Kony Visualizer

User Guide

Release V9

Document Relevance and Accuracy

This document is considered relevant to the Release stated on this title page and the document version stated on the Revision History page. Remember to always view and download the latest document version relevant to the software release you are using.

Document Relevance and Accuracy

This document is considered relevant to the Release stated on this title page and the document version stated on the Revision History page. Remember to always view and download the latest document version relevant to the software release you are using.

March, 2020

Revision History

For Kony Visualizer V9 GA

Date	Document Version	Description of Releases and Updates
10-03-2020	1.0	Document updated for Kony Visualizer V9 release. This update includes the following new and revised features: • Support for Responsive Web Components • Writing the Test Plan file • Build a Windows Native app on a Local Machine • Export and Delete Custom widgets • Quantum IQ Enhancements • Suggestions from Kony Library

For Kony Visualizer V9 P2

Date	Document Version	Description of Releases and Updates

17-02-2019	1.0	Document updated for Kony Visualizer V9 release.
		
		This update includes the following new and revised features:
		
		• Support for KSLint
		• Workflow Service support in the Data and Services Panel
		• Enhancements to Jasmine Test Automation
		  ◦ Modifying the Kony Automator port
		  ◦ Edit Test Case manually
		• Resolve Action Conflicts in Bulk
		• Project Settings Enhancements
		  ◦ Android Wear Settings
		  ◦ Platform Settings for iOS
		  ◦ Adaptive Web Settings
		  ◦ Disable Print Statements in builds
		• Import custom web widget to Kony Quantum
		• Removed references to Windows Phone (Deprecated) channel.

For Kony Visualizer V9 Preview

Date	Document Version	Description of Releases and Updates
12-12-2019	1.0	Document updated for Kony Visualizer V9 release. This update includes the following new and revised features: • Connect Kony Visualizer to an on-premise Kony Fabric Environment • Resolve Conflicts in Designer and Developer actions • Generate an app binary for the Kony Quantum App • Build Native Apps on a Local Machine • Added support for the WebP format for Images • Caching in Progressive Web Apps • Generate CRUD Forms for an Object Service • Generate Object Services UI for Responsive Web • Customize the Generation of Data Model Objects • Create the Storyboard of your App

Date	Document Version	Description of Releases and Updates

12-12-2019	1.0	Document updated for Kony Visualizer V9 release.
		
		This update includes the following new and revised features:
		
		• Kony Collab
		• Dashboard in Jasmine Test Framework
		• Quantum IQ Enhancements
		  • Pausing design suggestions
		  • Enabling/Disabling design suggestions
		  • Creating a component from design suggestions
		• Action Editor Enhancements:
		  • Search for an Action in Action Editor
		  • Consolidated Network Service Actions
		• Environment lists in Visualizer
		• Create a User-defined Template from an Auto-generated Segment Template
		• Map Section Header Template Widgets by using Mapping Editor
		• Set Data for the Segment in a Component
		• Set Data for Components with Contract by using Mapping Editor

Date	Document Version	Description of Releases and Updates

30-09-2019	1.0	Document updated for Kony Visualizer V9 release.
		
		This update includes the following new and revised features:
		
		• Project Creation Wizard modifications:
		  • Create a Kony Reference Architecture project
		  • Create a Free Form JavaScript project
		  • Create a project from a sample app
		• AndroidX Behavioral Changes
		• Manually Customize the Cordova-Generated Android Project
		• Bundle a Customized Cordova-Generated Android Project

Date	Document Version	Description of Releases and Updates

30-09-2019	1.0	Document updated for Kony Visualizer V9 release.
		
		This update includes the following new and revised features:
		
		• Data & Services Panel enhancements:
		  • Enhanced discoverability of Data Storage Object Services
		  • Disable/Enable the Categorization of Project Services
		  • Generate Object Services UI for Responsive Web
		  • Generate CRUD Forms for an Object Service
		  • Create New Data Table Objects
		  • Support for Enumeration (enum) Data Type in Object Service Data Model
		  • Configure an Object Data Model by Importing an Excel File
		  • Shifting of the Kony Fabric node from Project Explorer to the Data & Services panel
		  • Customize the Generation of Data Model Objects

Date	Document Version	Description of Releases and Updates

30-09-2019	1.0	Document updated for Kony Visualizer V9 release.
		
		This update includes the following new and revised features:
		
		• Breakpoint Forking Limitation while designing a Responsive Web app
		• Action Editor Enhancements
		• App Logo for Applications on the Enterprise App Store
		• Push Notifications for Cloud Build
		• Responsive Support for Collections
		• Swipe functionality for Segment widget
		• Jasmine test automation
		• Quantum IQ enhancements:
		  • Hike Search
		  • Language Translation

Date	Document Version	Description of Releases and Updates
30-09-2019	1.0	Document updated for Kony Visualizer V9 release. This update includes the following new and revised features: • Added support for terminal inside Visualizer by using Quantum IQ • Added support to identify duplicates of a design • Added support to debug in Android while using Jasmine test scripts • Create the Storyboard of your App • Android Properties: • Location Listener Type

Table of Contents

Visualizer User Guide

This document explains how to use Kony Visualizer, an integrated, intuitive development environment, to build omni-channel digital applications – all from a single code base. Its unified, shared design environment makes it easy to publish and share native prototypes and app designs with real-time app previews and collaboration. And that's just the beginning. Kony Visualizer serves as the front end of Kony's powerful, multi-channel JS API framework that enables designers and developers to securely deploy across a multitude of devices ranging from smartphones, tablets, desktops, and wearables on iOS, Android, and Windows operating systems.

Kony Visualizer is a powerful yet easy-to-use integrated development environment (IDE) for developing, building, testing, debugging, and deploying omni-channel digital applications (apps) for multiple platforms—all from a single code base. Kony Visualizer empowers you to rapidly develop digital apps while giving you the flexibility to integrate with back-end services as and when you need them. What's more, you can integrate into your app the very best of the native, web and hybrid environments.

Platform Compatibility Chart

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

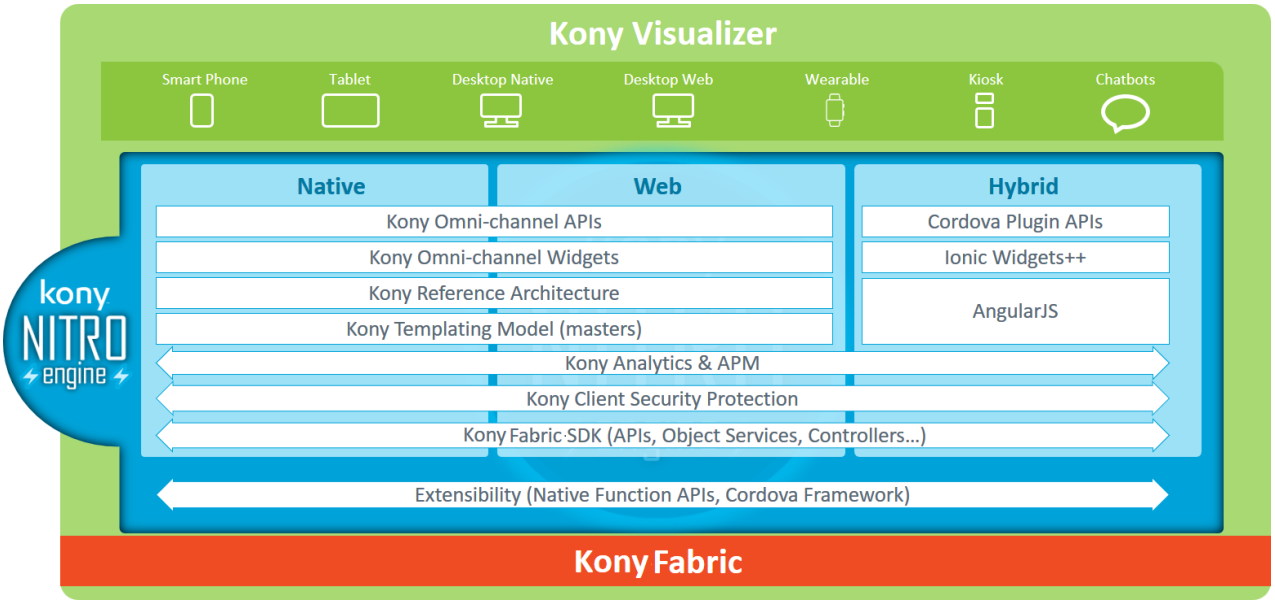
Product Version	Compatible With
Kony Fabric N (For example V8)	Kony Visualizer N (For example V8) Kony Visualizer N-1 (For example, V8 and 7.3)
Kony Visualizer N (For example V8)	Kony Fabric N (For example V8) Kony Fabric N-1 (For example, V8 and 7.3)

Kony Nitro

We call this approach Kony Nitro™, a patented strategy for omni-channel, enterprise-wide digital app development and deployment. Using Kony Visualizer along with Kony Fabric for back-end services integration, you can take advantage of multiple platforms' native APIs and tap into the web's most popular open standards such as HTML5, Cordova, and Angular, all while lowering your app development time by as much as 50% with extensive, reusable templates and objects. And if you're looking for something that's ready-to-run, you can explore Kony Apps, a portfolio of enterprise apps that offer you a best-of-both-worlds digital app solution: built-in best practices, proven business process workflows, and enterprise integrations that make it easy to get up and running fast, plus configuration options designed for cutting-edge customization.

> ***Important:*** On Native channels, Angular JS can be used through Cordova webview and through Kony Visualizer Browser Widget local file.
>
> For SPA and DesktopWeb channels, you can use Angular JS in Kony Visualizer Custom widget and through the Kony Visualizer Browser Widget local file.

So whether you're seeking speed, power, or choice in your app development–or all three–you can find what you need with Kony Nitro™.



Kony Visualizer and Kony Visualizer Classic

This user guide provides documentation for both *Kony Visualizer* and *Kony Visualizer Classic*. Kony Visualizer Edition, which is free to download and use, is an integrated, intuitive development environment for prototyping and building omni-channel digital applications—all from a single code base. Enterprise Edition, in addition to the design environment found in Kony Visualizer Edition, is integrated with Kony Fabric, giving you the ability to create robust back-end services for powerful, scalable, omni-channel, enterprise-wide digital app solutions.

Unless otherwise noted, you can assume that the content in this user guide applies to both editions of Kony Visualizer. If a given topic applies only to *Kony Visualizer*, the headline for the topic is followed by the phrase, "Applies to *Kony Visualizer*." Conversely, if a given topic applies only to *Kony Visualizer Classic*, the headline for the topic is followed by the phrase, "Applies to *Kony Visualizer Classic*."

Widgets – the building blocks of your digital app

Within Kony Visualizer's visual environment, you piece together discrete functional building blocks called widgets. Widgets range from the simple, such as a button or input field, to the complex, such as a Cordova Browser widget, or a 3D image with complex rotation and transform animations. All widgets have attributes and properties that you configure to suit the needs of your app, such as excluding or including a given widget based upon which device the app runs on.

Skins, Themes, and Components

To give your app a particular look, feel, and set of behaviors native to a platform, you can use ready-made skins. You can customize these to make them your own, or build them from scratch. You can even group skins together as a collection called a *theme* to maximize visual uniformity across devices. To reduce your app development time, you can create components to group and configure user interface elements and action sequences in a single definition, and then instantiate them anywhere throughout an application. You can publish your components on Kony Marketplace, or download third-party components from Kony Marketplace and use them in your applications.

Back-end services

To integrate your apps with back-end services, Kony Visualizer communicates directly and securely with your existing data system, writing the connection once and then applying it to all devices. Kony Visualizer also supports multiple classes for generating markup based on device capabilities, and devices can receive WML, cHTML or a combination of XHTML, CSS, and JavaScript based on capabilities detected at runtime.

Kony Studio Equivalents in Kony Visualizer

The Kony Studio product line has evolved to a new user interface and expanded functionality in Kony Visualizer. In addition to its many new and improved features, virtually all of the same functionality is available, but many of those familiar Kony Studio features are accessed differently in Kony Visualizer. This topic shows you how to accomplish in Kony Visualizer what you are familiar with doing in Kony Studio.

> ***Note:*** In Kony Studio 6.5, you could port a project from one platform to another, and to channels within the target platform. However, issues would arise due to the wide variations in form factors from one platform and channel to another. In Kony Visualizer, universality across platforms and channels is achieved through the use of components and masters; platform porting is not supported in Kony Visualizer. For more information, See Working with Components, Kony Marketplace, and Masters. .

Feature Locations

The following table outlines where different features are located in the respective user interfaces of Kony Studio and Kony Visualizer.

Feature	In Kony Studio	In Kony Visualizer
Application assets	Navigator View → <App Name>	Project Explorer → Assets tab
Clone All	Skins View → Clone All icon	Project Explorer → Skins tab → <widget type> → right-click <skin name> (e.g. btnFocus) → Fork

Collections	N/A	Library Explorer → Collections
Custom Widgets	Navigator View → <App Name> → customwidgets	Kony Library pane →Widget tab → Custom Widget section
Emulators (configure)	Emulators View → <Platform> → <Channel> → right-click emulator → Edit	Product menu → Emulators & Devices Configuration or Window menu → Preferences → Kony Visualizer → Emulators
Emulators (display)	Window menu → Show View → Emulators	Preview menu → Launch Emulator
Events/Actions	Events are added and edited by selecting a widget and then navigating to the Event section in the Widget Properties View	Global actions can be created and accessed from Project Explorer → Project tab → <Channel>, and open in the Action Editor Actions created for a specific widget are accessed by selecting the widget, followed by Properties Editor → Action tab
Flex Properties	Flex Properties View	Properties Editor → Look tab → Flex section
Font (change)	Skins view → navigate to widget type → navigate to channel and platform → double-click → Font tab	Select widget → Properties Editor → Skin tab → Fonts section
Fork a Skin	Skins View → <widget type> → right-click <skin name> (e.g. btnFocus) → Fork	Project Explorer → Skins tab → <widget type> → right-click <skin name> (e.g. btnFocus) → Fork or Properties Editor → Skin tab → General section → Platform ellipsis button→ Select platforms → OK
Forms	Applications View → <App Name> → forms	Project Explorer → Project tab → <Channel> → Forms

Hierarchical view of forms and widgets	Outline view	Project Explorer → Project tab
Images	Applications View → resources	Project Explorer → Assets tab
Modules	Applications View → <App Name> → modules	Project Explorer → Project tab → Modules
Offline Services	Applications View → <App Name> → offline services	Set up as a service on Kony Fabric that is then synced locally Project Explorer → Kony Fabric
Popups	Applications View → <App Name> → popups	Project Explorer → Project tab → <Channel> → Popups
Projects/Applications	Multiple applications can be open at one time	Only one project can be open at a time
Review/Comments	Not Available	Properties Editor → Review tab
Search (forms, widgets, and skins)	Search menu → Search	Project Explorer → Search tab
Search (modules)	Search menu → Search	Edit menu → Find/Replace (Search in Kony Visualizer)
Services in use by the application	Applications View → <App Name> → services Also: The Services View	Project Explorer → Kony Fabric
Skins	The Skins view	Project Explorer → Skins tab
Splash Screen	File → Application Properties → Splash Screen tab	Opened from Project Explorer → Project tab → <Channel>, and once opened, configured from Properties Editor → Splash Screen tab
Templates	Applications View → <App Name> → templates	Project Explorer → Templates tab → Components
Themes	The Skins view → Theme drop-down list	Project Explorer → Skins tab → Theme drop-down list

Web_module	Applications View → <App Name> →web_module	Handled as a part of services in the Kony Fabric Console Project Explorer → Project tab → Kony Fabric
Widgets	Widgets Palette	Library Explorer → Widgets tab
Widget Properties	Widget Properties View	Properties Editor

Service Definition

The following table outlines where service definition features are located in the respective user interfaces of Kony Studio and Kony Visualizer.

Feature or Task	In Kony Studio	In Kony Visualizer
Access and edit service definitions	Right-click app name in the Applications View, and then click **Open Service Definition**	Project Explorer → Project tab → Kony Fabric → Integration → Configure New button or Use Existing button
Sync Configuration	Window menu → Show View → Other → Kony Studio → SyncConfiguration	Project Explorer → Project tab → Kony Fabric → Synchronization

Form Editor

The following table outlines where various form editing features are located in the respective user interfaces of Kony Studio and Kony Visualizer.

Feature or Task	In Kony Studio	In Kony Visualizer
Create a new form	Applications View → <App Name> → forms → right-click <channel> → New Flex Form	Project Explorer→ Project tab → <channel> → right-click Forms → New Form → Flex Form
Form editing	Uses the Form Designer	Uses the Visualizer Canvas
Mapping Editor	Open the Event Editor → right-click Action Sequence → select to invoke a service, navigate to a form, or add mapping → Open Mapping Editor	Open the Action Editor → open an Action Sequence → right-click an action → Open Mapping Editor

Open a form	Applications View → <App Name> → forms → <channel> → double-click form	Project Explorer→ Project tab → <channel> → click Forms arrow → Click Form
Quick Preview	Click Preview in the Form Designer → select the platform and channel → open Emulators View → right-click emulator → click Open in Preview	The Visualizer Canvas is always in preview mode. From the drop-down lists at the top of the canvas, select the platform, channel, and device you want
Side-by-Side view	Not available	Window menu → Arrange → Side By Side
Toggle BVR (beyond visible range) Toggles between limiting what's viewable on the canvas to what a user would see on the screen, and displaying all application elements of a form, regardless of their position	Not available	On the Visualizer Canvas, click BVR to place and view application items beyond what's visible on the device screen. Click BVR again to limit the canvas display to the device screen To pan in BVR mode, you can press the space bar and drag.
Toggle Orientation Toggles the device preview between portrait and landscape orientations	Not available	At the top of the Visualizer Canvas, click the Toggle Orientation icon
Toggle Shell Toggles the device preview between displaying just the screen, and displaying the device's shape and dimensions beyond the screen	Not available	At the top of the Visualizer Canvas, click the Toggle Shell icon

Authentication and Licensing

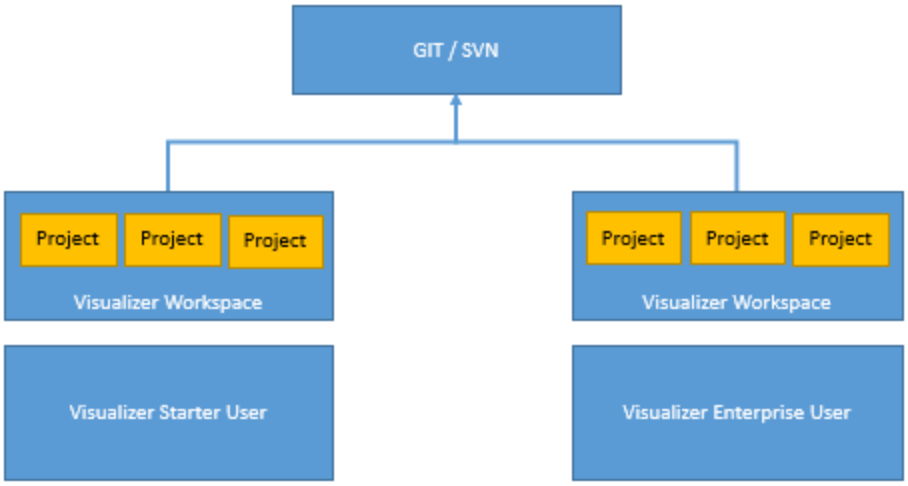
In Kony Studio, you were prompted for authentication and licensing when Kony Studio started. With Kony Visualizer, you are prompted for authentication and licensing when you first build a project, giving you the ability to use Kony Visualizer offline.

