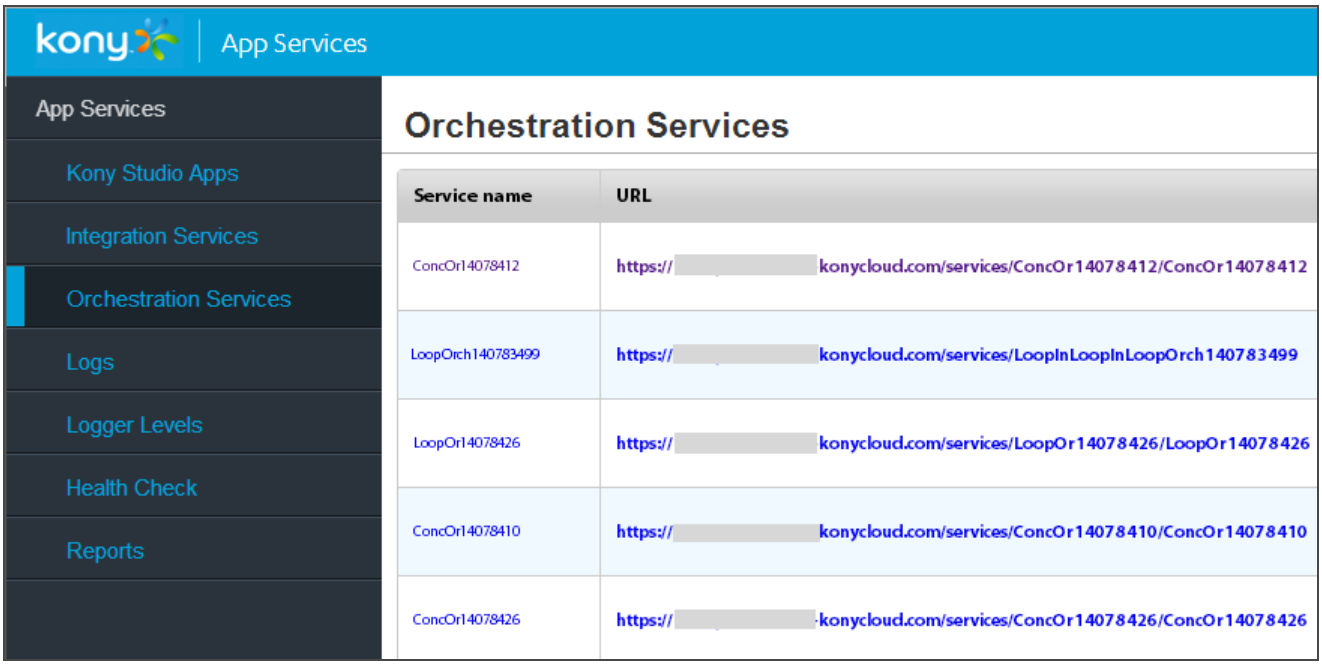
host	qaserver.qa-konycloud.com
content-type	text/plain; charset
x-kony-authorization	eyAiYWxnIjogIk5PTkUiLCAidHlwIjogImp3cyIgfQ.eyAiX3Njb3BlyI6ICJhY2NvdW50cyIsICJleHAiOiAxNDA5MjM2NDMyLCAiX2lzc2 92MS9tZXRhZGF0YS9YZFBsWnhEN2FfQXptWEJoMDc3cVl3PT0iLS
x-kony-server-console	ADMIN-PORTAL-LOGICAL-URL-CHANGE
x-newrelic-id	UwcPUlVSGwcGXFlbAgM
x-newrelic-transaction	PxRWAwNXCQsBV1MDBwcDX1YHFB8EBw8RVT8
content-length	101
x-forwarded-port	443
x-forwarded-proto	https
x-forwarded-host	qaserver.qa-konycloud.com
x-forwarded-server	10.0.252.135
connection	Keep-Alive

Response Headers

Access-Control-Allow-Methods	GET, HEAD, POST, TRACE, OPTIONS
Access-Control-Allow-Origin	*
Cache-Control	max-age
Cache-control	no-cache
Content-Type	text/plain;charset
Date	Thu, 28 Aug 2014 13:33:54 GMT
Pragma	no-cache
Server	Kony
Set-Cookie	cacheid
Set-Cookie	JSESSIONID
Vary	Accept-Encoding
X-NewRelic-App-Data	PxQAUF5QCwITUFdaDwgCVEYdFGQHBDcQUQxLA1tMXV1dORN GtWQhINB0MTG1ZKARoCTF9TUwlRAU4JCAUeEVcLUANWVQAH
Content-Length	1318
Connection	keep-alive

47.3 Orchestration Services

Click **Orchestration Services** from the left pane of the console to view a list of Orchestration services across the applications within your Kony Fabric environment.



For each of the services, following details are available:

1. Service name: Name of the Integration service.
2. URL: Link to invoke the corresponding operation.

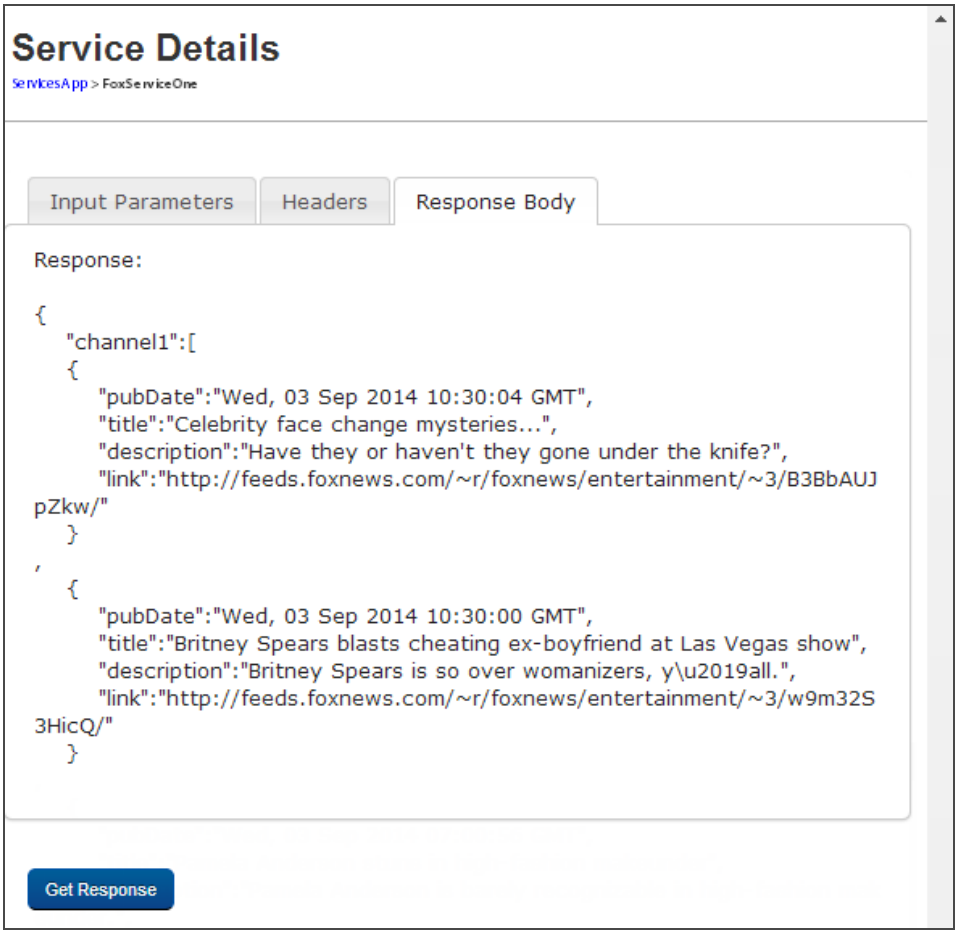
47.3.1 Invoking an operation

To invoke an operation, follow these steps:

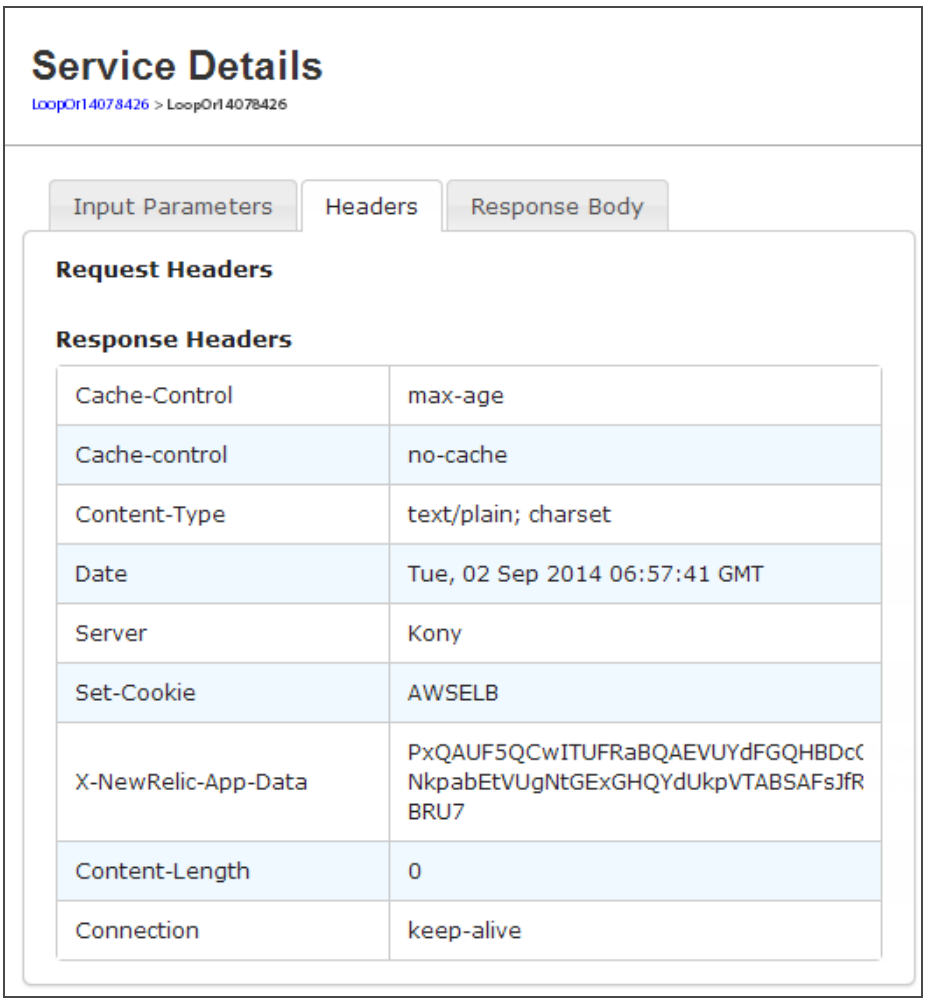
1. In the Orchestration page, click the corresponding URL of the operation to open the **Service Details** page.

2. Enter the input parameters, if required.

3. Click **Get Response**. The **Response Body** tab is displayed.



4. Click **Headers** to view the headers response.



47.4 Logs

Logs are automatically generated reports containing a list of activities that are performed based on the Logger Levels.

Logs show a list of instances with **Instance ID**, **IP Address** along with the links to view the **Archived** and **Snapshot** Logs. If the number of logs is more than ten, you can use the Next or Previous options to view other logs.

There are two types of cloud logs:

- Archived Logs
- Snapshot Logs

47.4.1 Archived Logs

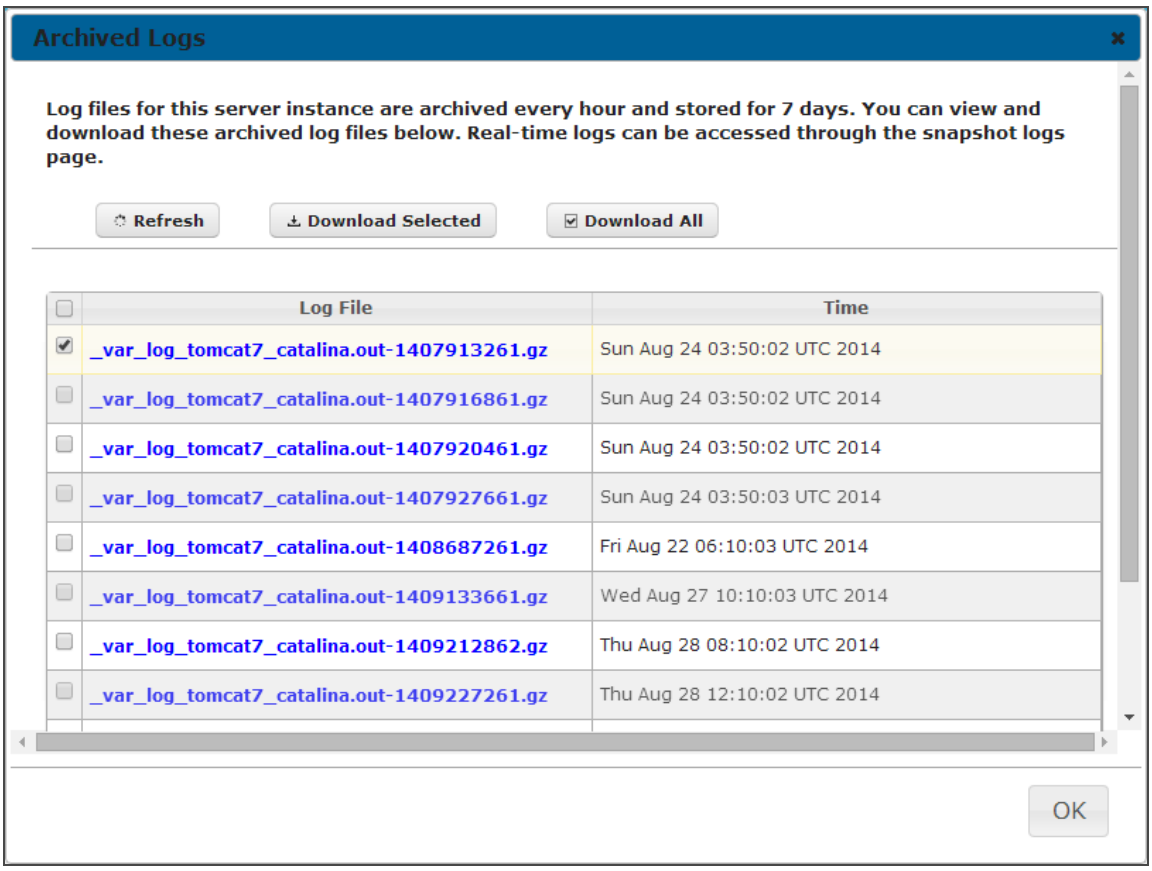
The log files for the server instance that are archived every hour and stored for seven days. You can view and download these archived log files. If the number of archived log files is more than 10, you can use Next or Previous to move to view other archived logs.

On Cloud Logs tab, you can view the list of archived logs and manage them. You can perform the following tasks:

- Viewing Archived Logs
- Downloading Selected Logs
- Downloading all the Logs
- Refreshing Archived Logs

47.4.1.1 Viewing Archived Logs

To view the archived logs, click **View Archived Logs** under Archived Logs. The Archived Logs pop-up appears with a list of the archived log files.

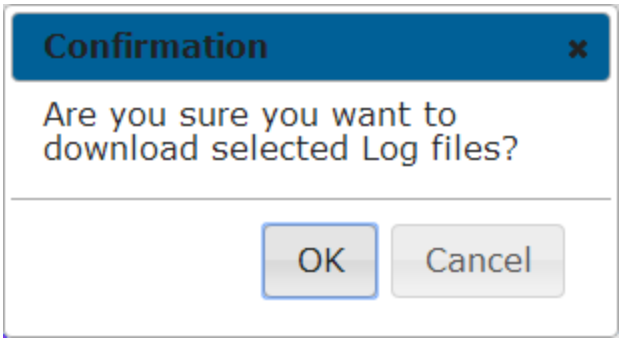


47.4.1.2 Downloading Selected Logs

To download the selected logs, follow these steps:

1. On the **Archived Logs** page, select the checkbox of desired archived file that you want to download and click **Download Selected**.

   A **Confirmation** pop-up appears.



2. Click **OK** to download the selected archived log files.

   A pop-up appears to choose the location to save the downloaded archived log files.

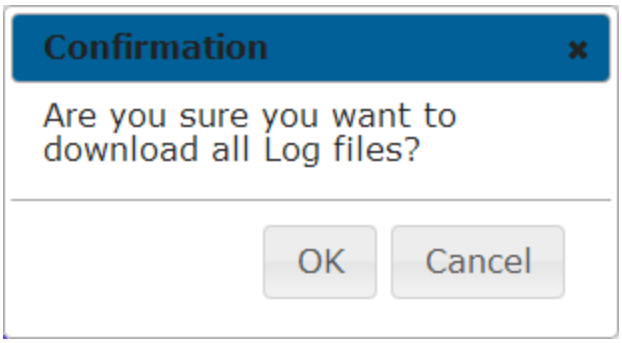
3. Select the location to save the downloaded archived log files, and click **OK**.

47.4.1.3 Downloading all the Logs

To download all the archived log files, follow these steps:

1. On the **Archived Logs** page, click **Download All**.

   A **Confirmation** pop-up appears.



2. Click **OK** to download the selected archived log files.

   A pop-up appears to choose the location to save all downloaded archived log files.

3. Select a location, and click **OK**.

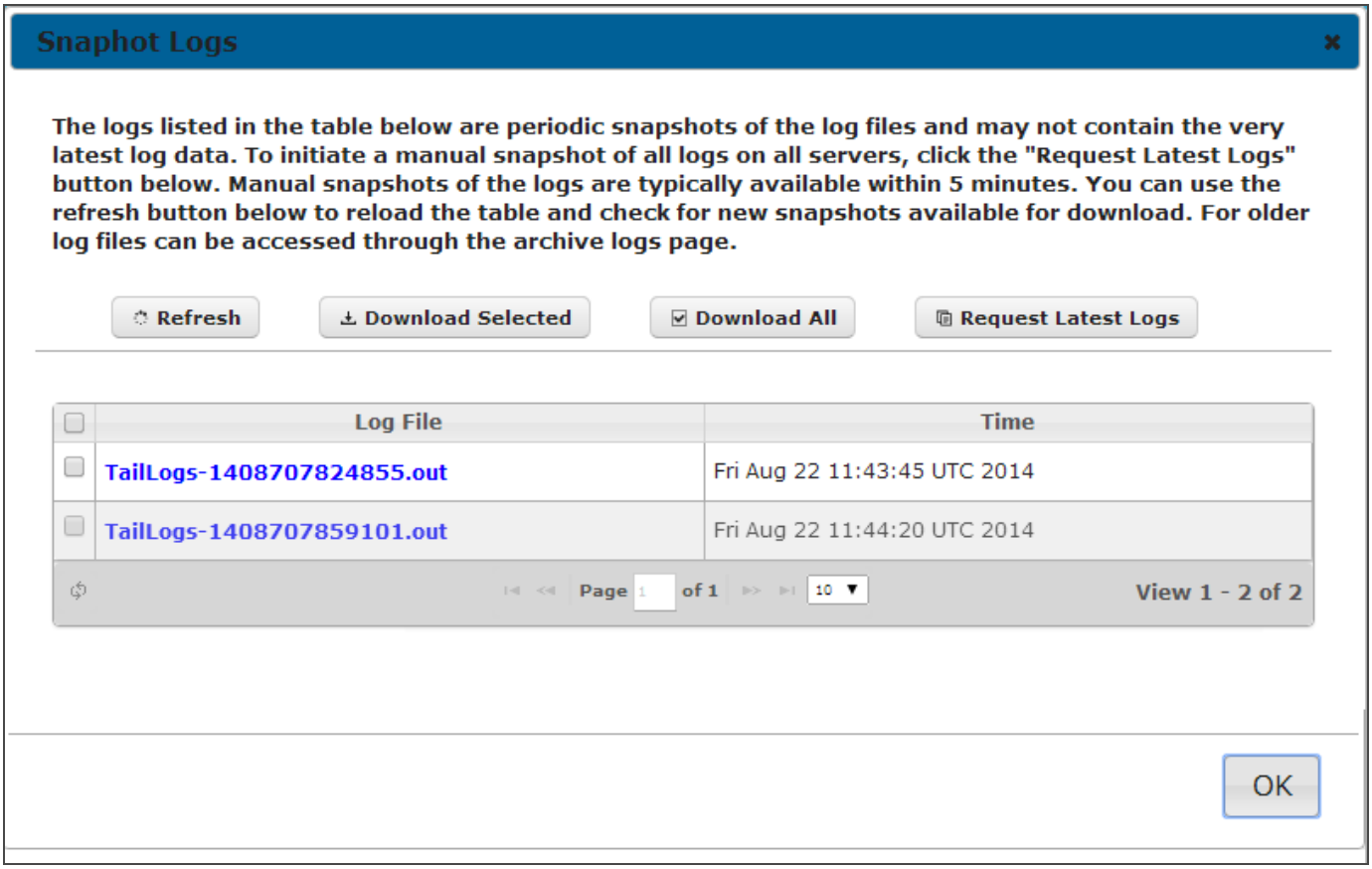
47.4.1.4 Refreshing Archived Logs

To refresh the archived logs, on the Archived Logs page, click **Refresh**.The Archived Logs are refreshed.

47.4.2 Snapshot Logs

You can access real-time logs through the snapshot logs page. Snapshot logs comprise the last manual log snapshot fetched from the server.The logs listed in the table are periodic snapshots of the log files and may not contain the very latest log data. You can view and download these snapshot log files. If the number of snapshot log files is more than ten, you can use Next or Previous options to move to more number of snapshot log files.

On Cloud Logs tab, you can view the list of snapshot logs and manage them.



You can perform the following tasks:

- Viewing Snapshot Logs
- Requesting Latest Snapshot Logs
- Downloading Selected Logs
- Downloading all the Logs
- Refreshing Snapshot Logs

47.4.2.1 Viewing Snapshot Logs

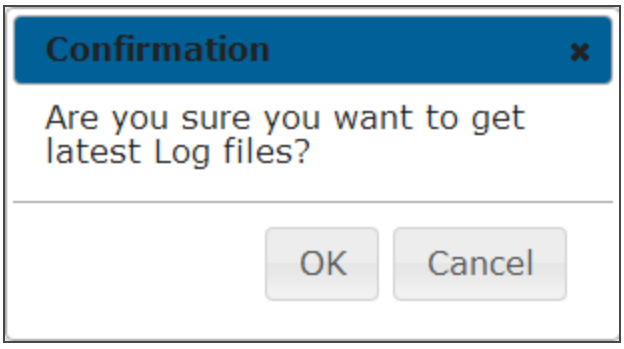
To view the snapshot logs, click **View Snapshot Logs** under **Snapshot Logs**. The **Snapshot Logs** pop-up appears with all the snapshot log files.