Best Practices for using GIT and SVN Repositories

In your development streams, we recommend the following best practices in coordinating the use of both Kony Visualizer and Kony Visualizer Classic.

- Since both editions of Kony Visualizer use the same file format, they can share a common GIT or SVN repository.
- You should maintain separate workspaces for Kony Visualizer projects and Kony Visualizer Classic projects.

To illustrate, your workspace work flow should look like this:



Import and Export

The following table outlines how import and export features function in the respective user interfaces of Kony Studio and Kony Visualizer.

Feature or Issue	In Kony Studio	In Kony Visualizer
File Format Compatibility	Since Kony Studio used the .kl file format and Visualizer uses the JSON file format, moving a project between one and the other required converting between the two formats, resulting in inconsistencies	Both Kony Visualizer: Kony Visualizer and Kony Visualizer Classic: Kony Visualizer Classic uses the JSON file format for seamless importing and exporting of projects between the two.
Importation of actions from earlier versions of Visualizer	When imported into Kony Studio, did not copy the actions of projects made with earlier versions of Visualizer	The "Designer Actions" created in earlier versions of Kony Visualizer, when imported into Kony Visualizer Classic, are replicated as "Developer Actions" for seamless integration and optimal functionality
Imported Assets	Imported project is copied by reference but is not actually added to the workspace	Imported project is copied to the workspace

Using Skins

The following table outlines differences in how skins are used in the respective user interfaces of Kony Studio and Kony Visualizer.

Feature or Task	In Kony Studio	In Kony Visualizer
Assigning skins	Assigned from the Widget Properties through a multi-click process	Assigned from the Skin tab of the Properties Editor for improved work flow
Common skins vs. widget-specific skins	Widgets, when added, automatically assume the default, common skin.	Widgets, when added, are automatically assigned their own unique, widget-specific skin

Utilities

The following table describes the differences in certain utilities between Kony Studio and Kony Visualizer.

Feature	In Kony Studio	In Kony Visualizer

Build Diff Tool	Applications View → right-click App name→ Utilities → Launch Build Diff Tool	Deprecated. Obsolete in Kony Visualizer
Form Merge Tool	For comparing forms and porting elements from one form to another to create functional parity across forms	This tool is deprecated since any form can be easily duplicated
Manage Custom Fonts	Skin View → right-click skin → Edit → Font tab → Platform Specific Font Names	Copy fonts directly into the project's Fonts folder
Platform Porting	For porting an application created in one platform to another platform	Deprecated. With flex layouts, Kony Visualizer dynamically renders from one platform to another, and can be copied seamlessly to other channels

Custom Widget Import

In Kony Studio, custom widgets could be exported as a .zip file and then imported into a different project. This functionality is not available in Kony Visualizer. As an alternative, you can add custom widgets. For more information, see Add Custom Widgets.

Code Editor

The following table illustrates a couple of the main differences in how the Code Editor functions between Kony Studio and Kony Visualizer.

Feature or Task	**In Kony Studio**	**In Kony Visualizer**
Code outline	Displayed in a separate panel	Displayed inline
Syntax Highlighting	Partially supported	Supported for the following languages: C, C#, C++, CSS, HTML4, HTML5, Java, JavaScript, LESS, Objective C, Python, Ruby, Sass, SCSS, XML

Search and Replace

In Kony Visualizer, you can search and replace code, and you can also jump to a code element's definition, such as a function. For more information, see Find and Replace and Jump to the Definition of a Code Element.

Local Preview

The following table outlines the differences in how previews are handled locally between Kony Studio and Kony Visualizer.

Feature or Task	In Kony Studio	In Kony Visualizer
Functional Preview Requests	Handled by Jetty	Handled by the Node.js runtime environment
Publish Destination	KonyServer	Kony Fabric
Publish Model	Applications and services published individually	In Kony Fabric, you create a Kony Fabric app to which you add services that you publish. When you're ready to publish the Kony Visualizer app, you bind it to the Kony Fabric app that contains the services that you want to use. After publishing the Kony Visualizer app, it accesses the published Kony Fabricservices.

Supported Digital Channels

Kony Visualizer allows you to build applications for several digital channels.

The following table lists the various platforms that are supported for each channel.

Channel	Platforms	Remarks
Mobile	• iPhone • Android • Windows	The application exists on the mobile device and is accessed from the device.
HTML5 SPA	• iPhone • Android • Windows	
Tablet	• iPad • Android • Windows • HTML5 SPA	
Watch	• Apple Watch	

Channel	Platforms	Remarks
Desktop Web	• Desktop Web	A Desktop Web application runs on a desktop as well as on the web browsers of a desktop. Desktop Web apps support the following major features: • Responsive Web: Enables your app to automatically resize and fit the content for different-sized layouts, with the use of breakpoints. This creates a glitch-free browsing experience. • PWA: Progressive Web Apps leverage the Responsive Web feature. A Progressive app feels like a Native app, but is available over all web browsers. This type of app works seamlessly even when the device is offline or has limited network speed.

Hikes

Hikes is a mechanism within Kony Visualizer that you can use to learn a few concepts of how to develop applications by using Visualizer. Hikes are step-by-step interactive walkthroughs that help you understand the basics of using Kony Visualizer.

> ***Note:*** Hikes are supported only on Kony Visualizer, and not on Kony Visualizer Classic.

Access Hikes in Kony Visualizer V8 SP3

You can access Hikes from the left navigation panel in Kony Visualizer V8 SP3. When you click the Hikes icon, a list of the available Hikes appears. Click the Hike that interests you.

A few UI elements are displayed as part of the Hike procedure to help you to navigate and perform the required steps. Arrows point to the area where you must focus to perform the action in a Hike step. For some Hike steps, after you successfully perform an action, the card automatically moves to the next step.

Once you have completed a Hike, you can either move on to the next Hike or close the current Hike. You can always access the list of Hikes from the navigation panel (as mentioned earlier) and start any Hike.

Access Hikes in Kony Visualizer V8 SP4

You can access Hikes from the left navigation panel in Kony Visualizer V8 SP4. Hikes are organized into Guided Tours in a catalog, and the Hikes are in a sequence to take you through a learning path. You should go through the Hikes in the specified order.

A few UI elements are displayed as part of the Hike procedure to help you to navigate and perform the required steps. Arrows point to the area where you must focus to perform the action in a Hike step. For some Hike steps, after you successfully perform an action, the card automatically moves to the next step.

> ***Note:*** While using a hike, if you want to go back to a card, click the **Previous** button. If you have rapidly moved from hike **D** card to hike **A** card, you can get back to the hike **D** card by clicking the **shift key** + **Next** button.
>
> To move directly to the first card on any hike, click **shift key** + **Previous** button.

Once you have completed a Hike, you can either move on to the next Hike or close the current Hike. You can always access the list of Hikes from the navigation panel (as mentioned earlier) and start any Hike.

Third-Party Licenses

This section acknowledges the third-party companies for the license of their products.

The third-party license in this section is for the *Code Assistance* and *Inline Debugger* features of Kony Visualizer.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Configuring Your Computer

This part of the Kony Visualizer User Guide provides you with the information you need to ensure that your computer is set up to run Kony Visualizer, and is divided into the following sections. Click a section's link to view its contents.

Hardware and Software Requirements

In this document, we will walk you through the hardware and software requirements for the following:

- Installing Kony Visualizer
- Installing Kony Visualizer Classic
- Building apps for Android platform
- Building apps for iOS platform
- Building apps for Windows platform

Requirements to Instal Kony Visualizer

The hardware and software requirements to install Kony Visualizer and Kony Visualizer Classic vary based on the operating system.

General Prerequisites

- Administrative rights on your computer to install Kony Visualizer.
- For Mac, in the **Security & Privacy** > **General settings**, ensure that **Select Anywhere** option is selected.
- The available space in the system should be at least three times the size of the installer.
- The Visualizer installer requires HTTP (direct or proxy) access during the execution of the installation process.

Kony Visualizer

The hardware and software requirements to install Kony Visualizer vary depending upon your operating system. To learn about the hardware and software requirements for each platform, refer:

- Windows install guide
- Mac install guide

Kony Visualizer Classic

The hardware and software requirements to install Kony Visualizer Classic vary depending upon your operating system. To learn about the hardware and software requirements for each platform, refer:

- Windows install guide
- Mac install guide

Additional Requirements for Building Apps

Depending upon the platform you are developing your apps, requirements vary. Detailed information on how to meet these requirements is provided in the procedures for installing each platform's respective software development kit (SDK) and emulator here.

iOS

- You need a Mac OS computer to develop iOS applications
- Latest version of Xcode (the Apple SDK for creating iOS apps)
- An iOS device. To test features such as Camera, Push Notifications, and accelerometer, you need an iOS device.

Android

- Latest version of JDK
- Ensure that the default JDK install location is *<Install Drive>:\KonyVisualizerEnterprise8.x.x\Java\jdk*
- Android SDK
- Apache Ant (Another Neat Tool) - An open-source tool that automates aspects of the Android build process

- Gradle - An advanced build toolkit that manages dependencies and allows you to define custom build logic

Windows

Windows Phone 8.1

- Latest Windows Phone 8.1 SDK (Optional. To test the built application(xap) on the emulator)

Windows 8.1

- Latest Windows 8.1 SDK

Windows Desktop

- .Net framework 2.0 & 3.5 - For the Build Release mode
- .Net framework 2.0 & 3.5 4.6.1 - Required if you are using Offline Objects feature for both Debug and Release modes

Windows 10 / Windows 10 Mobile

To learn about the requirements for developing apps for Windows 10/ Windows 10 mobile devices, refer to Build Applications For Windows10.

Install Kony Visualizer

The process of installing Kony Visualizer varies depending on what edition you are installing, and on what platform. For more information, click the option that applies to you.

Kony Visualizer Installation Guide for Windows

Kony Visualizer Installation Guide for Mac

Kony Visualizer Classic Installation Guide for Windows

Kony Visualizer Classic Installation Guide for Mac

> ***Important:*** Upgrading from previous versions of Kony Studio to Kony Visualizer is not possible through the plugins. You must use the latest Kony Visualizer installer to upgrade from Kony Studio.

Install and Configure Platform SDKs and Their Emulators

Applies to *Kony Visualizer Classic*.

Having installed Kony Visualizer, you're now ready to install and configure the software development kits (SDKs) and emulators for the platforms that you intend to run your digital app on. During the Kony Visualizer installation, you're given the opportunity to download and install various SDKs. If you took advantage of that opportunity, you already have a head start on this part of the process. The instructions that follow accommodate both scenarios.

Technically, only one SDK is essential for Kony Visualizer to build a generic mobile app, and that's the Java Development Kit (JDK). If, during the installation of Kony Visualizer, a compatible version of the JDK was not detected on your computer, Kony Visualizer automatically installed the JDK. However, to develop mobile apps for other platforms, you need to install those platform's respective SDKs.

Kony Visualizer supports the following platform SDKs and their emulators. Click a platform for instructions on how to install and configure it.

iOS SDK and simulator

Android SDK and emulator

Windows SDK and emulator

iOS SDK and Emulator

Applies to *Kony Visualizer Classic*.

Building and testing iOS applications in Kony Visualizer requires two primary resources: Java SE 7 runtime, which is required by Eclipse, the hosting application of Kony Visualizer; and Xcode, the SDK for iOS and Mac OS. You are prompted to install these two resources the first time you launch Kony Visualizer.

> ***Note:*** iOS-related code and applications can only be developed on a Mac OS computer.

In all, getting your system set up to build and test iOS applications in Kony Visualizer involves five tasks. This section describes them.

1. Confirm your system meets iOS development requirements
2. Download and install Java SE 7 runtime
3. Download, install, and configure Xcode
4. Configure iOS simulators in Kony Visualizer
5. Test your set up with the HelloWorld sample app

Confirm your system meets iOS development requirements

To develop for the iOS platform and run its emulators, your computer needs to meet certain hardware and software requirements.

- Apple computer with a x86-64 CPU (64-bit Intel Core 2 Duo, Intel Core i3, Intel Core i5, Intel Core i7, or Xeon processor.)
- 150 GB of internal storage
- 4 GB of memory

- Network interface card
- Mac OS X version 10.7 and higher

Download and install Java SE 7 Runtime

Eclipse, the integrated development environment (IDE) application that hosts Kony Visualizer, requires Java SE 7 to run, but it is not automatically installed when you install Kony Visualizer. You are prompted to install it when you first launch Kony Visualizer.

To download and install Java SE 7, do the following:

1. Launch Kony Visualizer. By default it is installed in the Applications folder.
2. As Eclipse and Kony Visualizer load, a dialog box displays informing you that Java SE 7 runtime is required. Click **More info** to launch the Apple download site for Java SE 7 runtime.
3. The Apple Support website opens in the Safari web browser. Follow the prompts to download Java SE 7 runtime. It will likely have an alternate name, such as Java for OS X.
4. Once the download package has downloaded to your Mac, open the *Downloads* folder, and double-click the file you just downloaded.
5. Follow the prompts to install Java SE 7 runtime.

Download, Install, and Configure Xcode

Xcode, the Apple SDK for creating iOS apps, contains the simulators you need to emulate iOS devices in Kony Visualizer. However, it is not automatically installed when you install Kony Visualizer. It is imperative to install Xcode before working with Kony Visualizer. Kony recommends using Xcode 9. With new capabilities, Kony plug-ins are supported on Xcode 9.

> ***Important:*** You must install, at a minimum, Xcode version 7.

To download, install, and configure Xcode, do the following:

1. In a browser, navigate to the Apple Developer site, and log in to your Apple developer account. If you do not have one, create one.
2. Navigate to the Apple developer download page. The URL is as follows: `https://developer.apple.com/downloads/index.action`
3. From the list of downloads, double-click the listing of the version of Xcode you want to download, and then click the listed .dmg file to initiate the download.

   > ***Important:*** You must install, at a minimum, Xcode version 7.

4. Once the download package has downloaded to your Mac, open the *Downloads* folder, double-click the file you just downloaded, and then follow the prompts to install Xcode.
5. When Xcode has finished installing, launch it. To do so, open the *Applications* folder, and then click **Xcode**.

   > ***Important:*** It is imperative that you launch Xcode so that it runs its initialization and configuration routines. Until it does so, you cannot use iOS simulators in Kony Visualizer.

6. Once it has finished its automated configuration, quit Xcode.

Configure iOS Simulators in Kony Visualizer

Now that you have downloaded, installed, and configured Xcode, you can configure an iOS Simulator in Kony Visualizer.

To configure an iOS Simulator in Kony Visualizer, do the following:

1. On the **Product** menu, click **Emulators & Devices Configuration**.
2. Click the gray arrow for **Emulators** to expose the available channels (Desktop, Mobile, and Tablet), and then click the gray arrow of the channel you want to configure an emulator for. Within that channel, click **iOS**.

3. The details of your Xcode installation should automatically populate the iOS Build Options fields. If they don't, click **Fetch Xcode Details**.
4. To configure a new emulator, click the green plus button, and then in the **Name** text box, type a name for the emulator. If you want to open the project in Xcode, check the **Open project in Xcode** checkbox. Additionally, if you want the simulator to support rendering for Apple Watch, check the **Enable Watch** checkbox.

   > ***Note:*** To use an emulator enabled for Apple Watch, your project must have at least one form created for the Watch channel. If you select to emulate a project using a Watch-enabled emulator but your project has no forms for the Watch channel, an error message displays.

5. When you are finished configuring the emulator, click **Save**.

   > ***Note:*** If you navigate away from the emulator configuration pane or click **OK** or **Cancel** without first saving the changes you made to the emulator, any changes you made to the emulator configuration are lost.

6. To configure additional emulators, repeat steps 2 through 5.
7. When you are finished, click **OK**.

Test your Set Up with the HelloWorld Sample App

Having configured an iOS Simulator in Kony Visualizer, you can ensure everything is working properly using the HelloWorld sample app.

> ***Note:*** If you want to take advantage of Kony Visualizer's right-click functionality on your Mac, make sure your right mouse button is set up properly. To do so, open **Mouse** in **System Preferences**, and then assign **Secondary Button** to the right portion of the mouse.

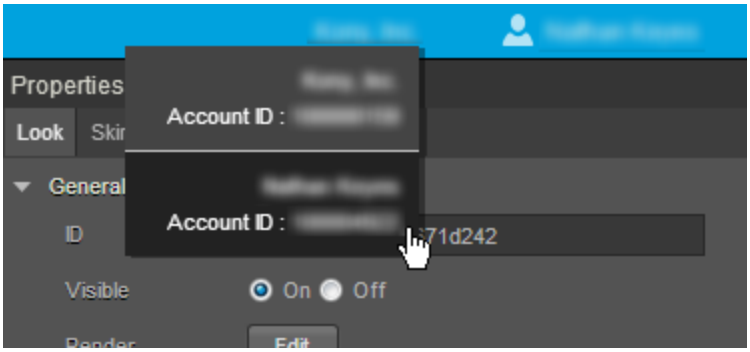
To test the iOS emulators with the HelloWorld sample app, do the following:

1. Download the HelloWorld sample app and import it into Kony Visualizer. To do so, navigate to the HelloWorld sample app page on GitHub, and then click **Download ZIP**. Next, in Kony Visualizer, on the **File** menu, click **Import**, and then in the Import Kony Application dialog box, click the **Browse** button for **Select project root**, navigate to the .zip file that you downloaded, open the **HelloWorld-master** folder, click the **HelloWorld** folder, click **Open**, and then click **Finish**.

2. If you have not done so already, in Kony Visualizer, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**. Kony Visualizer uses the The Kony Fabric URL configured in **Window > Preferences > Kony Visualizer > Kony Fabric** to sign in to Kony Fabric.