47.4.2.2 Requesting Latest Snapshot Logs

To request latest snapshot logs, follow these steps:

1. On the **Snapshot Logs** page, to initiate a manual snapshot of all logs on all servers, follow these steps:
2. Click **Request Latest Logs**.

   A Confirmation pop-up appears.

3. Click **OK** to download the latest log files.

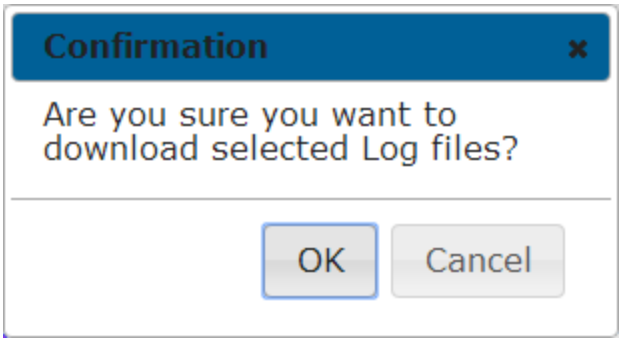
   The manual snapshots of the logs are typically available within five minute

47.4.2.3 Downloading Selected Logs

To download the selected log files, follow these steps:

1. On the **Snapshot Logs** page, select the checkbox of desired snapshot file that you want to download and click **Download Selected**.

   A **Confirmation** pop-up appears.



2. Click **OK** to download the selected snapshot log files.

   A pop-up appears to choose the location to save all downloaded archived log files.

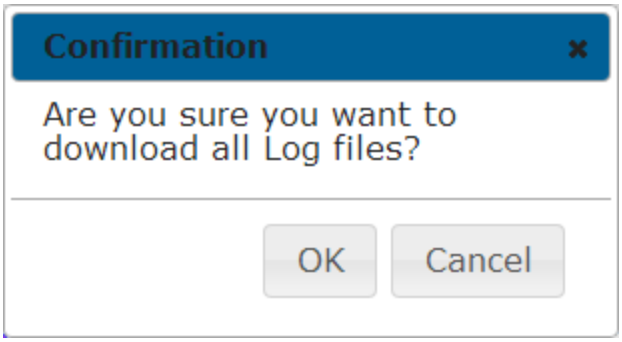
3. Select a location, and click **OK**.

47.4.2.4 Downloading all the Logs

To download all the snapshot log files, follow these steps:

1. On the **Snapshot Logs** page, click **Download All**.

   A **Confirmation** pop-up appears.



2. Click **OK** to download the selected snapshot log files.

   A pop-up appears to choose the location to save all downloaded snapshot log files.

3. Select the location, and click **OK**.

47.4.2.5 Refreshing Snapshot Logs

To refresh the snapshot logs and to reload the table with new snapshots available for download, on the **Snapshot Logs** page, click **Refresh**.

The Snapshot Logs are refreshed.

47.5 Logger Levels

Logger levels enable you to specify the type of logs recoded for Integration Services and Orchestration Services.

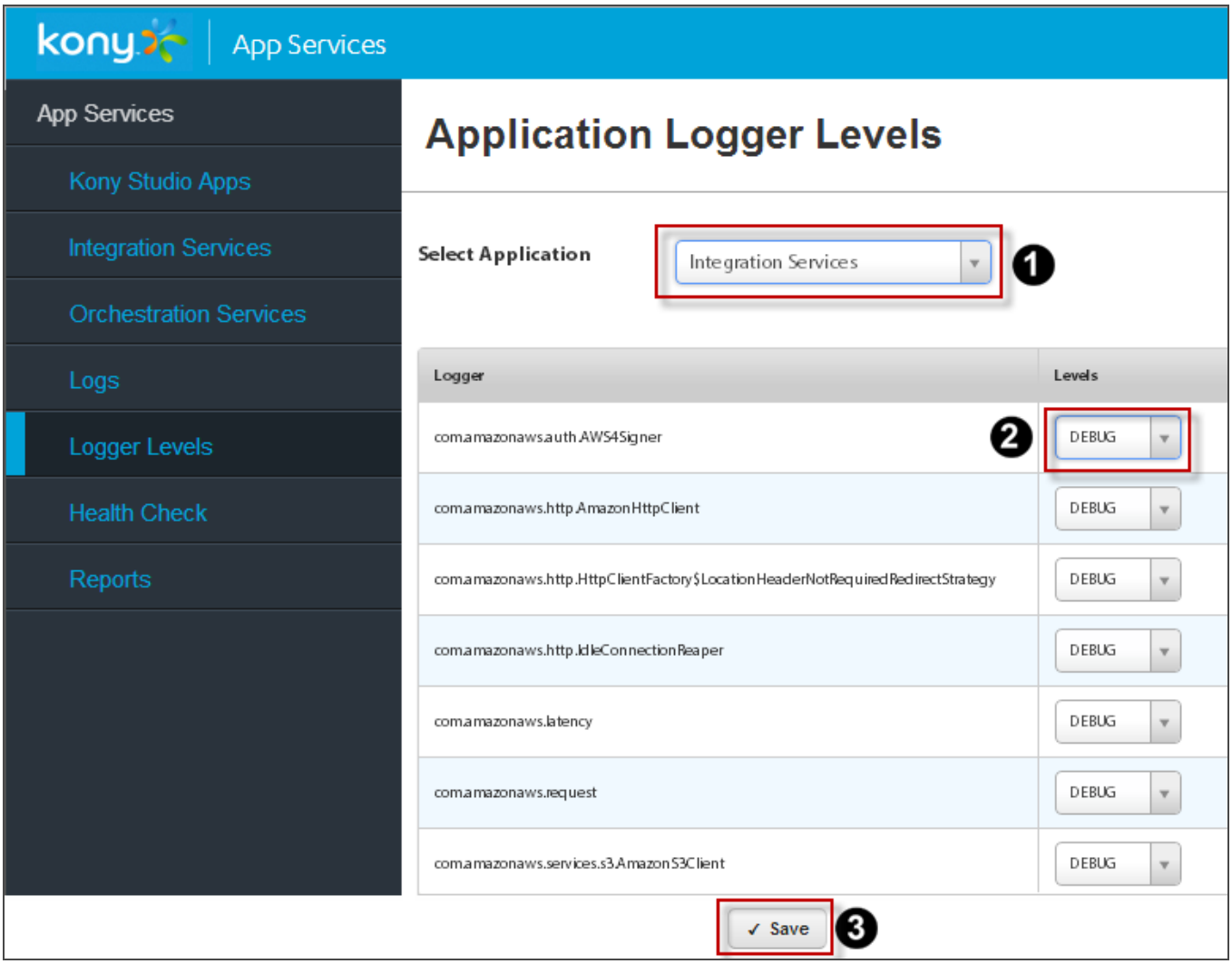
Following type of loggers are supported:

- **Trace**: It designates finer-grained informational events than the *DEBUG*.
- **DEBUG**: It designates fine-grained informational events that are most useful to debug an application.
- **INFO**: It designates informational messages that highlight the progress of the application at coarse-grained level.
- **WARN**: It designates potentially harmful situations.

- **ERROR**: It designates error events that might still allow the application to continue running.
- **FATAL**: It designates very severe error events that will presumably lead the application to abort.
- **OFF**: It designates that the logging is turned off.

47.5.1 Assigning a logger level

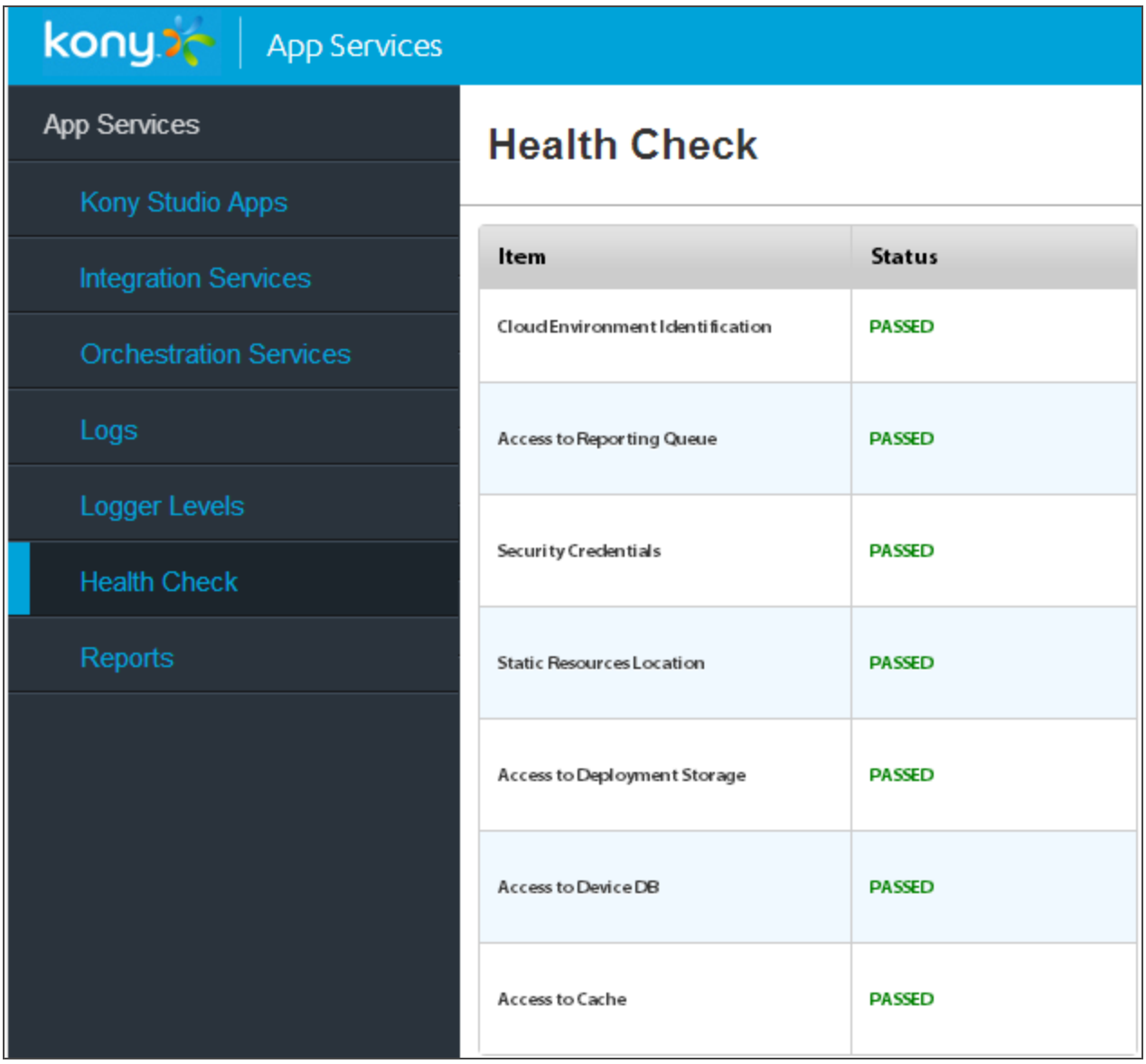
To assign logger levels for the services, follow these steps:



1. On the **Logger Levels** page, select the type of service from the **Select Application** list.
2. Update the levels for all the required loggers.
3. Click **Save**.

47.6 Health Check

The Health check view denotes the connection properties for App Server in a cloud environment.