   > *Important:* If you are not able to get beyond the login page for the Kony Fabric Console, it may be that you set up Kony Fabric using a self-signed certificate that made it possible to install Kony Fabric, but which Windows and Google Chrome does not trust to allow you to log in. To resolve this, locate the certificate (you may need to contact your system administrator to do so), and then import it into the Windows Certificate Store, into the Trusted Root Certification Authorities folder. For more information on how to import a certificate into the Windows Store, see Import or export certificates and private keys on the Microsoft web site.

3. With Kony Kony Fabric, you can create multiple cloud accounts and connect to any one of them. Now that you're logged in to your Kony account, select the cloud account that you want to use. To do so, in the status bar located over the Properties Editor, click the current cloud account name to display a drop-down list of available accounts, and then click the one you want.



4. If you have not yet selected a default environment for the particular cloud account that you selected in the previous step, you need to do so. On the **File** menu, click **Settings**. Next, click the **Kony Fabric Details** tab. At the top of this tab, under Kony Fabric Environment, select an environment from the drop-down list. Click **Finish**. If you do not see any environments listed, you need to create one. For more information, see Environments in the *Kony Fabric Console User Guide*.

5. To build the project and have it run in an iOS simulator, on the **Product** menu, point to **Run As**, next under **iOS**, point to the simulator you want to use, and then select whether you want it rendered as **Native** or **SPA**. Alternately, if you don't want to rebuild the project, from the **Product** menu, point to **Launch Emulator**, next under **iOS**, point to the simulator you want to use, and then select whether you want it rendered as **Native** or **SPA**.

6. If prompted enter your license information, either by logging in to Kony Fabric, or browsing to the license file located on your Mac. After making your selection, in the License Information dialog box, click **Finish**. In the case of logging in to Kony Fabric, its log in page appears. Enter your credentials.

7. The app proceeds to build. If prompted, choose the iOS simulator application, which is `Xcode.app`.

8. When prompted, if you have a particular debugger port that you use, enter it in the Debugger Host Port text box, and then click **Start**. If you want to forgo the use of a debugger port, click **Cancel**.

   After a while, the simulator renders the HelloWorld app.

Android SDK and Emulator

Applies to *Kony Visualizer Classic*.

Building and testing Android applications in Kony Visualizer requires two primary resources: Android SDK and Gradle. Gradle is an advanced build toolkit that manages dependencies and allows a developer to define custom build logic. This section describes how to install and configure these resources.

Prerequisites that you need to download:

- For V8 SP2
    - Required Android SDK Build Tool version is 26.0.2
    - Required Android SDK Platform API level is 26
- For V8 SP3
    - Required Android SDK Build Tool version is 27.0.3
    - Required Android SDK Platform API level is 26
- For V8 SP4
    - Required Android SDK Build Tool version is 28.0.3
    - Required Android SDK Platform API level is 28
- For V9
    - Required Android SDK Build Tool version is Android X
    - Required Android SDK Platform API level is 29

For any queries you may have about installing Android SDKs and Android Studio, see Android SDK and Platform FAQs.

To build and view applications on the Android platform, do the following:

- Download and install Android Studio
- Download and unzip the Android SDK and setup necessary support packages
- Configure Kony Visualizer to build for the Android platform
- Set the Android SDK Home Environment Variable
- Manually Set the Android Environment Variables
  (This is necessary only if the Android SDK was installed after installing Kony Visualizer)
- Configure Device for USB debugging
- Listing devices and viewing logs
- Configure an Android emulator
- Follow Gradle-related Changes for different Kony Visualizer Versions:
  - V8 SP2
  - V8 SP3
  - V8 SP4
  - V9
- Follow Gradle Recommendations:
  - General Recommendations
  - Build-related Recommendations
- Be Aware of Common Issues with Gradle Migration

Download and install Android Studio

You must install Android Studio for building and testing Android applications using Kony Visualizer. To download and install Android Studio, click Google Android Studio.

Download and unzip the Android SDK and support packages

Required SDKs and support packages can be optionally downloaded during the Android Studio installation process. If you have not downloaded the required SDKs during the Android Studio installation, you can download and install the Android Command line tools (stand-alone SDK tools) by doing the following:

To download and install the Android Command line tools and necessary support packages, do the following:

1. Using a web browser, navigate to the Command Line Tools download section on the Android studio and SDK tools download site.
2. Depending on the operating system of your computer, click the appropriate SDK tools package. After reviewing the terms and conditions of the Android SDK license agreement, if you agree to them, check the option indicating that you have read and agree to them, and then click the download button.
3. Once the zip file downloads, navigate to the downloaded zip file location and unzip it. Place the contents in a folder in your system.

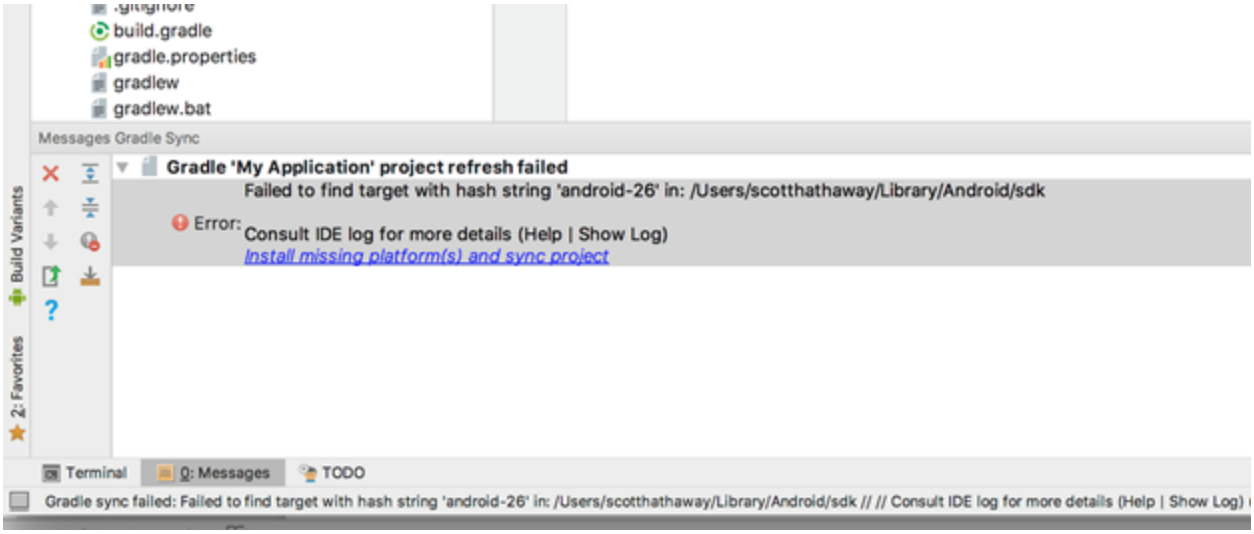
   > ***Important:*** For Windows users, install the Android SDK to a folder with a path that has no spaces, for example:
   > *C:\Android\android-sdk*
   > The default installation path contains at least one space, which may result in the emulator not being accessible in Eclipse and, therefore, Kony Visualizer.

4. Kony Visualizer Android project will download all required SDKs, support packages, and any project specific build library dependencies using the Gradle Auto Download mechanism during the Android native project build.

   All required SDK, support packages, and dependent libraries are auto downloaded by Gradle build.

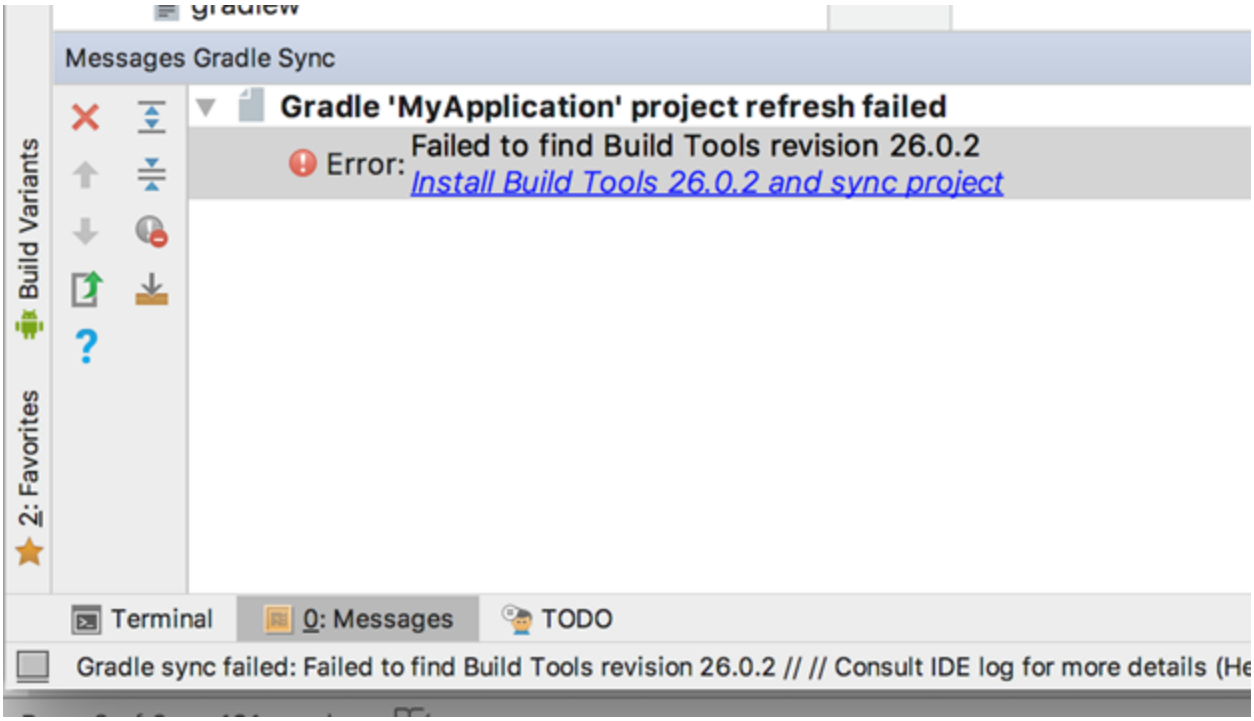
5. You can also download the missing artifacts using Android Studio.

Click **Install missing platform(s) and sync project** link in the message that appears.



Select the defaults through the Component Installer and wait until **Finish** is enabled.

Click the **Install Build Tools 26.2 and sync project** link in the message window.



Select the defaults through the Component Installer and wait until Finish is enabled.

Known Issues:

- When a download dependency is in progress, and the network disconnects in the middle, Android build may hang indefinitely. This is a known technical issue with Gradle. Refer https://github.com/gradle/gradle/issues/868 for more info. You may have to restart visualizer to build again.
- When an SDK component is partially downloaded or corrupted, the build fails. Delete the corrupted SDK component and then proceed with the build.

Configure Kony Visualizer to build for the Android platform

Now that you have installed the Android SDK, you need to configure Kony Visualizer to recognize the Android platform.

To configure Kony Visualizer to build for the Android platform, do the following:

1. In Kony Visualizer, click the **Window** menu, and then click **Preferences**.
2. In the left pane, double-click **Kony Visualizer**, and then click **Build**.
3. If it hasn't done so already, at this point Kony Visualizer auto-detects the Android SDK and asks if you would like to use the path that it has discovered as the Android Home. If you wish to, click **OK**. If Kony Visualizer did not auto-detect the Android SDK, in the **Android Home** text box, enter the path to the Android SDK packages. To ensure you don't introduce errors into the path that you type, you may want to click the accompanying **Browse** button, navigate to the Android SDK's location, and then click **OK**.
4. Click **OK**.
5. Specify which Android SDK to use when building an app. To do so, on the Project menu, click **Settings**.

6. In the Project Properties dialog box, click the **Native** tab. A row of secondary tabs displays. From this row, click **Android**.
7. In the **SDK Versions** section of the tab, from the **Minimum** drop-down list, ensure you set the minimum SDK version.
8. From the **Target** drop-down list, select the SDK version you would prefer to build for.
9. Click **Finish**.

Set the Android SDK Home Environment Variable

The following procedure is for the Windows environment, for the Mac, run the following command:
`export ANDROID_HOME=/<installation location>`

To set the Android SDK home environment variable, do the following:

1. Click **Start**.
2. Right-click **Computer**, and then select **Properties**.
3. Click **Advanced system settings**.
4. On the Advanced tab, click **Environment Variables**.
5. Under User variables, click **New**.
6. For Variable name, type the following value:
   ANDROID_HOME
7. For Variable value, type the path to the parent directory where your Android SDK is installed. For instance:
   C:\Android\android-sdk
8. Click **OK** until you have closed all dialog boxes. Do not click **Cancel**.
9. Restart your computer.

Manually Set the Android Environment Variables

Under some circumstances, you might have to add the Android SDK environment variables manually. This is most commonly necessary if you installed Kony Visualizer before installing the Android SDK. If you already had the Android SDK installed when you install Kony Visualizer, Kony Visualizer detects the presence of the Android SDK and adds the necessary environment variables automatically.

For more information, click the procedure you want.

Add Windows Android Environment Variables Manually

Add Mac Android Environment Variables Manually

Add Windows Android Environment Variables Manually

To manually set the Android environment variables for a Windows computer, do the following:

1. Click **Start**.
2. Right-click **Computer**, and then select **Properties**.
3. Click **Advanced system settings**.
4. On the Advanced tab, click **Environment Variables**.
5. Under System Variables, double-click **Path**.
6. Add to the Path variable the location of the `/bin` folder in your installation of the JDK. For example:

   ```
   C:\Java\jdk1.7.0_79\bin
   ```

7. Add to the Path variable the locations of the `/emulators`, `/tools`, and `/platform-tools` folders in your installation of the Android SDK. For example:

   ```
   C:\Android\android-sdk\emulator; C:\Android\android-sdk\tools;C:\Android\android-sdk\platform-tools
   ```

8. Click **OK** until you have closed all dialog boxes. Do not click **Cancel**.
9. Restart your computer.

Add Mac Android Environment Variables Manually

To manually set the Android environment variables for a Mac computer, do the following:

1. In the home directory, locate `.bash_profile`, and then open it. If you do not have the `.bash_profile` file, create it.
2. Add to the Path variable the locations of the `/emulators, /tools`, and `/platform-tools` folders in your installation of the Android SDK. For example: `C:\Android\android-sdk\emulator; C:\Android\android-sdk\tools;C:\Android\android-sdk\platform-tools`
3. Save the file and close it.

Enable USB debugging on your Android Device

On Android 4.1 and lower, the Developer options screen is available by default. On Android 4.2 and higher, do the following:

1. Open the **Settings** app.
2. Select **System**.
3. Scroll to the bottom and select **About phone**.
4. Scroll to the bottom and tap **Build number** 7 times.
5. Return to the previous screen to find **Developer options** near the bottom.
6. Scroll down and enable **USB debugging**.

List Devices and View Logs

To List the Android devices connected to the Windows 10 PC, do the following:

1. Navigate to **C:\Users\USERNAME\AppData\Local\Android\platform-tools**.
2. Open a command window.
3. Run `adb devices -l` to list the Android devices connected to the Windows 10 PC.

To connect to an Android device on your Windows machine, do the following:

1. Navigate to **C:\Users\USERNAME\AppData\Local\Android\tools**
2. Run `monitor.bat` and click on the connected device.

Configure an Android Emulator

Google has stopped supporting the standalone AVD manager and SDK Manager GUI tools, with latest Android SDK tools. When using, latest Android SDK tools >= 25.3.0, support for launching AVD Manager GUI to create android emulators and SDK manager to download missing components are deprecated from Kony Visualizer V8 release. You must install Android Studio on your machine to get GUI to create and use Android emulators. Click here for more information.

Alternatively, you can use avdmanager command line utility to create the emulators. Refer https://developer.android.com/studio/command-line/avdmanager.html for avdmanager command usage.

When using older Android SDK tools(< 25.3.0), you still would be able to create and launch AVDs using AVD Manager and SDK Manager GUI Tools.

Follow Gradle-related Changes for different Kony Visualizer Versions

This section describes the various Gradle-related changes pertaining to different versions of Kony Visualizer.

V8 SP2 Changes

- The Android plugin version has been updated to 3.0.1.
- Gradle version has been updated to 4.3.

- compileSdkVersion has been changed to 26.
- buildToolsVersion has been changed to 26.0.2.
- Support library dependencies changed to 26.0.0.

V8 SP3 Changes

- The Android plugin version has been updated to 3.1.0.
- Gradle version has been updated to 4.4.
- buildToolsVersion has been changed to 27.0.3.

- The Android plugin version has been updated to 3.2.1.
- Gradle version has been updated to 4.6.
- compileSdkVersion has been changed to 28.
- buildToolsVersion has been changed to 28.0.3.
- Support library dependencies changed to 28.0.0.
- Kony minsdkversion has been changed to API level 17 (4.2).

Important Considerations

- minsdkversion change has been made because of the following reasons:
    - A very small percentage of all Android devices are using API levels earlier than 16. Currently that figure is 1.1% or fewer. Click here for more information.
    - Google Play is discontinuing updates for lower API levels. Click here for more information.

- Support libraries, Recycler View, and AppCompat have been updated to 28.0.0.
    - Any support library dependencies, existing or newly added, (for example, com.android.support:support: or com.android.support:design:) in the build.gradle file must be of version 28.0.0.
      Mismatch in versions may result in version conflicts and gradle build will fail. To resolve this issue, follow the Compilation dependencies and Java symbol conflicts resolution methodology.
- New Gradle 3.2.1 strictly checks some of the Lint issues and leads to Gradle build failure with Lint errors in Release mode.
  The following Lint errors are observed:
    - **ExpiredTargetSdkVersion**: If the application target sdk version is earlier than the Google Play-mandated targetsdk version(Currently 26), then this leads to build failure in Release mode. Click here for more information. You must set the targetsdk version as Google Play-mandated version in release builds.
    - **MissingDefaultResource**: If a resource is only defined in folders with qualifiers such as **-l** and or **-en** (for example, drawable-en-hdpi) and there is no default declaration in the base folder (for example, drawable), layout, or values, then the app will crash. The app crashes when that resource is accessed on a device where the device is in a configuration, where the specified qualifier is missing. Click here for more information.

V9 Changes

- The Android plugin version has been updated to 3.5.2.
- Gradle version has been updated to 5.4.1.
- compileSdkVersion has been changed to 29.
- buildToolsVersion has been changed to 29.0.2.
- Support library dependencies changed to Android X.
- Kony minsdkversion has been changed to API level 19 (4.4).