The following are the various connection properties that denote the health of a App Server:

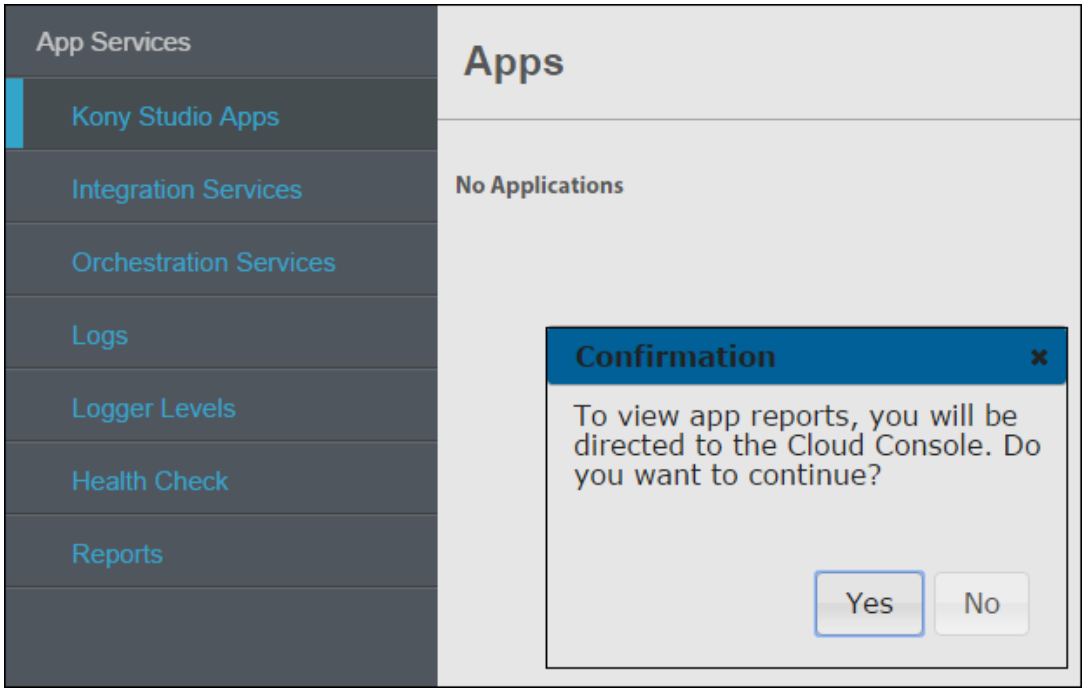
- **Cloud Environment Identification**: Indicates if the App server is running properly in the cloud environment.
- **Access to Reporting Queue**: Indicates if the environment has proper reporting Queue URL, and is able to connect.
- **Security Credentials**: Indicates if the App server has the cloud security credentials.
- **Static Resources Location**: Indicates if you are able to connect to the Amazon S3 bucket resources.
- **Access to Deployment Storage**: Indicates if you are able to connect to the Amazon S3 bucket storage repository.
- **Access to Device DB**: Indicates if you are able to connect to the Device DB.
- **Access to Cache**: Indicates if you are able to connect to the “Amazon Elastic Cache”.

47.7 Reports

Enables you to create Standard Reports, and Custom Reports.

To view or generate reports, follow these steps.

1. From the left pane of the console click **Reports**. A **Confirmation** pop-up appears.



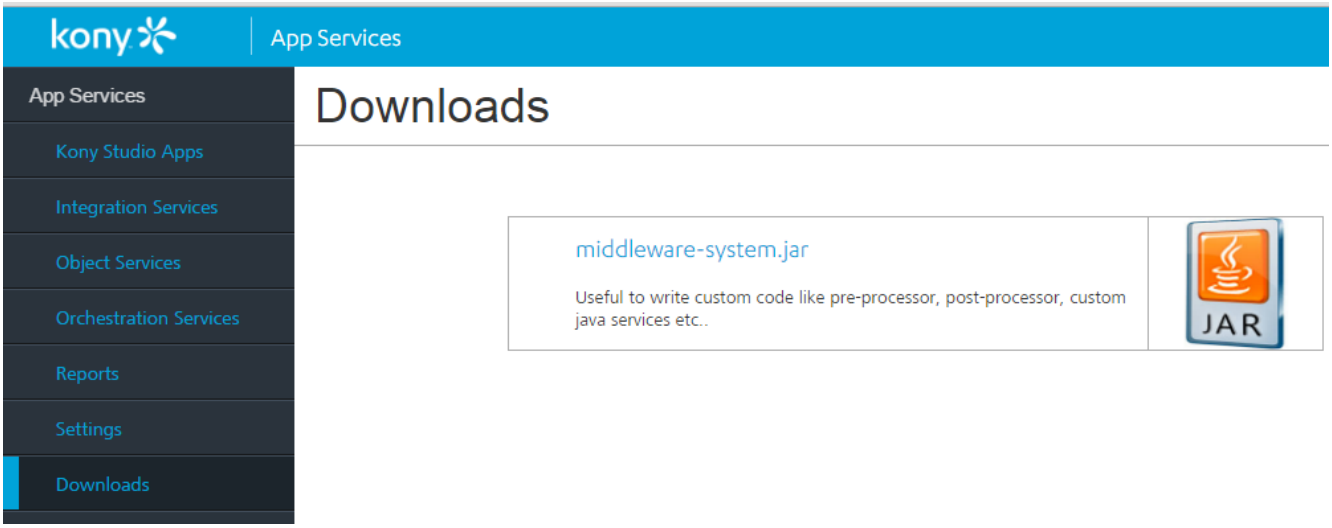
2. Click **Yes**. You are directed to the **Reports** page.

For more information on generation reports refer to:

- Refer to Kony Reporting and Analytics - Standard Metrics and Reports.
- Refer to Kony Reporting and Analytics - Custom Metrics and Reports

47.8 Downloads

Enables you to download the `middleware-system.jar` file.



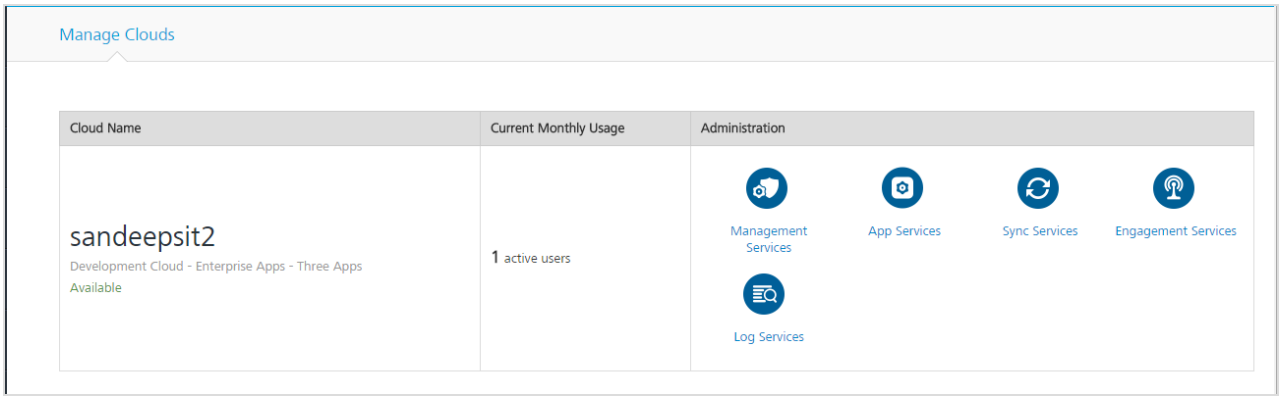
Click on the image to download the `middleware-system.jar` file.

The downloaded file is used to write the custom code for preprocessor, postprocessor and custom Java services.

- Refer to Preprocessor and Postprocessor

47.9 Log Services

A new feature log services is added under Cloud environments in **Features** section along with the existing features. This feature displays the log data of different servers.



Click **Log Services** icon to view the logged data in Kony Logs interface. This data is filtered depending on a particular environment. In this interface, a user can view the log data of different servers:

- Integration
- Sync
- Application crash logs

> ***Note:*** The log levels of the applications are set up from the corresponding consoles (such as Admin, Sync, engagement services and so on).

The Kony Logs interface is built with three different tabs.

- Discover
- Visualize
- Dashboard

47.9.1 Discover

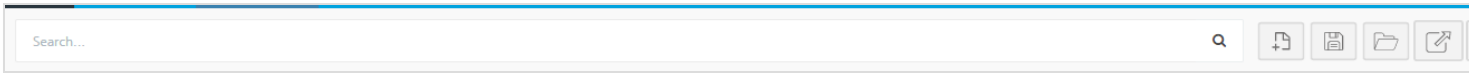
This tab is displayed by default with the list of log data at service level. The log data is displayed at a service level. Different filters are provided in this screen for the users to refine the data displayed.

> ***Note:*** Clicking on **Log Services** option displays the log data under **Discover** tab.

> ***Note:*** Some values useful for debugging, such as the **Service name** and the **Operation name** are displayed in the log message.

47.9.1.1 Search

You can search the particular logs from the displayed list of logs by providing a keyword in the Search field. Following actions can be performed in the Search pane.



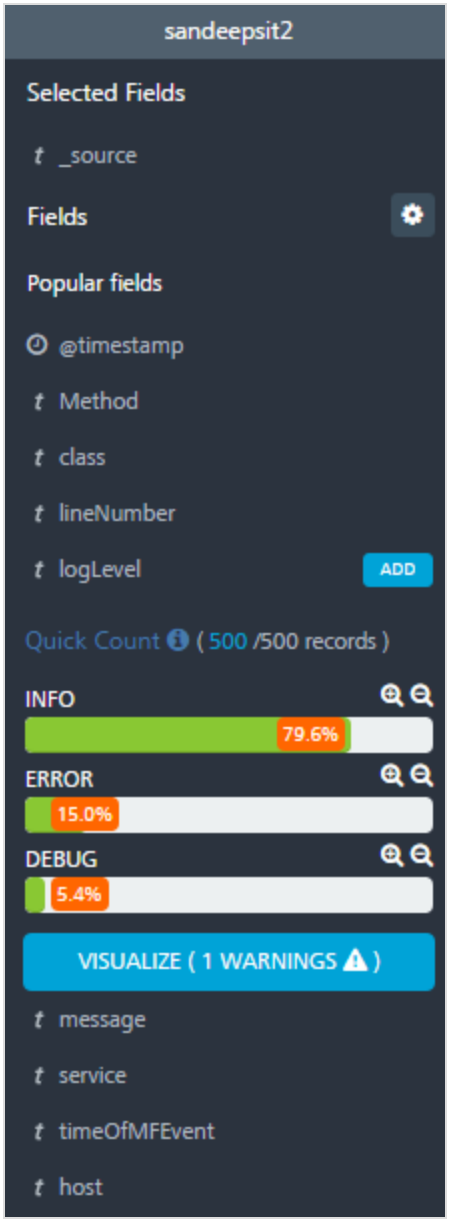
New Search: Click the icon to perform a new search. This refreshes the existing results and performs a new search by displaying the updated results.

Save Search: You can save the search performed for future use by clicking on Save Search. Saving the search is more efficient way to access the results more quickly instead of retyping the search keywords every time.

Load Saved Search: Click on Load Saved Search icon to load the previously saved searches. Select the saved search from the list of searches displayed.

47.9.1.2 Filters

You can refine the results displayed by using the pre-defined filters from the left pane of the screen. Select the required fields and provide the valid inputs to display the proper required log data.