Follow Gradle Recommendations

For using Gradle, you must go through the following sections:

- General Recommendations
- Build-related Recommendations

> ***Important:*** When you upgrade to Kony Visualizer 8.2 and later, ensure that there are no conflicts in the dependencies of the build.gradle and libs folder. For example, appcompat-v7 added to the build.gradle is X version and if the same file is in the libs folder is of Y version. Due to differences in versions of the file, following build exceptions occur.
> **duplicate entry exception or com.android.builder.dexing.DexArchiveMergerException: Unable to merge dex**
> For more information on how to debug these type of conflicts, click here.

General Recommendations

- Gradle generates a build error if it detects that a JPEG file is named as a PNG file (that is, the JPG has a .png extension). For instructions on how to automatically convert such JPGs, see Convert JPGs named as PNGs to the PNG Format.
- Android has deprecated libraries that are included in earlier versions using the `libproject`mechanism.
- The default NDK support has changed to arm-v7, so any third party FFI modules which use NDK need to include the arm-v7 libraries.
- Libraries to be included must be added in `.aar` format in the `lib` folder.
- Google-play-services are added from Gradle directly. You must install the Android Support Library so that the required libraries are detected.
- For guidelines on converting to the `.aar` format, see Migrating from Eclipse ADT on the Android Developer site.

- The Gradle build system strictly enforces its build requirements, so you must ensure that you have followed all of the Gradle requirements with regard to names, tags in xml, and so on when creating Native Android applications.

Build-related Recommendations

Build failures occur if there are any deviations from the Android-specified requirements. To build an understanding of how Android enforces its requirements, you can create a native application and test various scenarios. In creating such an app, you will want to be aware of the following:

- Any files not supported by Android must not reside in any part of the application folder (build folders which are generated). For example, if a file named `abc.txt` or `cert.crt` is added to the drawable folder, Gradle will fail the build.
- FFI JARs must use correct *pro-guard obfuscation techniques* (if you are planning to mask the jar). If they do not, build errors occur, such as *Unknown verification type [] in stack map frame*. This error indicates that there are masking issues with the JAR that need to be corrected.
- Gradle generates a build error if it detects that a JPEG file is named as a PNG file (that is, the JPG has a .png extension). For instructions on how to automatically convert such JPGS, see Convert JPGs named as PNGs to the PNG Format. Otherwise, you can use some third party application to convert such files. The error that is generated is as follows:

  ```
  libpng warning: iCCP: Not recognizing known sRGB profile that has been edited
  ```

Compilation Dependencies and Gradle Build Java Symbol Conflicts Resolution Methodology

If the build fails with the following exceptions and you do not know the root cause, following the debugging procedure provided here:

- Execution failed for the task :**app:transformClassesWithDexForDebug**. com.android.build.api.transform.TransformException: Error while generating the main dex list.
- **DexArchiveException** or **ProgramTypealreadyPresent** or **DuplicateClassesException**

The debugging procedure for these build issues is as follows:

1. Find the two jars/dependencies that are conflicting by making the following changes in the generated native Android build project. These latest tools will print exactly which classes conflict and their sources/origins. These are taken from Google's suggestions form:
    a. Go to the folder where the native Android project is generated:
        - For Mobile: **<workspace>\temp\<appid folder>\build\luaandroid\dist\<appid folder>**
        - For Tablet: **<workspace>\temp\<appid folder>\build\luatabandroid\dist\<appid folder>**
    b. Change `com.android.tools.build:gradle version` to **3.4.0-beta03** in the `build.gradle` file:
    ```
    buildscript {dependencies{classpath
    'com.android.tools.build:gradle:3.4.0-beta03'}}
    ```
    c. Change the `distributionUrl` in the `gradle-wrapper.properties` file (available in **dist\<appid folder>\gradle\wrapper**) to **https://services.gradle.org/distributions/gradle-5.1.1-all.zip**.
    d. Type the following command in the command prompt, with the same directory as the native Android project: `gradlew assembleDebug`

    The build log then prints the complete details of all the conflicting libraries that fetch duplicate classes.
2. Identify from where the conflicting dependencies are pulled by using this command: `gradlew dependencies`
    This command helps you to view the dependency tree hierarchy in your project. You can then locate the dependency version that was enforced by the relevant compilation dependency.
3. Resolve the conflicts by following these steps:
    a. Adopt the conflict resolution strategy, if dependency version conflicts is the reason. For example, if the project specifies design dependency version (say X) and another version

of design dependency (say version Y) is pulled form recursive dependencies of another dependency (say appcompat-v7), then the build fails.
To resolve this build issue, you can force the build to use only the Y version, regardless of any version included by the dependency tree by adding a snippet as shown in the build.gradle.

```
configurations.all {
    resolutionStrategy {
        force "com.android.support:design:Y"
    }
}
```

b. Remove any duplicate jars or classes found in any of the .aar files and libraries.
For example, you can use the following script to delete the volley and gson-2.2.4 jar files from libs if they conflict with your dependencies in the build.gradle file.

```
task deleteJars
{
 delete "libs/volley.jar"
 delete "libs/gson-2.2.4.jar"
}}
```

> ***Note:*** To add additional entries in the build.gradle file, select the **build.gradle entries to suffix** option under the **Gradle Entries** tab in **Application Properties**. For more information on Gradle properties, click here.

Proxy-Related Build Recommendations

If your computer has proxy settings, you can alleviate build errors at the system level, and at the project level.

System-Level Proxy Recommendations

To address proxy-related build issues at the system level, you can either disable your computer's proxy settings, or create a gradle.properties file to your Gradle installation.

To create a gradle.properties file, do the following:

1. Navigate to the following folder:
   For Windows
   `C:\Users\<UserName>\.gradle`

   For the Mac
   `/Users/<UserName>/.gradle`
2. Create a new text file using a text editor such as Notepad or TextEdit, and save it with the following file name:
   `gradle.properties`
3. Edit the `gradle.properties` file to include the following settings (replace the values given in the example with your own settings).

```
org.gradle.daemon=true
org.gradle.parallel=true
systemProp.http.proxyHost=www.somehost.org
systemProp.http.proxyPort=8080
systemProp.http.proxyUser=userid
systemProp.http.proxyPassword=password
```

4. Replace *http* with *https*, based on the proxy server settings.

   For more information, see The Build Environment on the Gradle web site.

Project-Level Automated Recommendations

To automate project-level requirements, do the following:

1. Create Android pre-compile and post-compile automated tasks. For more information, see Android Pre-compile and Post-compile Ant Tasks Support.
2. Add the `gradle.properties` file mentioned earlier to the project folder. For example:
   For Windows
   `C:\<workspace>\<ProjectName>`

   For the Mac
   `/Users/<UserName>/<Workspace>/<ProjectName>`
3. Copy the file `androidprecompiletask.xml` from the path
   `<workspace>\temp\DeepLinkApp\build\luaandroid\extres`
   to
   `<workspace>\DeepLinkApp`
4. Open `androidprecompiletask.xml` in a text editor such as Notepad or TextEdit, and edit it as your situation requires. For example:

```
<target name = "PreCompileSetup">
<echo message = "basedir = ${basedir}, appdir = ${app.dir},
isMobileBuild = ${isMobileBuild}">
<echo message = "Build mode = ${build.option}, Packagepath =
${packagepath}, x86 Support = ${supportx86}">
<delete file = "${app.dir}/gradle.properties">
<copy file = "${basedir}/gradle.properties">
tofile = "${app.dir}/gradle.properties">
</target>
```

Project-Level Manual Recommendations

To manually make the necessary changes at the project level, do the following:

> ***Note:*** This needs to be done following each time you build the project

1. In Windows Explorer or Finder, navigate to the following path:

For Windows

```
<Workspace>\temp\<ProjectName>\build\luaandroid\dist\<ProjectName>
```

For the Mac

```
/Users/<UserName>/<Workspace>/temp/<ProjectName>/build/luaandroid/dist/<ProjectName>
```

2. If you haven't done so already, edit the `gradle.properties` file to include the proxy settings mentioned earlier.
3. Open a command or terminal window.
    - To open a terminal on a Mac, from the Dock, select **Finder**, double-click **Applications**, next double-click **Utilities**, and then double-click **Terminal**.
    - To open a command window in Windows, Click **Start**, and then in the **Search programs and files** text box, type *cmd.exe*. When it appears in the search results, right-click it, and then click **Run as administrator**.
4. Navigate to the path in step 1.
5. Build the application using the following command line:
   `gradle assembleDebug`

Packaging Error Build Recommendations

Android generates a packaging error if one or more JAR files have duplicate files or classes. To solve this issue, do one of the following:

- Manually delete duplicate files from the JAR files and perform a command-line build. To build the APK file from the command line, do the following:
    1. Open a command or terminal window.
        - To open a terminal on a Mac, from the Dock at the top of the screen, select **Finder**, double-click **Applications**, next double-click **Utilities**, and then double-click

Terminal.

- To open a command window in Windows, Click **Start**, and then in the **Search programs and files** text box, type `cmd.exe`. When it appears in the search results, right-click it, and then click **Run as administrator**.

2. Navigate to the following path:
   For Windows
   `<Workspace>\temp\<ProjectName>\build\luaandroid\dist\<ProjectName>`

   For the Mac
   `/Users/<UserName>/<Workspace>/temp/<ProjectName>/build/luaandroid/dist/<ProjectName>`
3. Run the following command:
   `gradle assembleDebug`
4. Install the generated APK file.

- Use the post compile task and exclude the duplicate files from getting packaged by adding the `packagingOptions` tag in the `build.gradle` file. To do so, do the following:
  1. Navigate to the `build.gradle` file, which is located at the following path:

     For Windows
     `<Workspace>\temp\<ProjectName>\build\luaandroid\dist\<ProjectName>`

     For the Mac
     `/Users/<UserName>/<Workspace>/temp/<ProjectName>/build/luaandroid/dist/<ProjectName>`
  2. Open the `build.gradle` file in a text editor such as Notepad or TextEdit.
  3. In the `build.gradle` file, inside the Android tag, add the following lines.

```
packagingOptions {
    exclude 'META-INF/LICENSE.txt' //File name to delete }
```

Resolve AAPT errors on Windows 8 machines

1. Install the Windows 10 Universal C Runtime from : https://www.microsoft.com/en-us/download/details.aspx?id=48234&WT.mc_id=rss_alldownloads_devresources
2. Extract the downloaded zip file.
3. Double-click the **.msu** file that corresponds to your system configuration to run the Microsoft Update Standalone Package.

If you view the alert message `Windows Update cannot check for updates, the service is not running`, follow these steps to fix the error:

1. Click **Start** from the taskbar.
2. In the **Search** field, type **Administrative Tools**.
3. In the **Administrative Tools** window, double-click **Services**.
4. In the **Services** window, right-click **Windows Update**, and then click **Properties.**
   The **Windows Update Properties (Local Computer)** window appears.
5. In the **General** tab, for the **Startup type**, select **Automatic (Delayed Start)**.
6. In the **Services** window, ensure that the **Windows Update Status** has changed to **Started**, and close the window.
   If the Windows Update Status has not changed to Started, close the window and Restart your system.

Convert JPGs named as PNGs to the PNG Format

Gradle checks for PNG files while building your application. If a JPEG file in the folder is named as a PNG file (that is, the JPG has a .png extension) Gradle generates a build error. To avoid this issue, Kony developed a Python script tool to check if the image files are correctly named. You can use this tool to convert the JPEG files to PNG.

To use the Python JPG conversion tool, do the following:

1. Install the following Python executables:

   https://www.python.org/ftp/python/2.7.10/python-2.7.10.msi

   http://effbot.org/downloads/PIL-1.1.7.win32-py2.7.exe

2. Download the following archive from here and execute the following command:

```
$ python pngConversion.py -p E:\res\drawable(replace with the folder)
```

Common Issues with Gradle Migration

The following issues and errors can arise as a result of a Gradle migration.

Gradle Could not resolve com.android.tools.build:gradle:1.3.1

This condition can occur if there is no internet connection or when the system is being used when the proxy is not set. To resolve, set the proxy, or connect the computer to an internet connection.

Error: Gradle Wrapper not found in Android SDK

If you have been using Cordova applications in Visualizer and upgrade your Android SDK tools version to later than 25.2.5, when you build a project with Cordova support the Gradle Wrapper not found error occurs. Android removed Gradle wrapper package from the Android SDK tools version later than 25.2.5. As a result, Gradle wrapper does not exist in the Android SDK PATH and that results in the error.

To resolve, add Gradle to your path environment variable. For more information, refer to Gradle in Android Platform Guide.

Error: Unsupported type net, etc.

This condition can occur if the application is packing some internal files which are used in the android fwk, this would normally happen if the application is be packing *.xml which can be found in the following folder:

`..\resources\res\values`

To resolve, modify the XML files to change the custom tags net to string.

Duplicate Entry error

Occurs when the same .jar files are being added multiple times, or when different .jar files have the same classes. To resolve, remove the duplicate classes from .jar files.

Gradle OOM issue

To resolve, add javaMaxHeapSize (build.gradle), org.gradle.jvmargs(gradle.properties) values to the script. These are general options, which you can configure according to the needs of your app.

Peer not authenticated (proxy with https setting)

To resolve, download the local proxy server certificate and add it to the Java Key store.

Error: Could not create the Java Virtual Machine

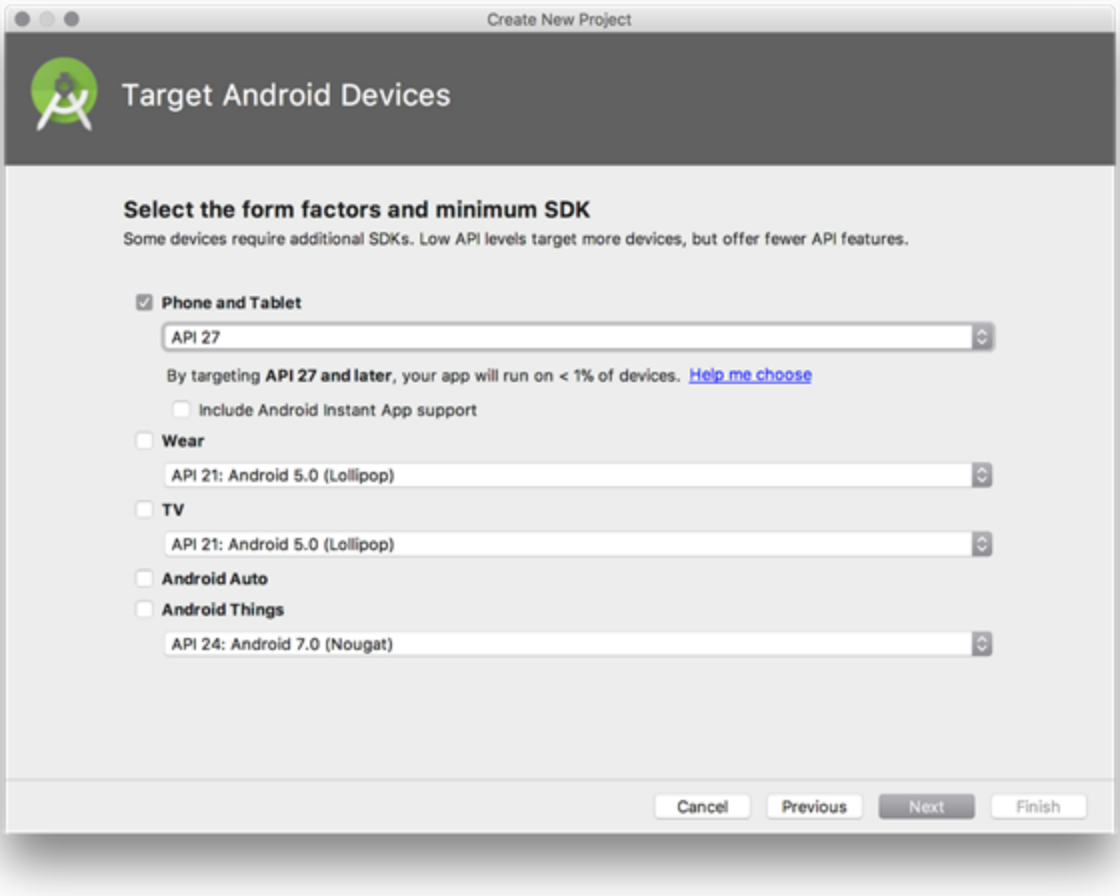
Usually occurs when Gradle is not able to allocate the required memory to build the project, generally noticed on 32-bit computers. If the error occurs on a 64-bit computer, add javaMaxHeapSize (build.gradle), org.gradle.jvmargs (gradle.properties) values to the script. These are general options, which you can configure according to the needs of your app.

Setting a New Android Environment in Windows

This section explains how to set up new Android Environment on Windows.

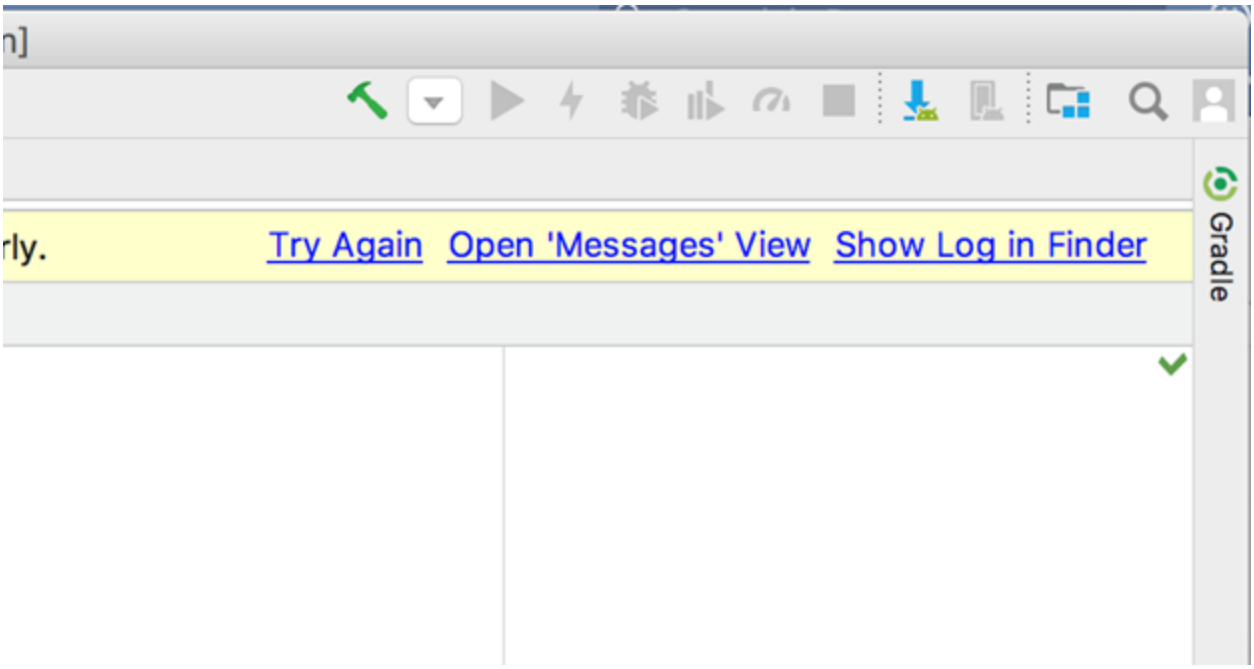
1. Download the latest version of Android Studio for the desired operating system. https://developer.android.com/studio/index.html
2. Install Android Studio.
3. Start Android Studio.
4. Following the instructions by the Android Studio install wizard to download and install SDK Components. Select default options as required.
5. Start a new Android Studio project.

6. In the Target Android Devices page, make sure to select the form factor and minimum SDK as **API 27**.

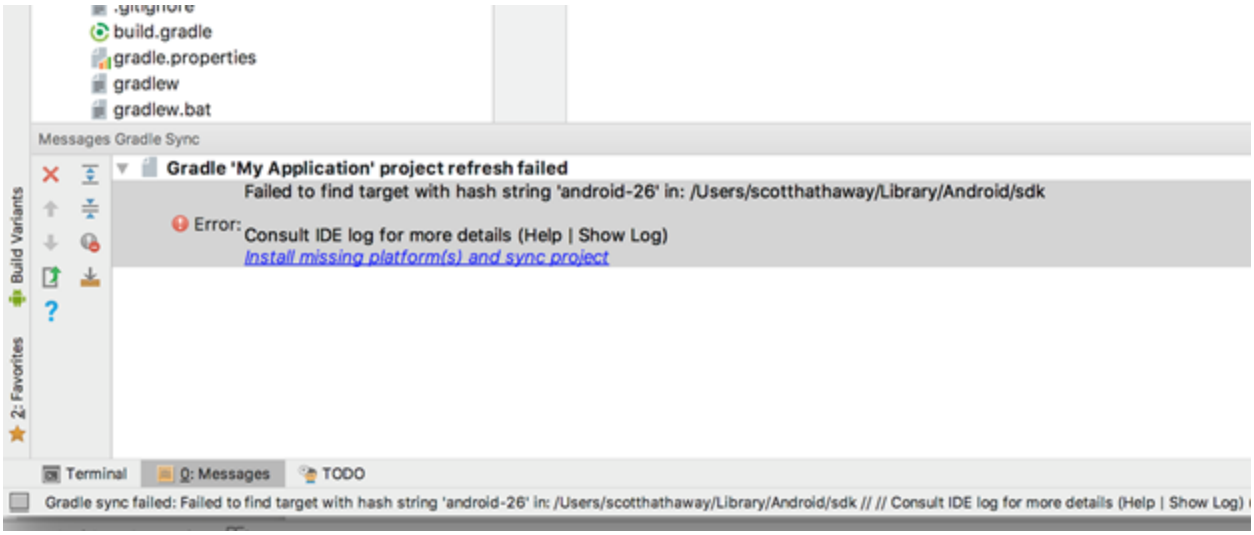


7. Continue selecting defaults until the **Finish**button is available.

8. At this point, the AVD button in the top right will be greyed out.

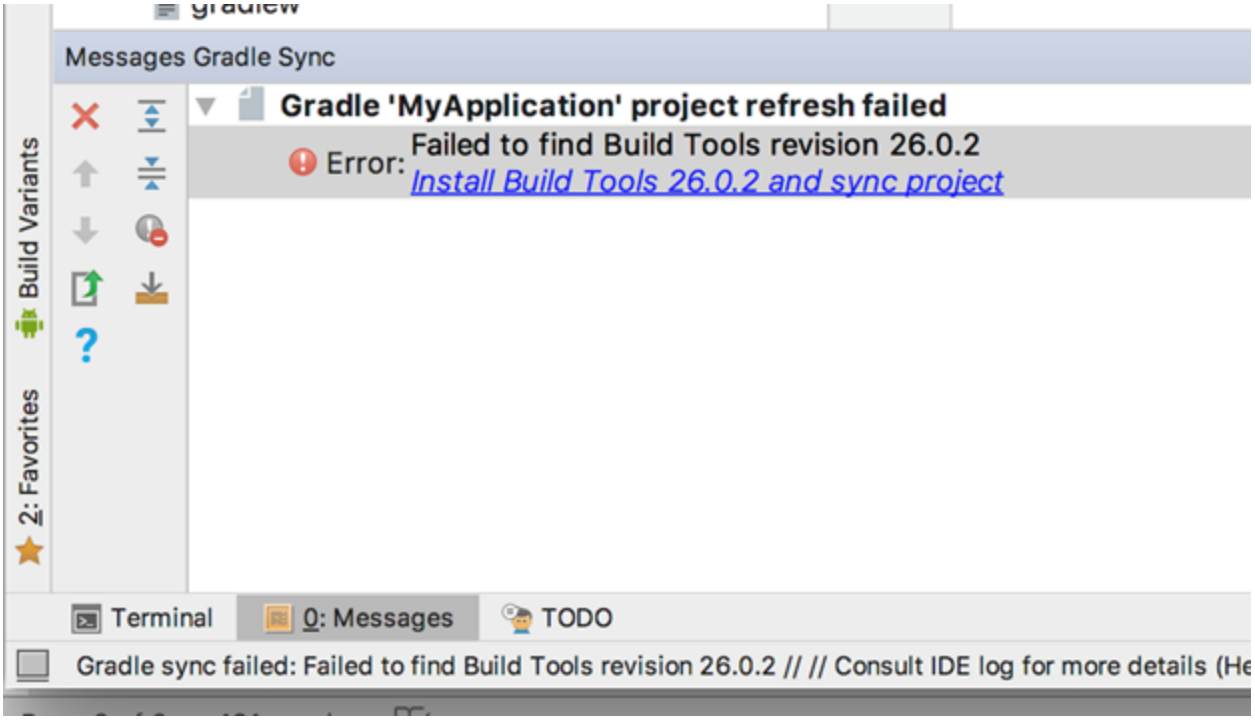


9. Click **Install missing platform(s) and sync project** link in the message that appears.



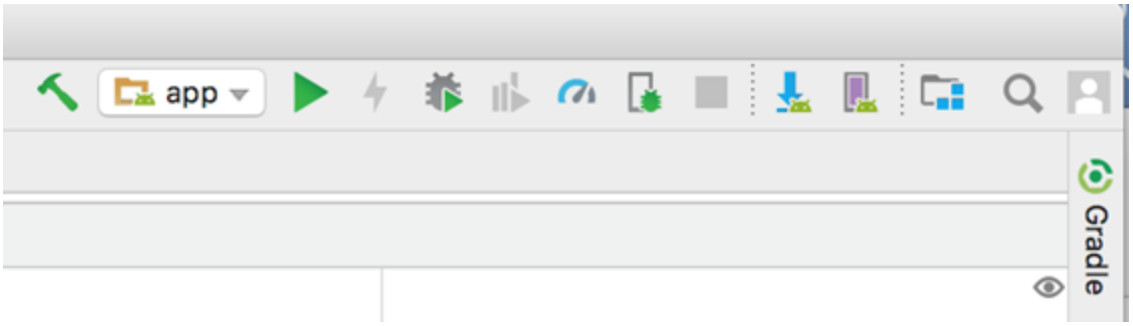
10. Select the defaults through the Component Installer and wait until **Finish** is enabled.

11. Click the **Install Build Tools 26.2 and sync project** link in the message window.



12. Select the defaults through the Component Installer and wait until Finish is enabled.

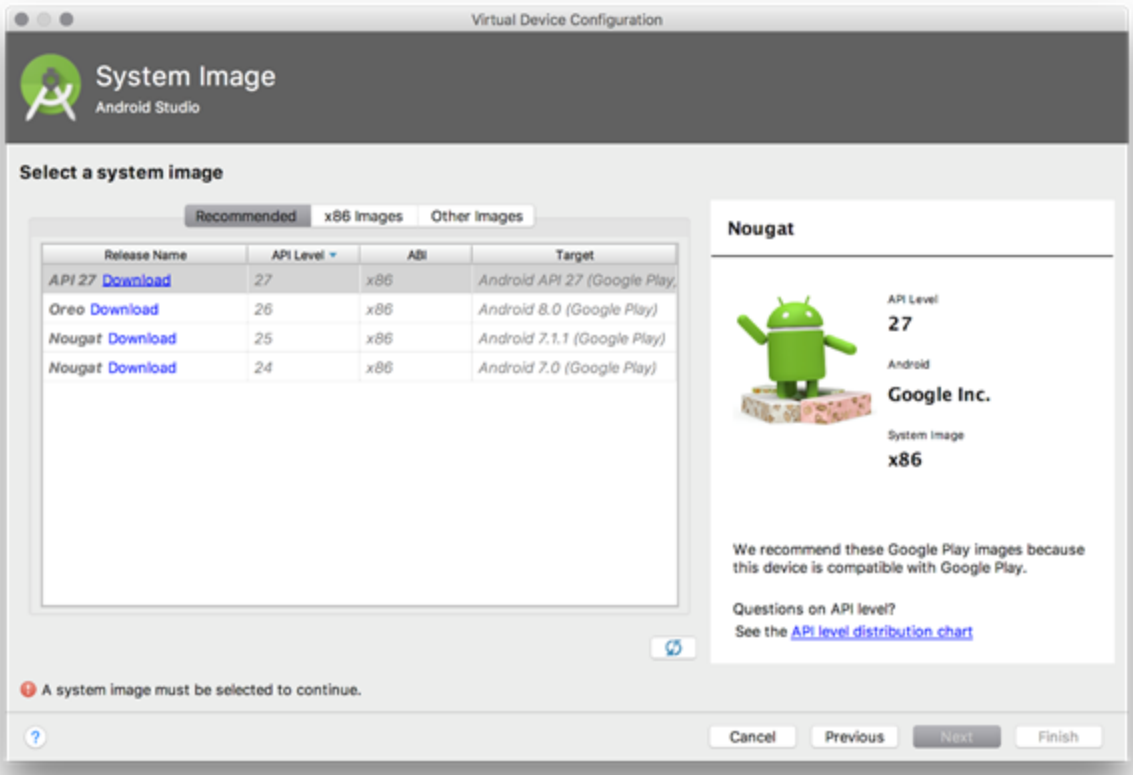
13. Wait for the Gradle build to complete, and the AVD button should now enable.



14. Click the AVD button and select **Create Virtual Device**.

15. Select **Nexus 5X** and then click **Next**.

16. Download the **Oreo API 26 x86 image** from the **Recommended** tab.



17. Once the download completes, finish the setup with defaults.
18. Launch the emulator from the **AVD window**.
19. Open Kony Visualizer V8.
20. Navigate to **Window** > **Preferences**.
21. Select **KonyVisualizer** > **Build**.
22. Set the Android Home field to the same location as the SDK install location of Android Studio and then click **Apply**.

    If you have trouble finding this, in Android studio open the SDK Manager and look at the Android SDK Location.

23. Open the Preferences again and navigating to **Kony Visualizer** > **Emulators** > **Mobile** > **Android**
24. Click on the **+** sign
25. Set the name to **Nexus5X**.
26. Set the emulator location by browsing to the SDK location tools directory and selecting the emulator launcher.
27. Select **Nexus_5X_API_276** for the AVD Name. (If it is not there, you might need to restart Visualizer)
28. Click **Save** and then **OK**.

> ***Note:*** If you are asked to install the Intel HAXM software, install the software.

> ***Important:*** There is a bug in Visualizer for Windows that stops the emulator from launch when you click **RunAs**. You must launch the emulator first from AVD to work around this issue.

Android Pie Behavioral Changes

In this document, we will explain the various Android Pie (9.0 and API Level 28) behavioral changes that are observed when apps target Android Pie and that run on Android 9.0 devices.

Network Security

Android Pie recommends using secure **https** connections henceforth. Network connections with scheme as **http** do not work on apps targeting Android Pie.

HTTP connections will fail with the following exception message: **Cleartext HTTP traffic to * not permitted**

Consequently, you must migrate all **http** URLs to **https** URLs used in Kony Android application.

If your app needs to enable cleartext for **specific domains**, you must explicitly set **cleartextTrafficPermitted** to **true** for those domains in your app's Network Security Configuration.

For example, if you want to permit clear text traffic to google.com and its sub-domains, use the following network_security_config.xml.

```
<?xml version = "1.0" encoding = "utf-8"?>
<network-security-config>
 <domain-config cleartextTrafficPermitted = "true">
 <domain includeSubdomains = "true">
 google.com </domain>
</domain-config> </network-security-config>
```

For more information, click here.

Restrictions on the Use of non-SDK Interfaces

For apps running on Android 9 (API level 28), Android introduces new restrictions on the use of non-SDK interfaces, whether directly, via reflection, or via JNI. For more information on these restrictions, click here.

For more information on the types of SDK interfaces and their descriptions, click here.

- **whitelist**: The SDK provided methods that can be used in the application.
- **light-greylist**: When the application uses non-SDK methods or fields that are still accessible, this log is printed in Debug mode: **Accessing hidden ...(light grey list)**.
  No log is printed in Release mode.
- **blacklist**: The usage of these APIs or fields are restricted regardless of the target SDK. The platform will behave as if the interface is absent.
  - Android Framework will throw **NoSuchMethodError** or **NoSuchFieldException** whenever the app tries to use blacklisted methods.

When the app tries to list or enumerate blacklisted fields or methods of a particular class, they will not be listed.

- **dark-greylist**: For apps whose target SDK is earlier than API level 28, the use of a dark greylist interface is permitted.
  For apps whose target SDK is API level 28 or later, the same behavior is observed as in the case of blacklist.

> ***Note:*** You must build a app that can be debugged, and then run your tests for the app. A logcat warning is printed of the form **Accessing hidden field|method...** for every use of non-SDK interfaces, including that are denied.

> ***Important:*** You must not use the hidden interfaces that are listed in darklist and blacklist via NFI or FFI. If any of these interfaces are used, the application may crash.

AndroidX Behavioral Changes

AndroidX is the new and enhanced version of the Android Support Library. Android Pie is the final release for `android.support`; all new feature releases will be available in `androidx-packaged`.

In this document, we will explain about Kony's AndroidX-supported plugins and the Jetifier flags that are automatically added in the `gradle.properties` file. Furthermore, we will discuss about the various issues that arise while you try to migrate your project to AndroidX and how to resolve these issues.

This document contains the following sections:

- Automatic addition of Jetifier flags in the `gradle.properties` file
- Issues while migrating to AndroidX and their resolutions
- Convert `android.support` references by using Jetifier tool

Automatic Addition of Jetifier Flags in `gradle.properties` File

From V8 SP4 Fixpack 47, Kony has released AndroidX-supported plugins. As part of these plugins, the following Jetifier flags are specified in the `gradle.properties` file:

- `android.useAndroidX=true`: The Android plugin uses the appropriate AndroidX library, instead of using a Support Library.
- `android.enableJetifier=true`: The Android plugin automatically migrates existing third-party libraries (such as, JAR and AAR) to use AndroidX by rewriting their binaries.

> ***Important:*** After both these flags have been set, if the native Android project contains any references to the Android Support Library in its sources files and manifest entries, the command-line build will fail due to an Android Tools issue.

> ***Note:*** With Android Studio 3.2 and later, you can migrate an existing project to AndroidX by selecting **Refactor** > **Migrate to AndroidX** from the menu bar. This action converts all `android.support` references to `androidx`. For more information about how to migrate an existing project to AndroidX, click here.

Issues while Migrating to AndroidX and their Resolutions

While consuming Kony's AndroidX-supported plugins and migrating your project to AndroidX, you must manually replace any `android.support` references in the following scenarios:

- **The plugins of a Cordova project or a React Native project that use `android.support` libraries.**
  Resolution: In this scenario, you can use the following Cordova and ReactNative project folders and click the **Migrate to AndroidX** option in Android Studio to convert all `android.support` references to `androidx`:
  - **Cordova**: Kony has provided you the control of the cordova `android` build folder (which gets generated during build time) to make the necessary changes. You can manually

customize the Cordova-generated Android project, bundle this customized project, and then check in the modified cordova `android` build folder. For more information about the details of this resolution, click here.

- **ReactNative**: Because you already have control of the react-native `android` project folder, you can directly integrate the AndroidX-migrated project. This folder gets integrated into the Kony project as a library project during build time.

- **For `android.support` references in NFIs and Android manifest tag entries added through Kony Visualizer.**
  Resolution: You must manually change all `android.support` references, and then build the application again. For more information about how to convert `android.support` references by using the Jetifier tool, click here. For more information about the mapping of namespaces from `android.support` to `androidx`, click here.

Convert `android.support` References by using Jetifier Tool

1. Search the JavaScript files, Android Manifest tag entries, and XML files for the occurrence of the `android.support` string.
2. The Jetifier tool only accepts a .JAVA or .XML file as input. In accordance with this, copy all the `android.support` strings that you find to one file. For example, `supportstrings.java`.
3. Click here to download the Jetifier zip file from the **Install jetifier** section, and then extract it.

   > ***Note:*** Your must install Java version 1.8 in your system to run the Jetifier standalone tool.

4. Go to the extracted folder > `bin` folder, and then run the following command to get `androidx` equivalent references of `android.support` in the output file.
   -i: The input file with `android.support` references.
   -o: The output file where the equivalent `androidx` references are to be saved.

```
jetifier-standalone -i <source-library> -o <output-library>
```

5. You can now use the contents of the `androidxstrings.java` file to replace all the `android.support` references.

Windows Phone and Tablet

Applies to *Kony Visualizer Classic*.

Using Kony Visualizer, you can develop applications on a number of Windows versions for different devices and channels. For instructions on how to do so, click one of the following options.

- Windows UWP Tablet

Windows UWP Tablet

With Kony Visualizer, you can create an app that supports Windows 10 universal Windows program (UWP) tablets. Getting your system set up to build and test Windows 10 applications in Kony Visualizer involves the following tasks.

- The Windows 10 SDK works best on the Windows 10, Version 1511 operating system. However, it is also supported on the following versions of Windows, although not all tools are supported on these versions: Windows 8.1, Windows 8, Windows 7, Windows Server 2012, Windows Server 2008 R2
- If you want to use the Windows 10 tablet simulator, your computer must be running, at a minimum, Windows 8.0.

Getting your system set up to build and test Windows 10 applications in Kony Visualizer involves the following tasks.

1. Confirm your system meets Windows 10 development requirements
2. Create and/or sign in to your Microsoft account
3. Install and Configure the Windows SDK 10

Confirm Your System Meets Windows 10 Development Requirements

To develop for the Windows 10 platform and run its emulator, your computer needs to meet certain hardware and software requirements. The procedures in this section guide you through confirming that your system meets them. At a minimum, it needs to be capable of running Microsoft Windows 8.