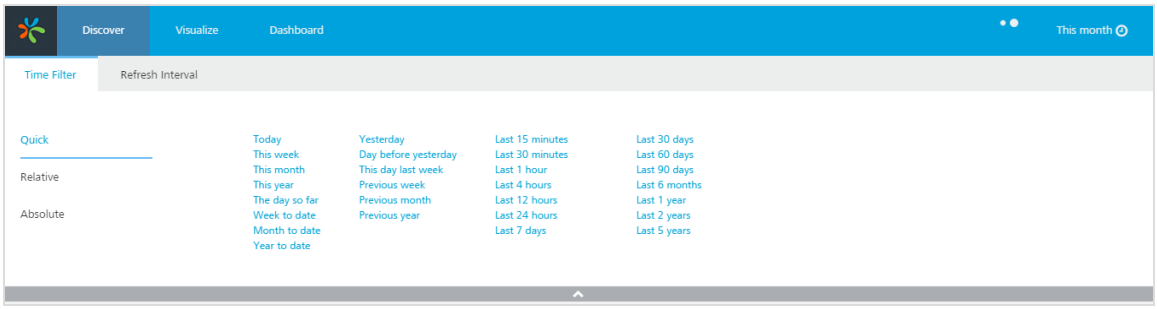
47.9.1.3 Time Filter

The time filter displayed on the top right corner of the screen shows the refresh time of the log data displayed.

You can modify the time filter to refine the log data using different options provided.

- Quick

- Relative and
- Absolute



Quick: Quick time filter displays different periods of time, typically day, week, month, year etc.

Results are displayed based on the selection of the filters.

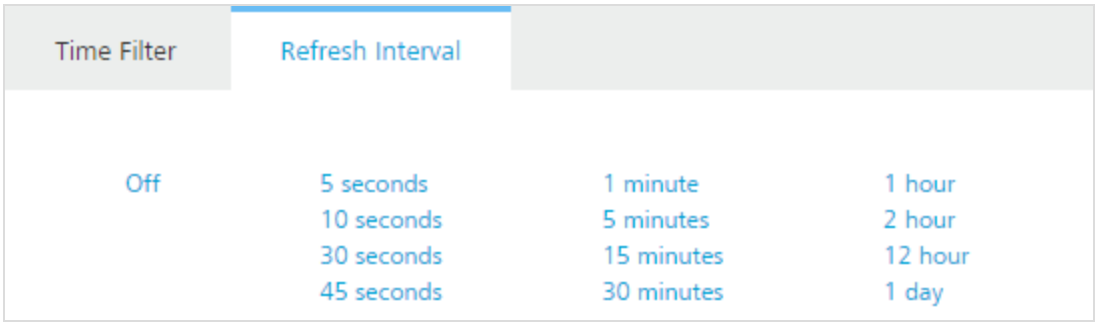
Relative: Relative filter is dependent on the Quick filter. Based on the selected period, user can set the time bound from to the current date and time. The time bound selected can be

- Seconds ago
- Minutes ago
- Hours ago
- Days ago
- Weeks ago and
- Months ago

Select the checkbox Round the day to filter the log data continuing the entire 24 hours of the selected day.

Absolute: Absolute displays the calendar to filter the data from a selected date and time to the selected date and time. Click Set to Now button in the To field to set the time period to the current date and current time.

Refresh Interval: The maximum refresh time interval can be set by the user in **Refresh Interval** tab. The Logging Service automatically check for the updates by providing the refresh interval time. Customize the refresh time of the application by selecting the refresh time interval you wish to use.

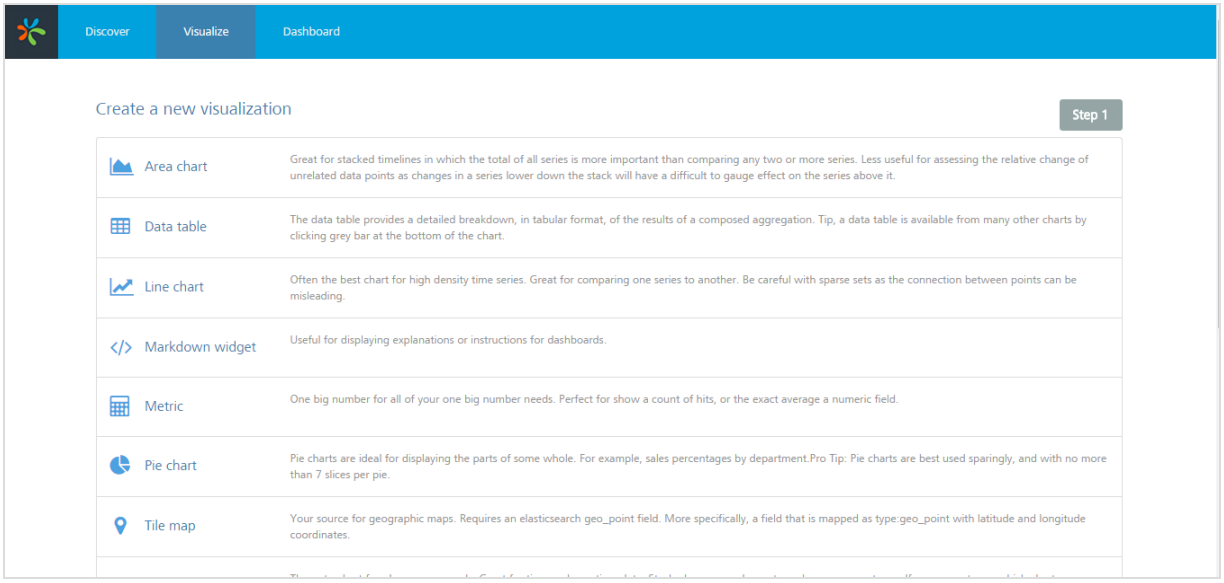


> ***Note:*** You can manually check for the updates by setting the refresh interval as **Off**.

47.9.2 Visualize

Visualization provides options to design different types of charts for the log data displayed. You can create a new visualization charts such as:

- Area chart
- Data table
- Line chart
- Markdown widget
- Metric
- Pie chart
- Tile map
- Vertical bar chart

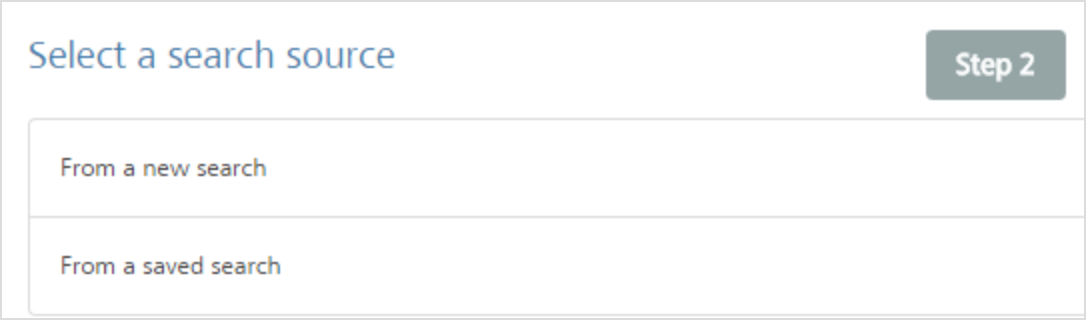


You can also view the saved visualizations at the bottom, if any.

47.9.2.1 Create a new visualization

To create a new visualization, follow these steps:

1. Click a type of chart to create, a new page **Select a search source** is displayed.
2. Charts can be prepared only on saved searches. Select the source to create a new chart.
    - From a new search
    - From a saved search



From a New Search

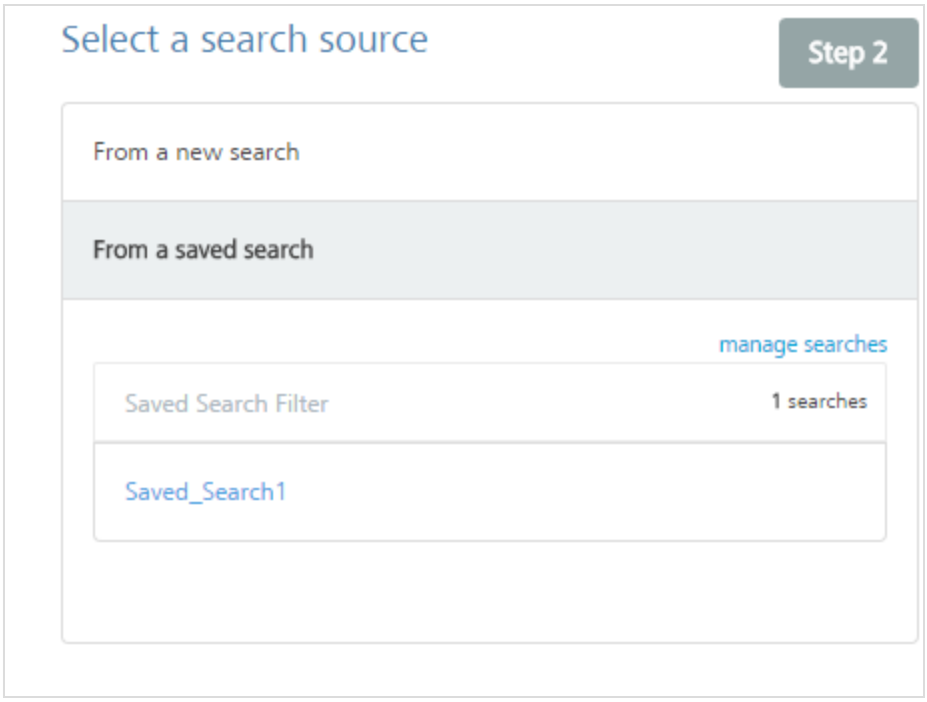
Follow these steps to create a chart from a new search:

- Select **From a new search** option to display a filter to provide the search options.
- Provide the required access filters and the visualization will be displayed.
- Save the visualization.

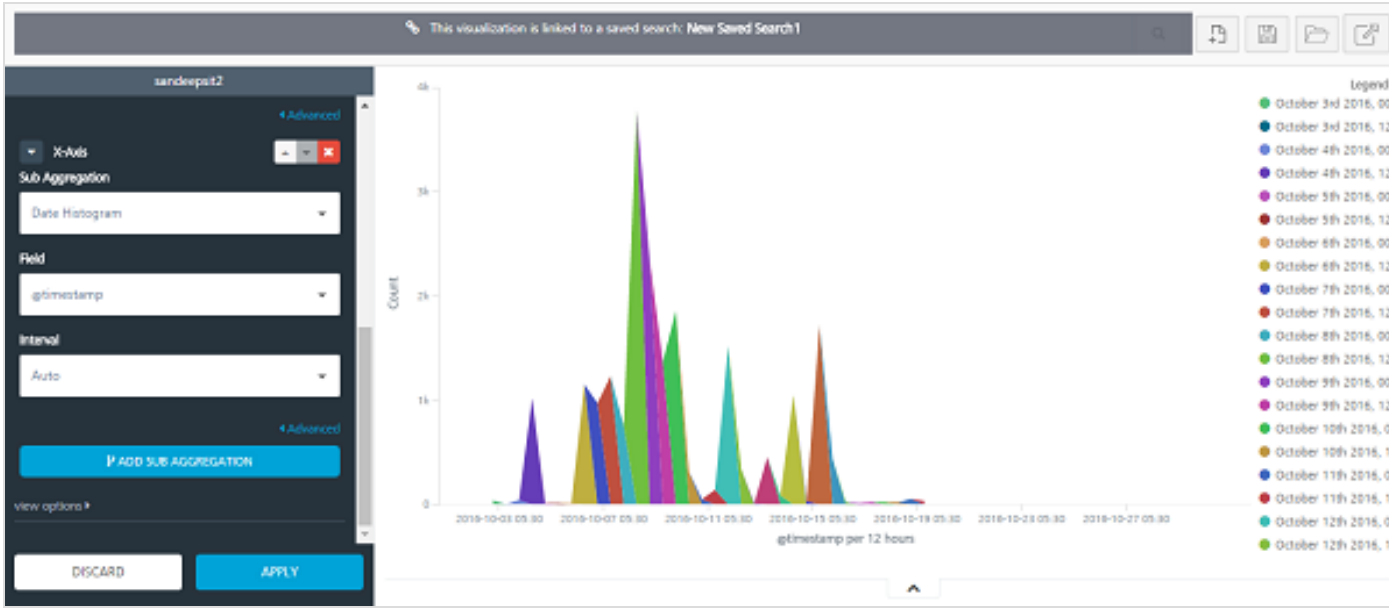
From a Saved Search

Follow these steps to create a visualization from a saved search

1. Click from a saved search option to create a visualization from a saved search. For more information on Saved search, refer Save search.