Confirm that your computer's CPU supports SLAT.

Confirm other system requirements for Windows 10.

Confirm that your computer's CPU supports SLAT

To run the Windows 10 tablet simulator, your computer needs a central processing unit (CPU) that supports Second Level Address Translation (SLAT) .

> ***Note:*** SLAT support is necessary to use the Windows 8 emulator. If your CPU does not support SLAT, you can still install and use the Windows 8 SDK in Kony Visualizer on a computer running Windows 8 Pro or greater, but you won't be able to use the Windows 10 tablet simulator.

To check if your CPU supports SLAT, do the following:

1. Download *Coreinfo.zip* from the Windows Sysinternals web page.
2. Extract the zip file to a folder, such as *C:\Coreinfo*.
3. Click **Start**, and then in the **Search programs and files** text box, type *cmd.exe*.
4. When it appears in the search results, right-click it, and then click **Run as administrator**.
5. In the command window that launches, type `coreinfo.exe -v`, and then press **Enter**.
6. **Edit the system environments**. Doing so opens the **System Properties** dialog box.
7. Navigate to the path where you extracted the zip file.
8. In the command window that launches, type `coreinfo.exe -v`, and then press **Enter**. After a few seconds, a license agreement for Coreinfo appears. If you agree to the terms, click **Agree**. Coreinfo will then display the results of its analysis.

   An asterisk (*) displays if the CPU supports the Windows 8 platform. If no asterisk displays, then the CPU does not support the Windows 8 platform.

   For example, for an Intel CPU, if the processor supports SLAT, an asterisk (*) displays in the EPT row, indicating support for extended page tables, as seen here.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd c:\

c:\>coreinfo.exe -v

Coreinfo v3.0 - Dump information on system CPU and memory topology
Copyright (C) 2008-2012 Mark Russinovich
Sysinternals - www.sysinternals.com

Intel(R) Core(TM) i7 CPU         930  @ 2.80GHz
Intel64 Family 6 Model 26 Stepping 5, GenuineIntel
HYPERVISOR      -       Hypervisor is present
VMX             *       Supports Intel hardware-assisted virtualization
EPT             *       Supports Intel extended page tables
```

If you have an AMD processor, then an asterisk (*) is displayed for NP (Nested Page tables), as seen here.

```
C:\>coreinfo -v

Coreinfo v3.0 - Dump information on system CPU and memory topology
Copyright (C) 2008-2012 Mark Russinovich
Sysinternals - www.sysinternals.com

AMD E-350 Processor
AMD64 Family 20 Model 1 Stepping 0, AuthenticAMD
HYPERVISOR      -       Hypervisor is present
SVM             *       Supports AMD hardware-assisted virtualization
NP              *       Supports AMD nested page tables
```

Confirm Other System Requirements for Windows 10

To develop for the Windows 10 platform and run its tablet simulator, your computer needs to meet the following additional system requirements for hardware and software.

- 64-bit (x64) CPU
- 2.5 GB of internal storage
- 4 GB RAM
- Windows 8 Pro edition or greater

Create and/or sign in to your Microsoft account

While it is possible to download and install the Windows SDK 10 without a Microsoft account, getting additional resources, such as emulators, requires one. So if you don't already have an account, creating one at this point in the process will cause the least amount of disruption to you.

To create and/or sign in to your Microsoft account, do the following:

1. Using a web browser, navigate to the Windows Dev Center.
2. Click **Sign in**.
3. Do one of the following, depending on which applies to you:
   - **You already have a Microsoft account.** Enter the email address and password of your account, and then click **Sign in**.
   - **You do not have a Microsoft account.** Do the following.
     1. Scroll down to the prompt that asks *Don't have a Microsoft account?*, and then click **Sign up now**. Doing so opens the *Create an account* page.
     2. Fill in the required fields, and then click **Create account**. A verification page appears informing you that Microsoft has sent you an email message to confirm your identity.
     3. Following the instructions on the verification page, check the email account you associated with your Microsoft account, and using the link in the email message Microsoft sent you, verify your identity.

        > ***Note:*** If you do not see the email message in your inbox, check your spam or junk mail folder.

4. Once you have verified your Microsoft account, if you are not already signed in, do so now by following the initial steps of this procedure.

> ***Note:*** If you are signed in, the email address you associated with your Microsoft account appears in the upper right corner of the web page.

Install and configure the Windows SDK 10

The Windows SDK 10 contains resources to develop universal Windows apps (UWA) for Windows 10, as well as apps and desktop applications for Windows Phone, including emulators.

To install the Windows SDK 10, do the following:

1. Using a web browser, navigate to the Windows SDK 10 page on the Microsoft Download Center.
2. Click **Download the standalone SDK**.
3. Choose the language version you want to install, and then click **Download**. Doing so initiates the downloading of *sdksetup.exe*.
4. Once *sdksetup.exe* has downloaded, launch it and follow the prompts to install the SDK.

Validate the Product License

Applies to *Kony Visualizer Classic*.

If you did not enter your Kony credentials when you installed Kony Visualizer, the first time you launch it, you are prompted to sign in using your Kony credentials so that you can validate that you have a product license. If you do not know your Kony credentials, contact your system administrator.

If you do not have access to a network connection when you launch Kony Visualizer Classic, it bypasses the sign-in credentials, and you will continue to have access to all the product's features. You can bypass the presence of a network connection for up to 15 days, during which you can skip entering your Kony credentials. After 15 days, however, Kony Visualizer Classic once again prompts you for credentials every time you launch it. If another 30 days pass without entering your Kony credentials, you will not be able to use Kony Visualizer Classic until you enter them.

To validate the Kony Visualizer product license, do the following:

1. Launch Kony Visualizer, and if prompted, specify the workspace you want to use. The Enter License Information dialog box displays.
2. In the Enter License Information dialog box, do one of the following:
    - **Sign in.** When you sign in to your Kony account, your license is validated, and a license file called `ide.lic` is downloaded to your computer, validating your product license when you're not signed in.
    - **Upload an existing license file.** If you have an existing license file (`ide.lic`) that you want to use to validate your Kony Visualizer license, select **License**, click its corresponding **Browse** button, navigate to your `ide.lic` file, select it, click **OK**, and then click **Next**. Kony Visualizer uploads your license file to the Kony cloud, where it is validated.
    - **Skip license validation.** For a period between 15 and 30 days from the time you first launch Kony Visualizer, you can skip license validation and have still have full use of the product. However, once this period of time has passed, you must validate your Kony Visualizer product license before you can continue using the product.

Kony Visualizer provides different verity of licenses for each environment. Here are the classifications of each license:

1. Standalone License - Provided as a **ide.lic** file.
2. Cloud Subscription - Provided as an online subscription.

Standalone License

To change your Kony Visualizer License:

1. In Kony Visualizer, navigate to **Help** >**About Kony License.** The **Kony Visualizer License Information** dialog appears.
2. Click **Manage License**.The **Developer Login Details and License** dialog appears.

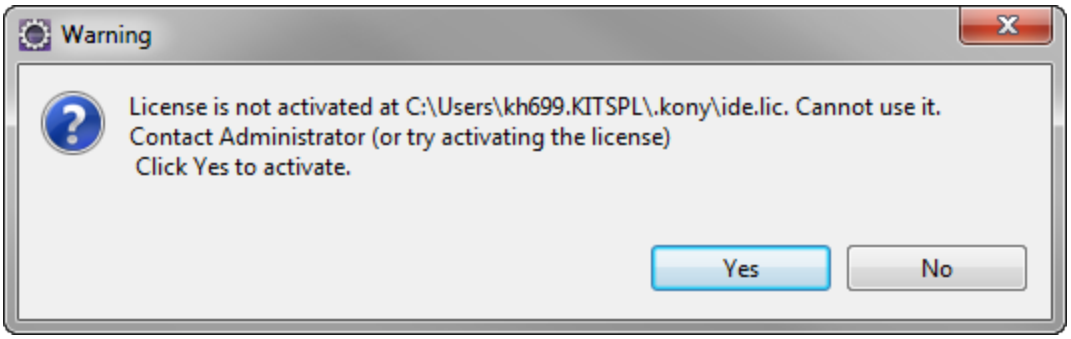
3. You can change your license to IDE or Cloud. The following steps help you to perform the required task:
    a. To change your account to Cloud License, type your log in credentials and click **Finish**. Upon successful login, your username appears on the notification bar.
    b. To change your account to IDE License, select **License** and then select the license file by clicking **Browse**.
        i. Click **Next**. The **License Agreement** dialog appears.
        ii. To accept the agreement, select **I accept the agreement** and then click **Next**. The **Customer Details** dialog appears.
        iii. Enter your personal details and click **Finish**. The Registering License dialog appears to indicate successful login.
        iv. Click **Finish**.

To deactivate your Kony Visualizer License:

1. In Kony Visualizer, navigate to **Help** >**About Kony License.** The **Kony Visualizer License Information** dialog appears.
2. To deactivate the license click **Deactivate License**. A confirmation dialog appears.
3. To de-activate your license, click **Yes**. A message appears stating that the license has been deactivated.
4. Click **OK**.

To activate your Kony Visualizer License:

1. Launch the instance of eclipse for which the license is deactivated. The following message appears.

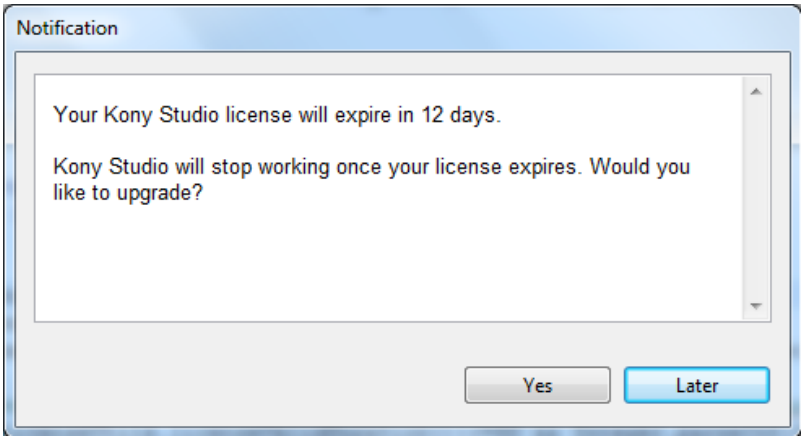


2. Click Yes to activate the license.
3. Follow the On- screen instructions to update your license.

For more information about activating the license, refer to the *Licensing User Guide*.

Before License expiry:

1. Launch Kony Visualizer. Before viewing the Studio, the following Notification dialog box is displayed when the license validity is lesser than or equal to 30 days.

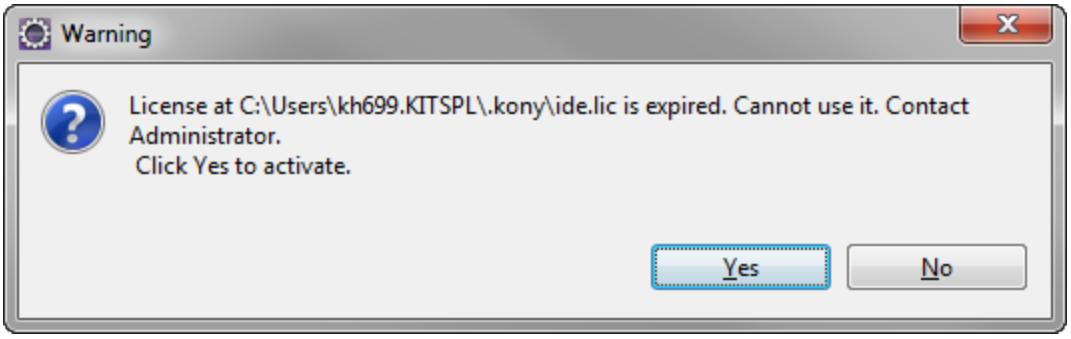


2. Do any one of the following:
   - To upgrade your license some other time, click **Later**.
   - If you have a valid license with you and you wish to upgrade it now, click **Yes**. The **Enter**

License Details dialog appears. Follow the on screen instructions to upgrade your license.

When a license has expired:

1. If your license has expired and you try to launch eclipse, the following dialog appears:



2. Click **Yes** to activate the license if you have a valid license with you.
3. Follow the onscreen instructions to activate your license.

Cloud Subscription

The following table describes the features provided in each subscription. Kony provides two types of Cloud subscriptions:

1. Enterprise Apps
2. Customer Apps

Enterprise Apps	LITE	STANDARD	PROFESSIONAL
Mobile App Development			
Mobile Web Apps (HTML5/HTML4)	✓	✓	✓
Hybrid App	✓	✓	✓
Mixed Mode App		✓	✓
Single Page App (SPA)		✓	✓
Native App		✓	✓

Enterprise Apps	LITE	STANDARD	PROFESSIONAL
Desktop App Development			
Desktop Web			✓
Native Desktop Apps			✓
EMM			
EMM Standard		✓	
EMM professional		✓	
Platform Services Capabilities			
Device Database & Detection	✓	✓	✓
Basic Reporting			✓
Advanced Reporting & Analytics			✓
Engagement Services			✓
Campaign Management			✓
Kony Direct Sync			✓
Kony Persistent Sync			✓
Service Integration & Connectors			
Standard Web Service & Database Connectors	✓	✓	✓
Enterprise Connectors			✓

Customer Apps	LITE	STANDARD	PROFESSIONAL
Mobile App Development			
Mobile Web Apps (HTML5/HTML4)	✓	✓	✓
Hybrid App	✓	✓	✓
Mixed Mode App		✓	✓
Single Page App (SPA)		✓	✓
Native App		✓	✓
Desktop App Development			
Desktop Web			✓
Native Desktop Apps			✓
Platform Services Capabilities			

Customer Apps	LITE	STANDARD	PROFESSIONAL
Device Database & Detection		✓	✓
Basic Reporting			✓
Advanced Reporting & Analytics			✓
Engagement Services			✓
Campaign Management			✓
Service Integration & Connectors			
Standard Web Service & Database Connectors	✓	✓	✓
Enterprise Connectors			✓

Allow Anonymous Usage Data Collection

To help Kony, Inc. understand how you use Kony Visualizer and provide you with relevant improvements to its products, Kony Visualizer includes functionality allowing it to collect anonymous usage data regarding your projects. No personal or corporate data is collected; only anonymous data regarding how the product is used, and participation in the collection of this data is entirely voluntary.

You can also opt in to allow anonymous usage data collection when you install Kony Visualizer.

To allow anonymous user reporting, do the following:

1. On the **Window** menu (the **Edit** menu in Kony Visualizer), click **Preferences**.
2. Do one of the following, depending on the edition of Kony Visualizer you are using:
   - **Kony Visualizer.** In the left pane, click **General**. Next, in the right pane, select **On** for **Anonymous Usage Data Collection**, and then click **Apply**.
   - **Kony Visualizer Classic.** On the left, click **Kony Visualizer**. On the right, check the **Anonymous Usage Data Collection** check box. Next, click **Apply**, and then click **OK**.

Use a Proxy Server

If your network configuration uses a proxy server, you can configure Kony Visualizer to recognize it, whether it is a Basic or NTLM proxy. If yours is an NTLM proxy server, you must follow the procedures for both the Basic proxy and the NTLM proxy.

Basic Proxy Configuration

NTLM Proxy Configuration

Basic Proxy Configuration

Configuring Kony Visualizer to recognize a Basic proxy server involves the following procedures:

Add Proxy Information to the Eclipse Initialization File

Configure the Proxy Server in Kony Visualizer

Configure a Manual Proxy for Your Computer

Add Proxy Information to the Eclipse Initialization File

Kony Visualizer uses Eclipse, an open-source integrated development environment (IDE), for some of its underlying functionality. To use Kony Visualizer with a proxy server, you need to modify the Eclipse initialization file.

To add proxy information to the Eclipse initialization file, do the following:

1. If it is currently open, close Kony Visualizer.
2. Using your computer's file explorer, navigate to the Eclipse initialization file.
    - On the Mac, the file name is `eclipse-orig.ini`. It is located within the Eclipse.app folder, and can be accessed using the **Show Package Contents** command.
      You can access the eclipse.ini file by doing the following:
      a. Navigate to the Kony Visualizer Classic installation path.

b.Right-click **Kony_Visualizer-Enterprise**, and then select **Show Package Contents**.
c. Navigate to **Content** > **MACOS**. Open the **eclipse.ini** in a text editor.

- On a Windows computer, the file name is `eclipse.ini`, and is located at the following location:

  `<VisualizerEnterpriseInstallFolder>\_Kony Visualizer_installation`

3. Using a text editor, open the Eclipse initialization file.
4. Add the following entries to the file:

```
-Dkony.http.proxyHost=xxx.xxx.xxx.xxx
-Dkony.http.proxyPort=xxxx
-Dkony.http.proxyUser=xxxxxx
-Dkony.http.proxyPassword=xxxxxx
-Dkony.noProxy=xxx.x.x.x
```

> ***Important:*** Configure **Dkony.noProxy** only if you want to bypass the proxy for the localhost domain.

For example:

```
-Dkony.http.proxyHost=10.0.0.100/host.example.com
-Dkony.http.proxyPort=8880
-Dkony.http.proxyUser=HannahSmith
-Dkony.http.proxyPassword=3y*ANjScw8BH
-Dkony.noProxy=127.0.0.1,localhost
```

> ***Important:*** For an NTLM proxy server, use the following values for the proxyHost and the proxyPort:
> `-Dkony.http.proxyHost=127.0.0.1`
> `-Dkony.http.proxyPort=3128`

5. Save and close the file.

6. Restart Kony Visualizer.

Configure the Proxy Server in Kony Visualizer

The following procedure configures Kony Visualizer to recognize the proxy server.

To configure the proxy server in Kony Visualizer, do the following:

1. In Kony Visualizer, on the **Window** menu, click **Preferences**.
2. From the left pane of the dialog box, double-click **General**, and then click **Network Connections**. Doing so displays the Network Connections options in the right pane.
3. Set Active Provider to **Manual**.
4. In the Proxy entries table, select the HTTP schema, and then click **Edit**.
5. In the Host text box, enter the proxyHost value that you added to the Eclipse configuration file.

   > ***Important:*** For an NTLM proxy server, the Host value should be `127.0.0.1`.

6. In the Port text box, enter the proxyPort value that you added to the Eclipse configuration file.

   > ***Important:*** For an NTLM proxy server, the Port value should be `3128`.

7. Click **OK**.

Configure a Manual Proxy for Your Computer

In addition to configuring Eclipse and Kony Visualizer to recognize the proxy server, your operating system needs to be configured, as well. Doing so differs between Mac and Windows computers.

Configure a Manual Proxy for a Mac

Configure a Manual Proxy for Windows

Configure a Manual Proxy for a Mac

To configure a manual proxy for a Mac, do the following:

1. Click the Apple menu, next click **System Preferences**, and then click **Network**.
2. From the list, select the network service you are using, such as Wi-Fi or Ethernet.
3. Click **Advanced**, and then click **Proxies**.
4. From the **Select a protocol to configure** list, clear the check box for Auto Proxy Discovery and Automatic Proxy Configuration.
5. From the **Select a protocol to configure** list, check the check box for **Web Proxy (HTTP)**.
6. With **Web Proxy (HTTP)** selected, in the **Web Proxy Server** text box, enter the proxyHost value that you added to the Eclipse configuration file.

   > ***Important:*** For an NTLM proxy server, the Web Proxy Server value should be `127.0.0.1`.

7. In the accompanying port text box, enter the proxyPort value that you added to the Eclipse configuration file.

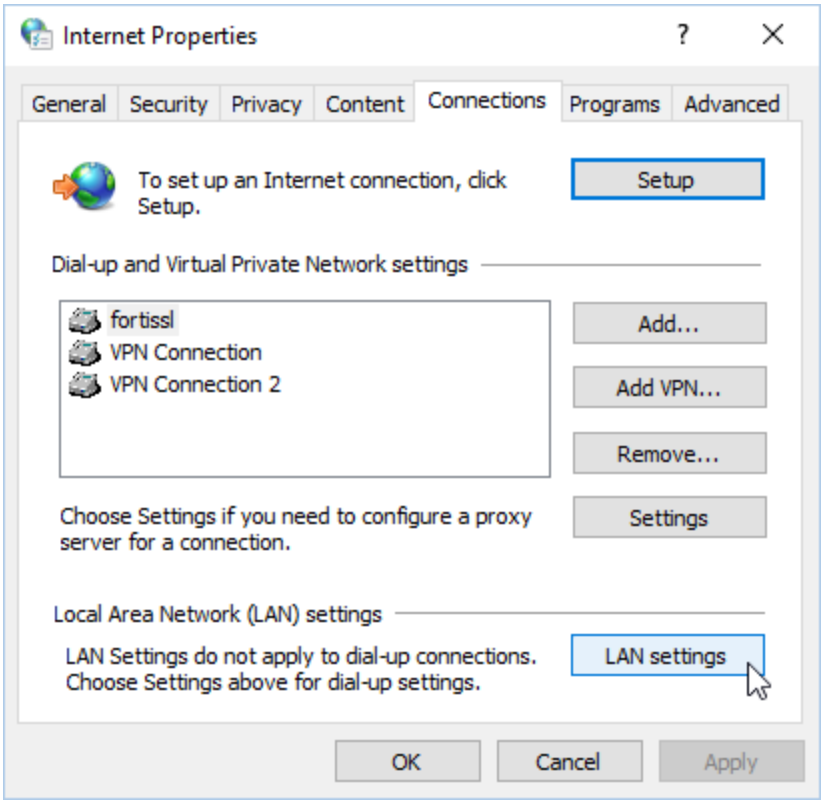
   > ***Important:*** For an NTLM proxy server, the Port value should be `3128`.

8. Repeat steps 5 through 7 for the **Secure Web Proxy (HTTPS)** protocol and the **FTP Proxy** protocol.
9. Check the **Exclude simple host names** check box.
10. Click **OK**.

Configure a Manual Proxy for Windows

To configure a manual proxy for Windows, do the following:

1. Click the **Start** menu, and then click **Settings** (for Windows 10) or **Control Panel** (for Windows 7 and earlier).
2. In the search box, type *Internet options*, and then press **Enter**.
3. From the search results, click **Internet options**. Doing so brings up the Internet Properties dialog box.
4. Click the **Connections** tab, and then click **LAN Settings**.



5. Clear the **Automatically detect settings** check box.
6. Under Proxy server, check the check box for using a proxy server for your LAN.
7. In the **Address** text box, enter the proxyHost value that you added to the Eclipse configuration file.

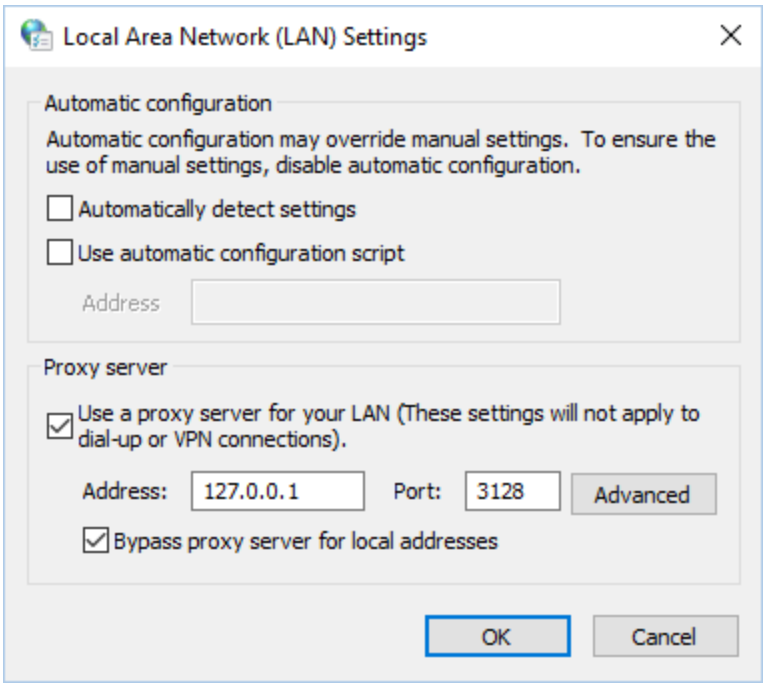
> ***Important:*** For an NTLM proxy server, the Address value should be `127.0.0.1`.

8. In the Port text box, enter the proxyPort value that you added to the Eclipse configuration file.

   > ***Important:*** For an NTLM proxy server, the Port value should be `3128`.

9. Check the check box for **Bypass proxy server for local addresses**.

   Your settings should look something like this (settings shown are for an NTLM proxy server):



10. Click **OK**, and then click **OK** again.

NTLM Proxy Configuration

To configure Kony Visualizer to recognize an NTLM proxy server, you must follow all the procedures related to both the Basic proxy and the NTLM proxy. If you have not yet followed the procedures for a Basic proxy, see Basic Proxy Configuration. Once you have done so, completing the NTLM proxy configuration involves two additional tasks.

Run a Cntlm Process

Allow CONNECT Requests from Non-SSL Connections and Ports

White-List Essential Domains

Run a Cntlm Process

Cntlm is an HTTP proxy that efficiently provides NTLM authentication on the fly. Cntlm integrates TCP/IP port forwarding, SOCKS5 proxy mode, and standalone proxy configuration to allow intranet, Internet, and corporate web server access with NTLM protection.

You run the Cntlm process configured with the proper NTLM host and port information, and any additional needed authentication data.

For more information, refer to the following links:

The Cntlm Authentication Proxy web site

Download Cntlm

Cntlm Documentation

Allow CONNECT Requests from Non-SSL Connections and Ports

By default, proxies such as Squid and ISA do not allow http CONNECT requests from non-SSL connections and ports it deems unsafe. As a result, the proxy server has to be explicitly configured to allow such connections and ports. To do so, contact your system administrator and request that CONNECT requests from non-SSL connections and ports be allowed.

White-List Essential Domains

Kony Visualizer access particular domains that the proxy server must be configured to allow. Contact your system administrator and request that the following domains be white-listed:

https://manage.kony.com
https://api.kony.com
https://visualization.kony.com
https://prototypetransit.kony.com.s3.amazonaws.com
https://accounts.auth.konycloud.com

Modify the Cloud Configuration of Kony Visualizer

To take advantage of Kony Visualizer's full range of functionality, you need to have a Kony user account and sign in to it from within Kony Visualizer. Features related to being signed in to your Kony user account include cloud-based functional preview, publishing, and the Kony Fabric Console.

If you do not have a Kony user account, you can create one at the Kony Cloud Registration site.

The features available via the Kony Fabric Console include:

- Identity - To authenticate users from different types of identity providers.
- Integration - To define and configuring services.
- Orchestration - To leverage the concept of combining multiple integration services
- Offline Synchronization - To configure sync services to your app for backend data, thus enabling offline functionality.
- Engagement Services - To configure notifications.

> ***Important:*** If you are not able to get beyond the login page for the Kony Fabric Console, it may be that you set up Kony Fabric using a self-signed certificate that made it possible to install Kony Fabric, but which Windows and Google Chrome does not trust to allow you to log in. To resolve this, locate the certificate (you may need to contact your system administrator to do so), and then import it into the Windows Certificate Store, into the Trusted Root Certification Authorities folder. Or you can import the certificate into the Java trust store in Kony Visualizer, which is in the following location:
> `C:\KonyVisualizerEnterpriseX.X.X\Kony_Visualizer_Enterprise\jre\lib\security\cacerts`
> For more information on how to import a certificate into the Windows Store, see Import or export certificates and private keys on the Microsoft web site.

Log In to Your Kony Account

To log in to your Kony account, do the following:

1. In the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens.
2. Enter the email and password credentials of your Kony user account, along with any other requirements tied to your custom authentication, and then click **Login**. Once you are signed in, Kony Visualizer has access to your Kony Fabric apps and services.
3. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.
4. With a Kony account, you can be a member of multiple clouds. When you log on to your Kony account, Kony Visualizer defaults to one of those clouds. Any cloud that you are working in has to have at least one Kony Fabric environment associated with it. If the cloud does not have a Kony Fabric environment, the cloud icon in the status bar contains a warning symbol.

If you know that you have access to another cloud that has a Kony Fabric environment, you can select it instead. To do so, on the **File** menu, click **Settings**. Click the **Kony Fabric Details** tab, and then from the **Cloud Account** drop-down list, select a cloud that you know has a Kony Fabric environment. You can then select the environment you want from the **Kony Fabric Environment** drop-down list (if only one environment is available, Kony Visualizer defaults to it). Click **Finish**. The status bar indicates that you are attached to a cloud with a Kony Fabric environment. For more information, see Environments in the *Kony Fabric Console User Guide*.

Connect Kony Visualizer to a Kony Fabric On-Premise Environment

From Kony Visualizer V9 onwards, you can connect to an On-Premise Kony Fabric environment from Kony Visualizer.

> ***Note:*** Ensure that there are no uppercase letters in the Hostname URL of the Kony Fabric environment. Kony Visualizer does not support the use of uppercase letters in the Hostname URLs.

To connect to a Kony Fabric On-Premise Environment, follow these steps:

1. In Kony Visualizer, from the **Edit** menu, click **Preferences**.
2. From the left navigation pane, click **Kony Fabric**.
3. In the **Kony Fabric URL** text box, enter the base URL of the Kony Fabric on-premise environment, and then click **Validate**.
4. Once the Kony Fabric URL is validated, click **Done**.

When you change the Kony Fabric URL, you will be logged out of Kony Visualizer, and the Sign-in page of the Kony Fabric on-premise environment appears. You must sign in to Kony Visualizer using the credentials of your on-premise Kony Fabric environment.

> ***Note:*** While using an on-premise Kony Fabric environment, any cloud-related actions like **Build and Publish Native**, **Package**, and **Publish to EAS** (for Web apps) will not be available. The **Publish Live preview** option is also disabled.

Log In by using Custom Authentication

If your company receives custom authentication support, you must configure Kony Visualizer to include the custom identity of your company.

To configure Kony Visualizer to use Custom Authentication, follow these steps:

1. In Kony Visualizer, from the **Edit** menu, click **Preferences**.
2. From the left navigation pane, click **Kony Fabric**.

3. Select the **Enable External Authentication** check box.
   The **External Login Path** text field is displayed.
4. In the **External Login Path** text field , specify the name of the External Authentication Provider that Kony has assigned.
   For example, if the URL of your custom identity provider is `https://manage.kony.com/<CustomIdentity>`, specify <CustomIdentity> in the **External Login Path** field.
5. Click **Done**.

Once you configure the Custom Identity Provider, the Login page changes to include the Enterprise Login option.

1. On the **Login** page, click **Enterprise Login**. The **Enterprise Login** page appears.
2. Specify the name of your company, and then click **Next**. The **Sign In** page appears.
3. Type the credentials of your Custom OAuth Provider, and then click **Sign In**.

> ***Note:*** You can disable the Enterprise Login option by clearing the URL of the External Authentication Provider from the **External Login Path** field in **Preferences > Kony Fabric** tab.

For information on how to configure **External Authentication for your Kony Fabric cloud account**, refer External User Authentication.

Workspaces: Repositories for Your Projects

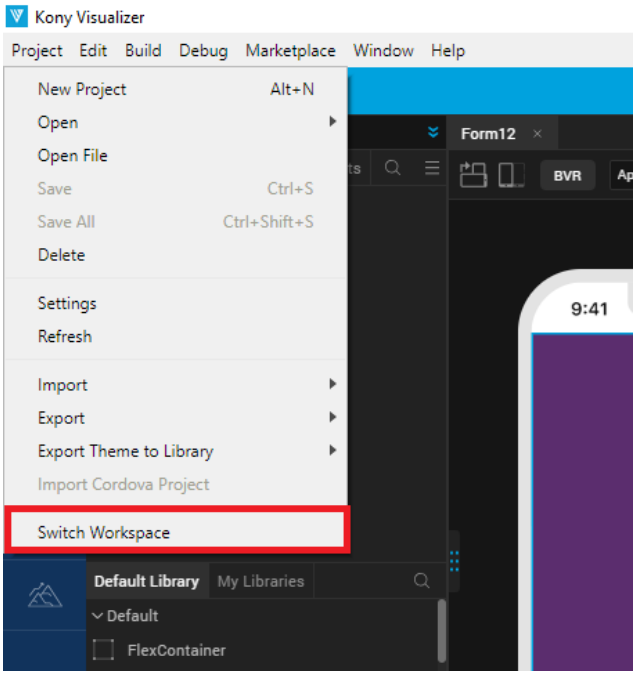
A workspace is a repository for the projects that you create, and it resides on your computer as a folder. You can keep all your projects in one workspace, or you can cluster related projects together by creating multiple workspaces. When you launch Kony Visualizer, it prompts you to indicate the workspace that you want to work in. The default workspace is home to the Kony Visualizer sample app, which resides in *C:\Users\<USERNAME>\KonyVizEWS*.

You can switch your workspace either during the Kony Visualizer startup process or from within Kony Visualizer. This feature helps you to switch the workspace when the default workspace has an issue or is not valid. Furthermore, you can switch the workspace if you want to access projects that are located in another workspace.

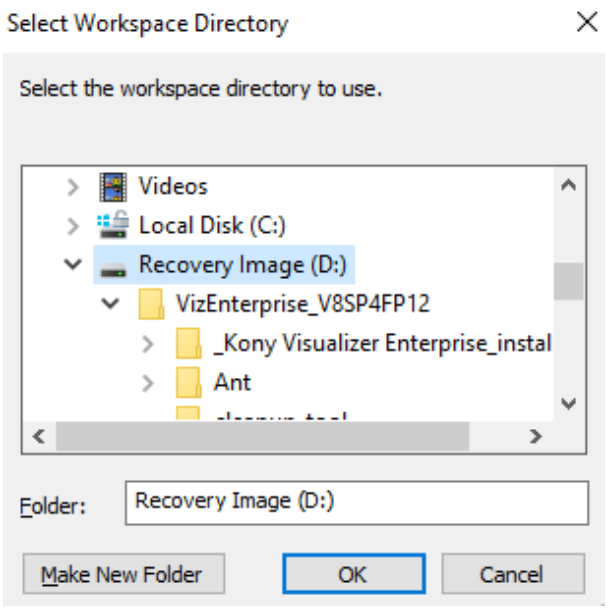
For Kony Visualizer

To switch your workspace in Kony Visualizer, follow these steps:

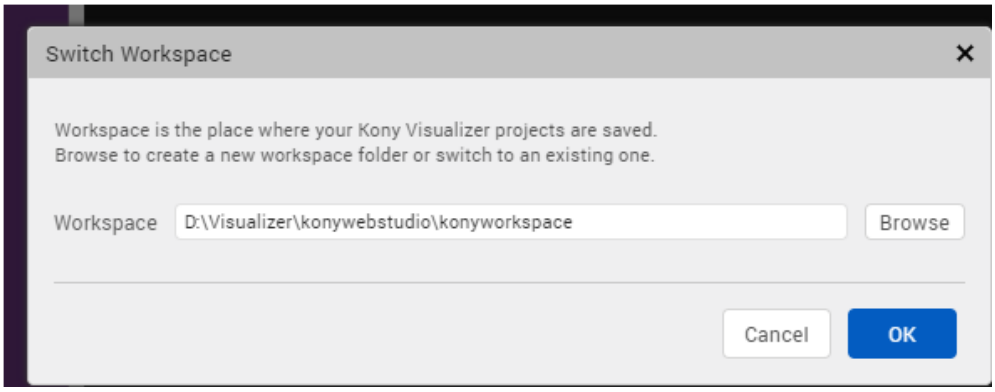
1. From the **Project** menu, click **Switch Workspace**.
   The **Switch Workspace** window appears.



2. Click **Browse**. The **Select Workspace Directory** window appears.



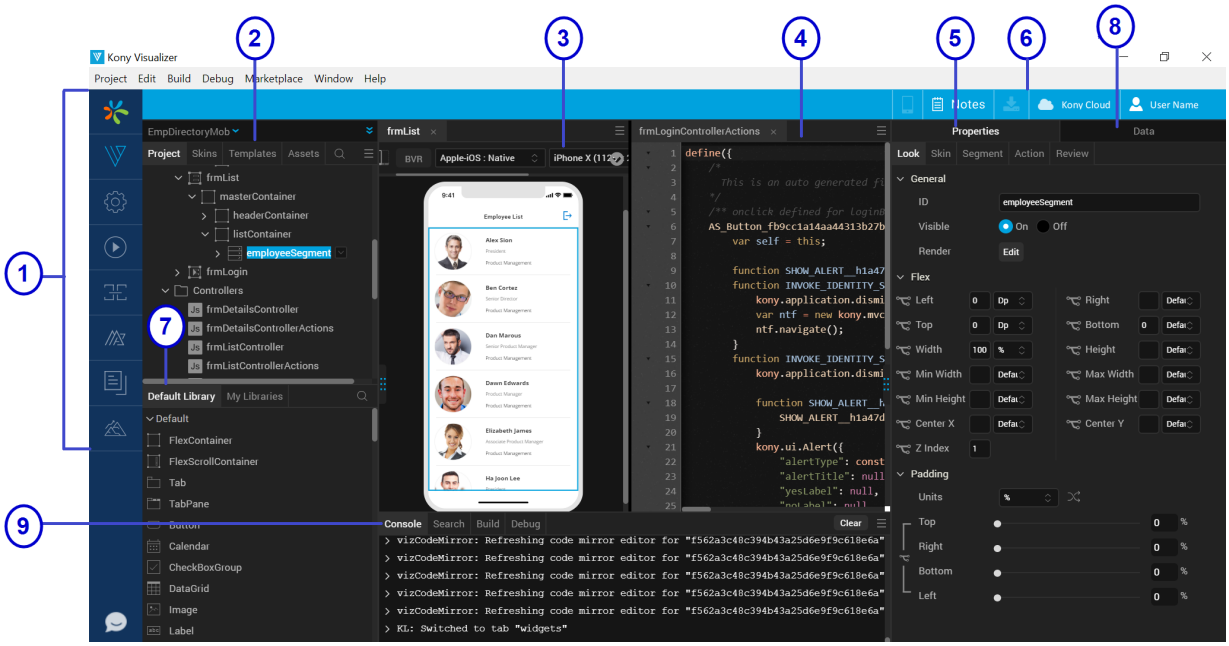
3. Select the workspace that you want to switch to, and then click **OK**.



4. Click **OK**. Kony Visualizer prompts you to save your current project if any elements require saving, and then it restarts, accessing the workspace that you just selected.

The Kony Visualizer Default Perspective

Kony Visualizer's default perspective is optimized to help you create your app. This is what it looks like, its panes labeled for the sake of illustration. Click a number to locate its description in the sections that follow.



Left Navigation Bar

Corresponding to item number 1 in the diagram, the left navigation bar gives you access to a number of frequently-used features.

Kony on web. Launches a browser window that opens the product page for Kony Visualizer.

Visualizer. This icon is selected by default. It enables you to switch to Visualizer from the Fabric environment.

Project Settings. Launches the Project Settings window.

Run. Builds and runs the project.

Kony Fabric. Launches the Kony Fabric console and opens the fabric application that is linked to the Visualizer project.

Marketplace. Launches the Marketplace window

Documentation. Launches a web browser window that opens to the documentation for Kony Visualizer.

Hikes. Launches the hike catalog.

Quantum IQ. Launches the Quantum IQ chat window that can answer your queries and give suitable suggestions from Marketplace, Base Camp, and Documention.

Project Explorer

Corresponding to item number 2 in the diagram, this pane lists the currently open application, as well as the elements that comprise it, including forms, pop-ups, templates, script modules, web modules, media resources, online services, and offline services. This is the pane you use to open forms and scripts for editing.

If the Project Explorer is too narrow, cutting off the names of various project elements and assets, you can widen it by hovering over its right edge until the cursor becomes a double-headed arrow, and then drag the Project Explorer wider.

The Project Explorer is organized into the following tabs:

- **Project.** Contains global project settings, the forms and popups of your project organized by device type, the code modules, as well as actions and services.
- **Skins.** Lists the available skins according to widget type.

- **Templates.** Lists available, pre-configured elements such as headers and footers, organized by device type.
- **Assets.** Lists the various files that comprise your application.
- **Search.** Makes it possible for you to quickly locate a project element or asset.
- **Context Menu.** Displays all the elements in the selected tab. When you click on an element, you are navigated to the selected element in the tab.

Right-clicking a file in this pane displays a set of Kony commands available for that particular file type. Double-clicking a file opens it using either the default Kony functionality (as in the case of a form), or the default Windows program assigned to that file type (such as a graphics program for an image file).