2. Based on the search parameters, Visualization is displayed.



3. Make the required changes by using the filters provided in the left pane.

4. Click **Save Visualization** to save it.

5. **New Visualization**: You can create a new visualisation by clicking on the **New Visualisation** icon. This option provides different graphical representations allowing you to create a new visualisation.
6. **Save Visualization**: Click **Save Visualisation** to save the created visualization.
7. **Load Saved Visualization**: click this option to load the previously saved visualization.
8. **Share Visualization**: Selecting this option will provide you two links where one link can be embedded to HTML source and the other link can be shared.

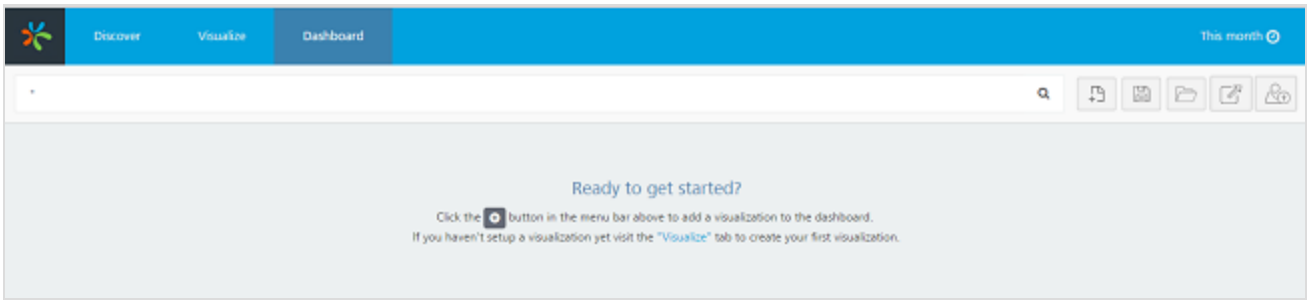
   > ***Note:*** The client embedding the link to their source code should have access to Kibana.

9. **Refresh**: Refreshes the existing data

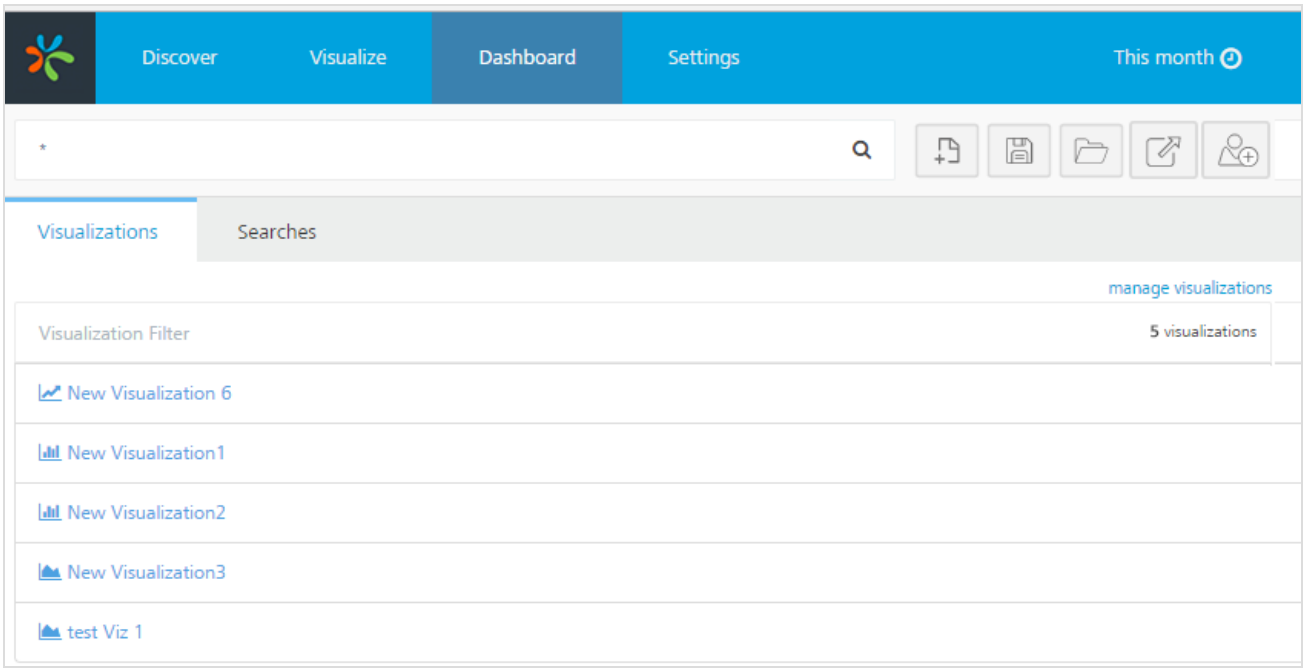
47.9.3 Dashboard

You can create dashboards for saved visualizations. To create a dashboard, follow these steps:

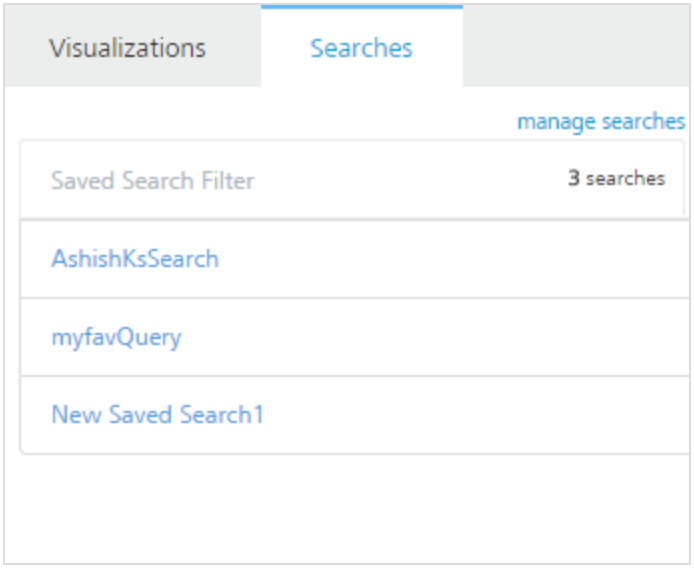
1. Click the **Dashboard** tab. The Dashboard home page is displayed.



2. Click the '+' button to add a visualization to the dashboard.
3. The dashboard page displays 2 tabs
   - Visualizations
   - Searches



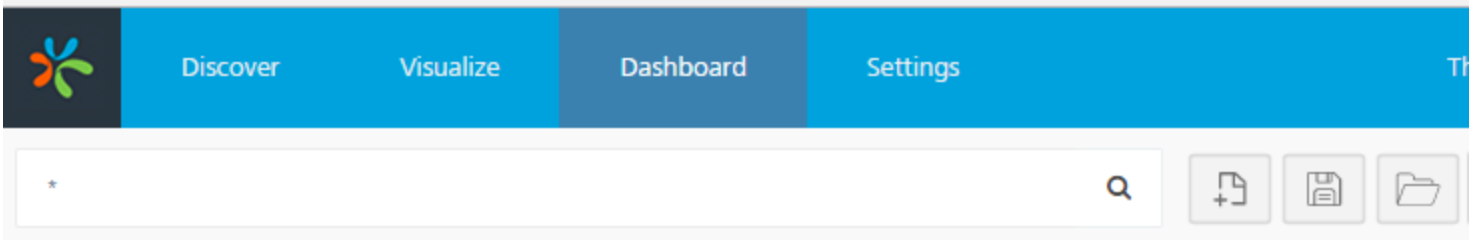
4. **Visualizations**: This tab displayed the list of saved visualizations. Select the visualizations to add them to the dashboard.
5. **Searches**: This tab displays the list of saved searches. Select the saved search to add them to the dashboard page.



6. **Manage Searches** : This option displays the list of saved searches, visualizations and dashboards.
7. You can edit, delete and view the saved items by selecting each item.
8. Once the dashboard is created, click **Save** to save the dashboard.

47.9.3.1 Search

You can search for the dashboards by providing a keyword in the Search field. Following actions can be performed in the Search pane.



- **New Dashboard**: Click the icon to create a new dashboard.
- **Save Dashboard**: Click the icon to save the created dashboard.
- **Load Dashboard**: Click the icon to load the saved dashboards.
- **Share**: Selecting this option will provide you two links where one link can be embedded to HTML source and the other link can be shared.

  > ***Note:*** The client embedding the link to their source code should have access to Kibana.

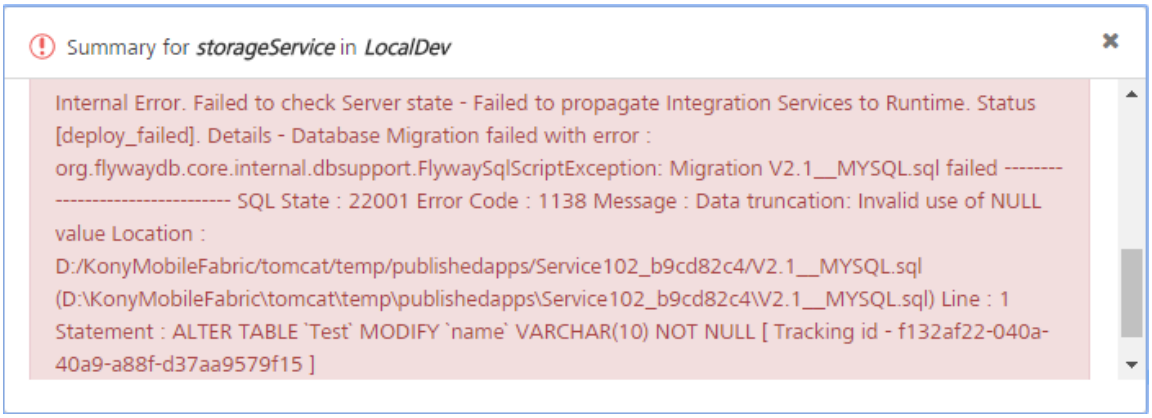
- **Add Visualization**: Click **Add Visualization** to add a saved visualization to the new dashboard.

48. Appendix - Frequently Asked Questions (FAQs)

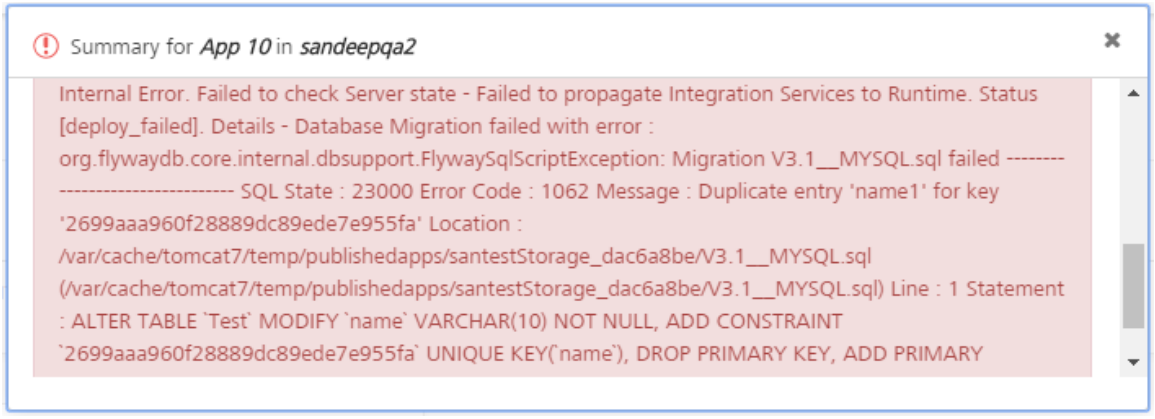
- **Issue**

  Republish of app with modifications in storage services fails.
  The following are a few of the scenarios that cause republish fail of an app:

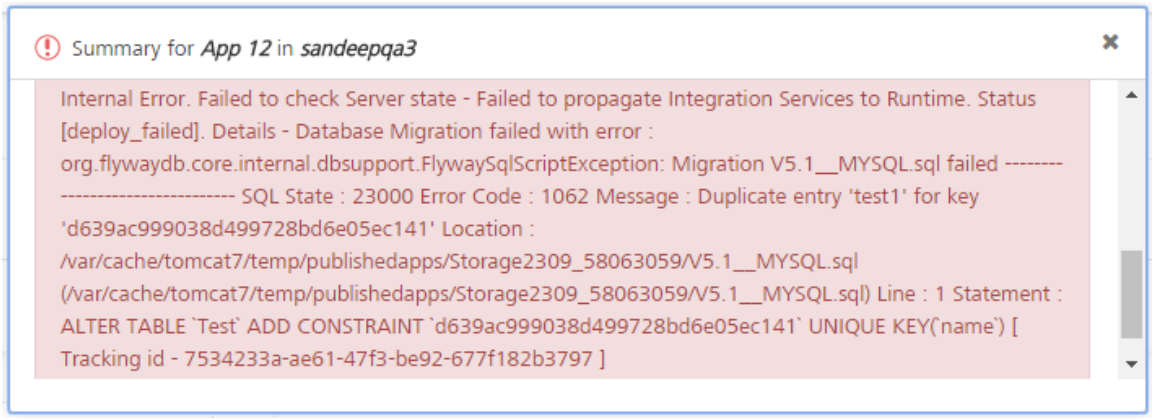
  - If you change a field nullable to **FALSE** after there is some data with null values in that field, republish fails.



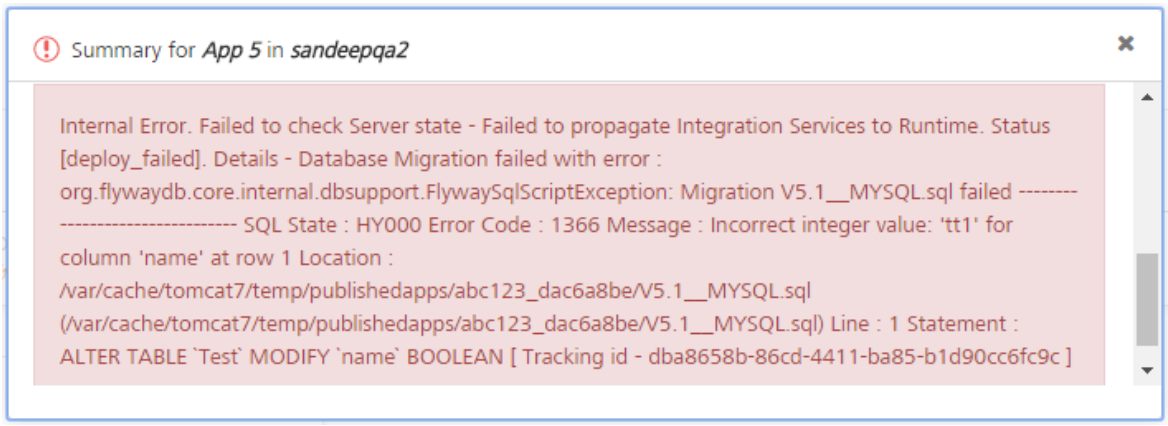
  - If you change the **primary key** of a field, which has duplicate values to **TRUE** and republish, republish fails.



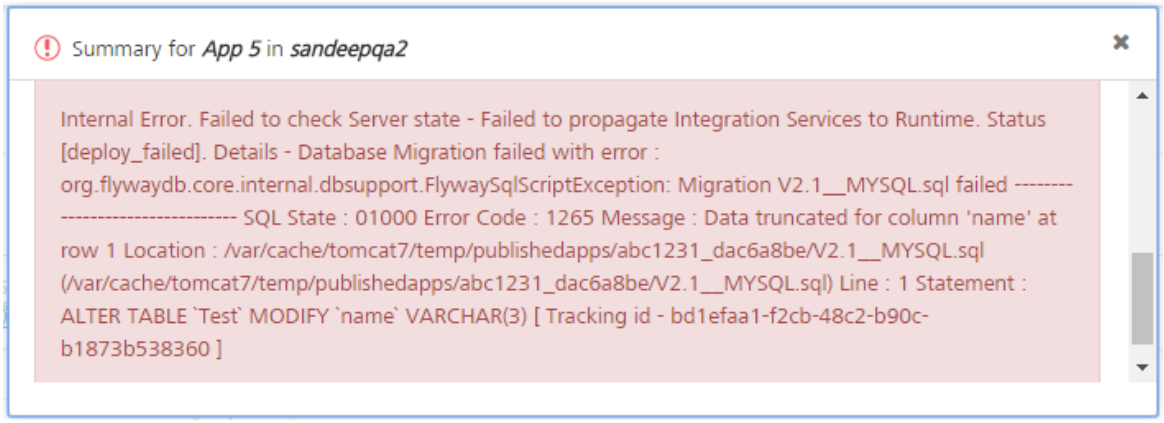
- If you change **unique** to **true** when already there is data in that field with duplicate values, republish fails.



- If you change **datatypes** (for example: String to Boolean, Number to Date) and there is some data in that field, republish fails.



- If you reduce the **size of a string column** if the table already has data that exceeds the changed value and republishing, republish fails.



- If you have more than one field that has auto-generated property set to **TRUE** in an object (MySQL InnoDB engine limitation).

Workaround:

1. Export the app.
   For more details, refer to Exporting and Importing an Application.
2. Check the version for script.
3. Import the modified app into the app.

   > ***Important:*** While importing an app to an existing app, if the app names are same, the system overrides the existing data with new data in the imported .zip file. The app name will not be changed.

4. Publish the app.

- **Issue**

  While launching Kony Fabric apps from Internet Explorer 8 and Internet Explorer 9, the `init` fails to load the apps.

  Workaround:

Enable domain settings to access the identity service in your Kony Fabric account.
For more details to enable CORS, refer to API Management > Identity > SERVICE CONFIGURATION > Identity Service Cross Domain Security (CORS).

- **Issue**

  Kony Fabric Console is unable to explore MySQL stored procedures from integration service for WebLogic with MSSQL,Oracle as consoleDB. Because WebLogic loads 5.1.22 mysql jar by default. (This jar is older version).

  Workaround:

  You can resolve this issue by any of the following two ways:

  - Update the mysql jar in WebLogic itself. For more details, refer to https://docs.oracle.com/middleware/1212/wls/JDBCA/third_party_drivers.htm#JDBCA233

    Or

  - Configure the new mysql jar path in the required war files. This workaround it recommended by Kony.

  To configure mysql (com.mysql.jdbc) jar path in WebLogic.xml, follow these steps:

  1. Open the WebLogic.xml file. For example, WEB-INF->INF->Weblogic.xml.
  2. Add the following property in the four war files (mfconsole.war, middleware.war, admin.war, and services.war) located in Weblogic.xml.

     ```
     <package-name>com.mysql.jdbc.*</package-name>
     ```

- **Issue**

  When auth and Visualizer are in different nodes and they are not in time sync, auth gives 401 response for `request_token` call but does not give any valid reason for the failure.

  Workaround:

  Check if the Visualizer host time is set correctly.

- **Issue**

  Connection Issues While Creating a Salesforce Service Using Kony Fabric When you create a Salesforce service by providing a wrong **client ID**, the system throws the following exception:

  ```
  Error retrieving the Salesforce metadata:
  {"error_description":"invalid client
  credentials","error":"invalid_client"};
  ```

  Workaround: Refer to Creating Connected Application.

- **Issue**

  When you create a Salesforce service by providing a wrong **secret key**, the system throws the following exception:

  ```
  Error retrieving the Salesforce metadata:
  {"error_description":"client identifier
  invalid","error":"invalid_client_id"};
  ```

  Workaround: Refer to Creating Connected Application.

- **Issue**

  When you create a Salesforce service by providing a wrong **password with suffix**, the system throws the following exception:

```
 Error retrieving the Salesforce metadata:
{"error_description":"authentication failure - Invalid
Password","error":"invalid_grant"};
```

  Workaround: Refer to Configuring Salesforce Account.

- **Issue**

  When you create a Salesforce service by providing a wrong **password without suffix**, the system throws the following exception:

```
 Error retrieving the Salesforce metadata:
{"error_description":"authentication failure - Failed: API
security token required","error":"invalid_grant"};
```

  Workaround: Refer to Configuring Salesforce Account.

- **Issue**

  When you log into Salesforce by providing a wrong **UserName**, the system throws the following exception:

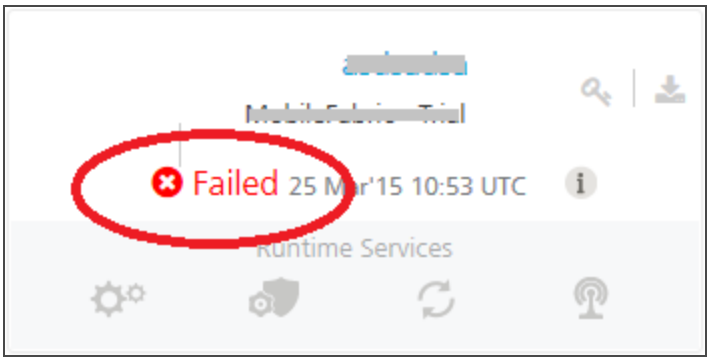
```
 Error retrieving the Salesforce metadata:
{"error_description":"expired access/refresh
token","error":"invalid_grant"};
```

  Workaround: Refer to Configuring Salesforce Account.

48.1 Issues Publishing an Application Using Kony Fabric Console

- **Issue**

  While publishing, an app goes to the `InProgress` state, and the status for the app changes into `Failed` only after 20 minutes. The user is not allowed to perform the publish operation on this app during those 20 minutes.



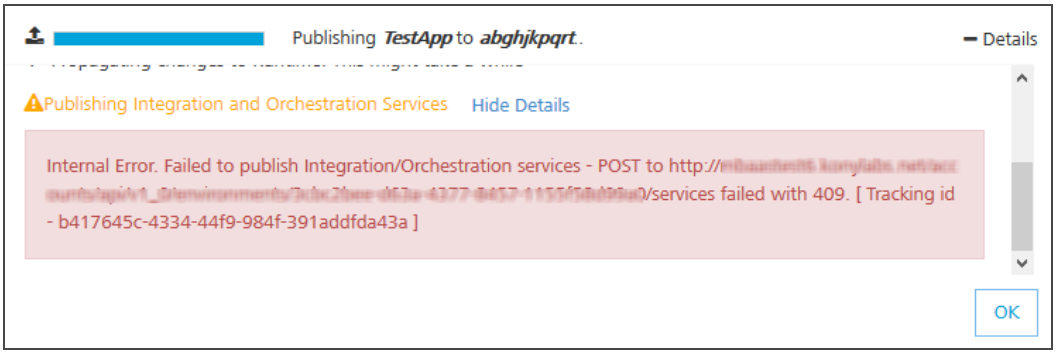
  An app publish status can go to the `InProgress` state based on factors such as, a browser refresh, a request timeout, and network issues.

  The InProgress/Failed issue is specific to publishing behavior.

- **Issue**

  When another deployment is in progress for the server, the system throws the following error:

```
409 -Failed to Publish Integration and Orchestration Services.
```



Workaround: You must wait until an app in the queue is published successfully, and then re-publish the app.

- **Issue**

  - If the size of your application is more than the variable size of the server database, an error occurs while publishing. Ensure the variable `max_allowed_packet_size` is set to a higher value than the application size.

  - While saving a jar file, the system throws the error message: `Size of jar file is too large.`

  - While saving a client binary file, the system throws the error message: `Size of Client Binary is too large.`

  Workaround:

  Increase the network packet size value for DB server. For example, to set max_allowed_packet for MYSQL, refer to http://docs.oracle.com/cd/E19509-01/820-6323/gicxk/index.html

  For example, if your application size is 1024 kilobytes (1MB), and you try to publish a 2048 kilobyte (2 MB) application from Kony Visualizer, an error appears while publishing. Increase the size of the server database to a value more than 2048 kilobytes (2 MB) for publishing the application.

- **Issue**

  An app is created with the same name as an integration service or orchestration service. When these services are published from Kony Fabric Console and the app is published from Kony Visualizer, the app will override these services or these services will override the app.

  Workaround

  While creating an integration service or orchestration service in Kony Fabric Console and an app in Kony Visualizer, use different names for integration and orchestration services, and app names.

  For example, you can name an integration service as `SampleIntegrationService`, an orchestration service as `SampleOrchestrationService`, and Kony Visualizer app name as `MySampleAppName`.

- **Issue**

  Uploading huge `Jars/binary files/wsdl` files fails with "Size of the <..> file is too large"

  - BLOB content is sent over the wire from Kony Fabric to MySQL Serve in hexadecimal encoding. Hence the size of the MySQL query doubles. There is a limit to the packet size of the MySQL query. Hence adjusting the packet size to accommodate the whole binary would solve the problem.
  - Redo log writes for large, externally stored BLOB fields could overwrite the most recent checkpoint. The 5.6.20 patch limits the size of redo log BLOB writes to 10% of the redo log file size. The 5.7.5 patch addresses the bug without imposing a limitation. For MySQL 5.5, the bug remains a known limitation. As a result of the redo log BLOB write limit introduced for MySQL 5.6, innodb_log_file_size should be set to a value greater than 10 times the largest BLOB data size found in the rows of your tables plus the length of other

variable length fields (VARCHAR, VARBINARY, and TEXT type fields). Failing to do so could result in “Row size too large” errors. No action is required if your `innodb_log_file_size` setting is already sufficiently large or your tables contain no BLOB data.

Related Links: https://dev.mysql.com/doc/refman/5.5/en/packet-too-large.html http://stackoverflow.com/questions/25246074/row-size-too-large-8126-exception-on-adding-appcenter-application

Workaround

Change the following MySQL Server variables:

- Change the `max_allowed_packet` to double the size of the binary with maximum size you want to support.
- Change the `innodb_log_file_size` to a value greater than 10 times the largest BLOB data size found in the rows of your tables plus the length of other variable length fields.

- **Issue**

  When you try to test an operation with preprocessor or postprocessor that has a logic to call another service, the Test operation fails during design time. This is a known issue.

  Workaround

  Publish the app and call the service from the device. After republishing the app, the app successfully works with preprocessor or postprocessor that has a logic to call another service.

- **Issue**

  When you create an integration for `WSDL`, the service fails with `NoClassDefFoundError`.

Workaround

While creating a SOAP service by uploading `WSDL`, ensure that all the referenced `WSDLs` and `XSDs` have path to URLs and not to any local file system.

- **Issue**

  If EMM is installed on Tomcat and Apache Server, and Kony Fabric installed on WebSphere 8.5.5.5 or 8.5.5.6, and when you attempt to add EMM in the **Environments** tab in Kony Fabric Console, the system throws the error into `accounts.log` and will display following stack-trace. This happens because some older versions of some of the IBM libraries were not able to handle `Diffie Hellman with 2048 bit key`.

  ```
  javax.net.ssl.SSLException

  java.lang.ArrayIndexOutOfBoundsException: Array index out of
  range: 64
  ```

  Workaround

  To fix the issue, update to the newer versions of `ibmjsseprovider2.jar` and `ibmjceprovider.jar`.

  For more details, refer to IBM updates to these libraries as noted in the **Problem Conclusion** section at http://www-01.ibm.com/support/docview.wss?uid=swg1IV74069

- **Issue**

  Unable to read metadata from an integration service (for example, SOAP) that is protected by an SSL certificate.

  Workaround

When you configure an integration service that is protected by an SSL, enable the integration service by importing the SSL into your cacerts in your Kony Fabric install location.

To import an SSL certificate of an integration service, follow these steps:

1. Download the required SSL certificate.
2. Import the SSL certificate into your `cacerts` of your Kony Fabric install location.

   For example:

   ```
   keytool -import -alias somealias -file <cert-file> -
   keystore C:\KonyFabric\jre\lib\security\cacerts
   ```

- **Issue**

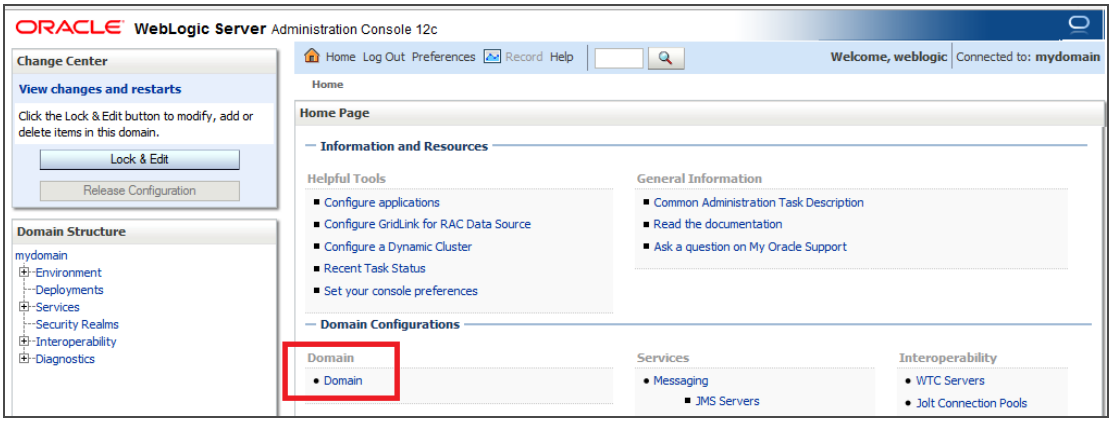
  If your Kony Fabric is installed on the WebLogic Server, when a user builds an app in IDE and tries to publish the app, the system fails to deploy the app.

  Workaround

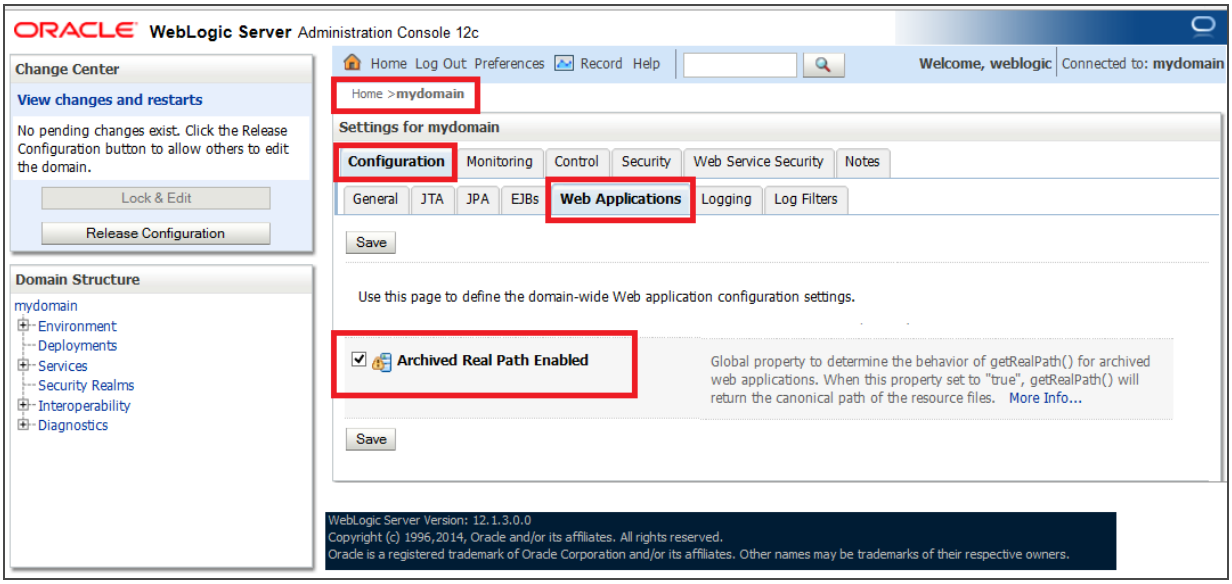
  Enable the **Archived real path Enabled** property in the WebLogic Console.

  To enable Archived real path Enabled, follow these steps:

1. Log in to **WebLogic Console**.
2. In the **Home** page, click the **mydomain** link.



The **Settings for mydomain** details appear.



3. In the **Settings for mydomain** section click the **Configuration** tab, and then click the **Web Applications** tab.
4. In the **Web Applications** section, select the **Archived Real Path Enabled** check box.
5. Click **Save** to save the configuration.

6. Restart the WebLogic Server.
7. Republish the app.

49. Limitations

- Kony Fabric Engagement is not supported for the BlackBerry platform.
- PhoneGap with Windows is not supported.
- A service parameter value should be enclosed in double quotation marks (“”) if the value is a string.

49.1 Kony Fabric app name should not be same as of Kony Visualizer app

- **Issue**

  An app, with Web client name and other services, is created with the same name in Kony Fabric and Kony Visualizer and the app is published from Kony Visualizer. When you publish the same app from Kony Fabric, the services from Kony Fabric will be overridden or deleted.

  Workaround

  Ensure that the Web client name and service name should not be the same. While creating apps in Kony Fabric and Kony Visualizer, use different names for Web client name and service name.

50. Index

A

D

E

F

J

K

L

M

N

O

P

R

S

W

X