Visualizer Canvas

Corresponding to item numbers 3 and 4, the Visualizer Canvas can be divided into two side-by-side panes, as illustrated in the diagram. To split the Visualizer Canvas, on the **Window** menu, point to **Arrange**, and then click **Side by Side**. In the diagram, the side-by-side panes display a form for a mobile phone in the left pane, and the Code Editor in the right. But you can open any number of assets in these panes, such as another form so that you are displaying two forms side by side, or another editor, such as the Action Editor, where you can construct and assign actions to the widgets on your forms. Or, if you want, you can switch to single pane mode so that one pane uses the entire Visualizer Canvas.

The **BVR** button (beyond visual range) gives you the flexibility of positioning and viewing widgets beyond the visual limitations of the device's screen so that they're present, but currently not visible to the user. This is especially helpful for animations where, for example, you want a screen element to come swooping in from the side onto the screen as the result of the user doing something, such as pressing a button . You can place any widget you want beyond the visible range and then have it change its position when it's needed.

The platform-selection dropdown has a list of platforms and channels, and the device-selection dropdown has a list of devices based on the selected platform. You can select a platform, channel, and a mobile device from the dropdowns to view the form in the selected device.

At any given time, the properties of only the element that currently has focus are displayed. On the Visualizer Canvas, the name of the element with focus is highlighted. If the element with focus is either an action or a code editor, even though the Properties pane of the last-used widget appears, those properties at the moment are read-only.

> ***Note:*** To pan in BVR mode, you can press the space bar and drag.

Properties Panel

Corresponding to item number 5 in the diagram, this pane displays the properties for a number of aspect of your application, divided into five tabs:

- **Look.** Contains properties that are specific to the selected widget, such as its ID, the channels to render for, padding.
- **Skin.** Contains properties for assigning, importing, copying and pasting skins for the selected widget, and setting the properties for the skin. The properties that are available vary from widget to widget.
- **Form/ Widget.**Contains properties for the selected form or widget. They include general properties and specific properties for various platforms such as iPhone, Android, Windows8.
- **Action.** Contains general and platform-specific actions that you can apply to the current widget when a particular event occurs. For example, you can edit the action you want to take place for the event *onHide*. When you initiate the editing of an action, the Action Editor opens on the Visualizer Canvas. For more information, see *Action Editor*.
- **Review.** Provides a way to create and read notes and comments by you or others regarding forms and widgets.

Corresponding to item number 6 in the diagram, the Data & Services panel enables you to link back-end data services to your application's user interface elements seamlessly with low-code to no-code.

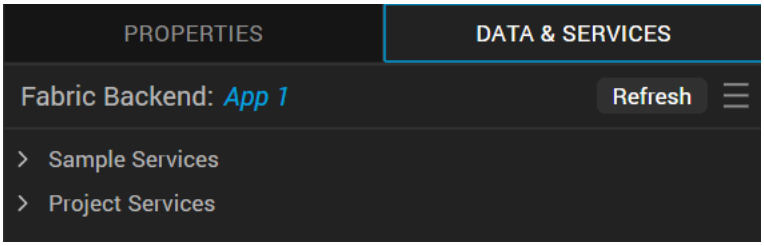
Using the sample services in the Data & Services panel, you can bind back-end services to your apps and test the user interface. If you are an advanced user of Visualizer and have previously created back-end services in your Kony Fabric instance, you can view those services in the Data panel. Further, you can create new back-end services from the Data panel and associate the services with your apps.

The Data & Services panel contains two lists:

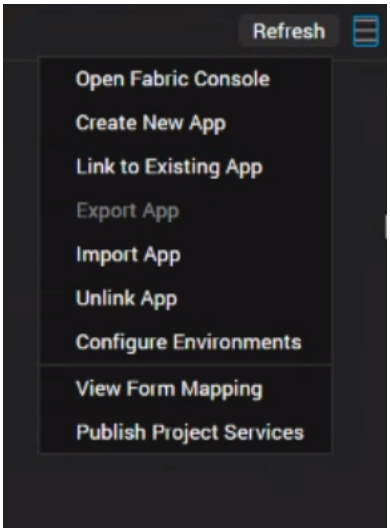
- **Sample Services.** The Sample Services drop-down list contains several sample services that you can start using straight away in your app. These sample services come pre-built with Kony Visualizer.
- **Project Services.** Using Project Services, you can create a new service.

You can also perform various other actions from the hamburger menu icon  of the Data & Services panel:

- Manually refresh the data in the Data & Services panel by using the Refresh button.



- Publish your Kony Visualizerapp to Kony Fabric, and much more.



Account Information

Corresponding to item number 8 in the diagram, the Account Information area displays a and connected devices icon, Notes button, download icon, your kony cloud account, your Kony account user name.

- **Connected device.** Displays the names of the connected devices.
- **Notes.** When you click on the notes button, it synchronizes the project notes so that you can see notes about the project that others have added.
- **Download icon.** On clicking the download icon, it notifies you about the latest Visualizer version available. You can download the latest version by clicking on the download button. It also shows the latest release notes.
- **Kony cloud account.** Displays your kony cloud account name.
- **Kony account username.** Displays your Kony account user name. If you are not logged in to your Kony account, the Account Information area displays the Login. On clicking, the Login dialog box is opened.

Library Explorer

Corresponding to item number 7 in the diagram, the Library Explorer consists of three panes: Default Library, My Libraries, and Search bar.

- **Default Library.** Contains a list of widgets, collections, and components. You can drag and drop any widget from the Default Library onto the form in the canvas.

  > ***Note:*** From Kony Visualizer V8 SP4 Fixpack 28 onwards, the Default Library filter has been enhanced to segregate components according to their respective channels. As a result, mobile-only components are displayed in the Default Library when a Mobile form or a Tablet form is open at that time on the Project Canvas. Similarly, web-only components are displayed in the Default Library when a Web form is open on the Project Canvas. However, when there is no form open on the Project Canvas, all the available components are displayed in the Default Library.

- **My Libraries.** Contains imported or custom designed collections and skins. Collections of widgets that make up a functional unit, make it easier for you to develop your app. Skins provide your app with a unified look and feel across widgets, platforms, and channels.
- **Search bar.** Enables you to search for the custom collections or skins.

Console

Corresponding to item number 9 in the diagram, the lower pane consists of the four tabs: Console, Search, Build and Debug.

- **Console.** Displays a running record of Kony Visualizer's activity, from loading a workspace and initializing services to building an app, launching it, and monitoring its activity.
- **Search.** Enables you to search for any word from the controllers and replace it everywhere with a new word.
- **Build.** Displays the build status when a build is in progress.

- **Debug.** Displays the debugger that it attached.
- **Terminal.**

Quantum IQ

Quantum IQ is a chatbot designed to help Kony Visualizer users through their journey to create applications for various channels on Kony AppPlatform. The chatbot can assist users in various ways: building a binary, importing projects, searching for help materials on Kony Base Camp in a single query, and more.

Quantum IQ interacts with users and provides answers for some FAQs. This chatbot can answer casual queries such as "how to upgrade Visualizer?" and "what is my Visualizer version?". Using Quantum IQ, users can also get information such as the version of Visualizer and Fabric that is installed on their system.

The chatbot also helps users in optimization, translation, and in helping them complete many more time-consuming tasks with ease. It uses advanced AI algorithms to provide the best-suited recommendations that enhance users' app-building experience.

> *Important:* Users must have an internet connection for Quantum IQ to work. If the network connectivity is lost after launching Quantum IQ, users must reconnect to the internet and then click the Refresh icon in the chat window.

Enable Quantum IQ

The Quantum IQ feature is available by default for all Kony developers and users who are **Early adopters** of Kony Visualizer.

Non-Kony users (who are not Early adopters) can enable the Quantum IQ feature manually. To do so, follow these steps:

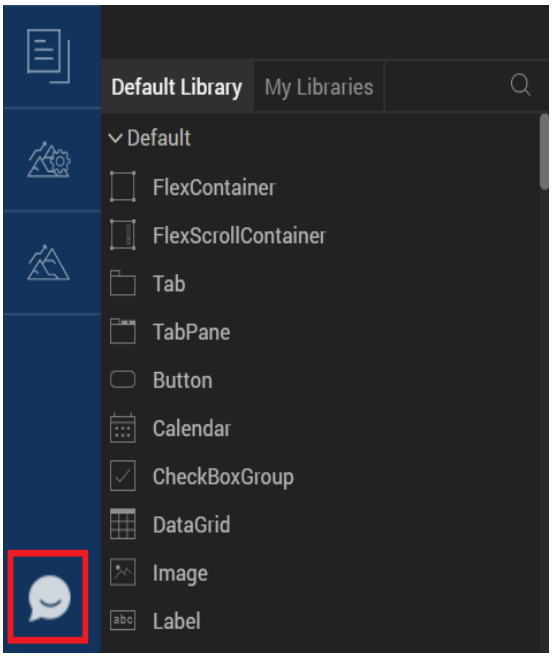
1. Open the Visualizer configuration file that is located at: **<installation>\install_config.js**
2. Search for the **enableIQChat** key. If `exports.enableIQChat!=1` or

`exports.enableIQChat=null` is already available, modify it to `exports.enableIQChat=true`. Otherwise add the `exports.enableIQChat=true` entry.

> ***Note:*** To disable Quantum IQ, modify `exports.enableIQChat=true` to `exports.enableIQChat!=1`.

3. Save and close the **install_config.js** file.
4. Restart Kony Visualizer.

   The Quantum IQ icon appears at the lower-left corner of Visualizer.



Interact with Quantum IQ

In Kony Visualizer, you can find the Quantum IQ icon at the lower-left corner of the canvas. When you click on the icon, a chat window opens; using which you can start interacting with Quantum IQ.

To open the Quantum IQ, follow these steps:

1. Open Kony Visualizer.
2. From the left navigation bar, click on the Quantum IQ icon.
   The Quantum IQ chat window is displayed.
3. Enter your query into the chat box. Quantum IQ then provides an appropriate response.

Quantum IQ responds in different ways based on the type of query entered in the chat box. The types of responses of Quantum IQ when you enter a query are as follows:

- Provides an appropriate response in the chat window.
- Performs action on Kony Visualizer.
- Prompts for more information. In this case, Quantum IQ requires more information to provide the appropriate response for your query. If you do not want to continue the conversation, you can enter **cancel** in the chat box.

> ***Note:*** In Kony Visualizer, you can move or resize the Quantum IQ chat window. Drag the lower-right part of the chat window to resize it.
> You can also move the Quantum IQ chat window out of Kony Visualizer. Click the pop-out icon at the upper-right corner of the chat window to move it out of Kony Visualizer. In the popped-out window, you can click the pop-in icon at the upper-right corner of the chat window to move it inside the Kony Visualizer.

Capabilities of Quantum IQ

Quantum IQ can perform the following tasks:

- Design Suggestions
- Generic search
  - Search from Kony Marketplace
  - Search from Kony Base Camp

    - Search from YouTube
    - Search from Hike Catalog
- Translate project to different languages
- Visualizer actions
    - Clean up project
    - Import/ Export project (Local/ Cloud)
    - Import components (Local/ Cloud)
    - Run project
    - Build project

Quantum IQ Design Suggestions

The Quantum IQ Design Suggestions feature assists you to develop applications faster. Quantum IQ predicts the UI of your form and recommends relevant designs. These predictions can help you to reuse existing components and designs, without having to create them from scratch.

For instance, if you design a form with two TextBox widgets, and one Button widget, Quantum IQ predicts that you could be creating a Login form and recommends designs such as the Login component from Kony Marketplace.

Furthermore, Quantum IQ can detect duplicate designs in your project. To fetch the duplicates of a design, type a command, such as “identify duplicates”, in the Quantum IQ chat window and press Enter. A list of duplicate design groups is displayed in the Quantum IQ Design Suggestions window. You can then view each duplicate design and copy it to a form.

Interact with Quantum IQ Design Suggestions

While you are designing a form, Quantum IQ compares the UI of the form with the existing designs from the current project and Kony Marketplace. Quantum IQ then generates a list of recommended designs based on the visual, textual, and widget hierarchical similarities of the designs.

> ***Note:*** For Quantum IQ to show design suggestions for a form, you must add at least three widgets to the form.

When similar designs are detected in the current project or in Kony Marketplace, a Quantum IQ bubble appears on the lower-left corner of the screen. The bubble provides the following options:

- **Yes:** Enables you to view a list of design suggestions in the **Quantum IQ Design Suggestions** window.
- **Pause:** Enables you to pause the design suggestions for an hour.
- **Turn Off Suggestions:** Enables you to disable the design suggestions until they are re-enabled.

Here is the list of sources from which you receive the suggestions:

- Current project
- Kony Marketplace
- Kony Library

Designs from the Current Project

You can perform the following actions on a design suggestion from the current project:

- **Navigate:** Navigates you to the location of the design in the project.
- **Copy:** Copies the design to the clipboard.
- **Convert to component:** Enables you to convert a design suggestion into a component without contract, and imports it into the project.

Components from Kony Marketplace

You can perform the following actions on a design suggestion from Kony Marketplace:

- **View:** Opens the component in Kony Marketplace and displays details of the component.
- **Download:** Downloads the component into your local system.

> ***Note:*** Quantum IQ Design Suggestions is an offline feature. However, you must have an active Internet connection to download a recommended Marketplace component.

Designs from Default Library

You can perform the following action on a design suggestion from the Default library:

- **View:** Displays the suggested design in the **Default Library** section of Visualizer.

Enable/ Disable Design Suggestions

In Visualizer, you can enable or disable the design suggestions from the main menu directly. To do so, follow these steps:

1. From the main menu, click **Help**.
2. If you've paused or turned off design suggestions, you will see **Enable Design Suggestions**. Click **Enable Design Suggestions** to fetch the design suggestions for the current form.

   If the design suggestions is already enabled, you will see **Disable Design Suggestions**. Click **Disable Design Suggestions** to stop the design suggestions from being displayed.

You can also enable or disable the design suggestions by using a command in the Quantum IQ chat window. For example, you can enter commands such as stop design suggestions, or turn on suggestions.

Generic Search

The Generic search functionality helps you fetch data from Marketplace, Base Camp, and YouTube, based on the availability of results.

Here are some generic search queries:

- Segment
- What is DBX?

- How to integrate FFI?

Kony Marketplace Search

The Kony Marketplace search feature is integrated with Quantum IQ. Using the bot, you can search for components from Kony Marketplace, and import those components into your Visualizer project.

To fetch results exclusively from Marketplace, enter @marketplace <Text query to search in Marketplace>

Here are a few Kony Marketplace search queries:

- @markerplace Placelocator
- @marketplace login
- @marketplace Rangeslider

Kony Base Camp Search

The Kony Base Camp search functionality is integrated with Quantum IQ. Using the bot, you can search for articles and documents in Base Camp. The bot shows a list of articles that are related to your query.

To fetch results exclusively from Kony Base Camp, enter @basecamp <Text query to search in Base Camp>

Some Base Camp search queries are as follows:

- @basecamp NFI examples
- @basecamp error in importing component
- @basecamp what is preprocessor?

YouTube Search

The search in the Kony YouTube channel is integrated with Quantum IQ. Using the bot, you can search for any videos on the Kony YouTube channel.

To fetch results exclusively from YouTube, enter @youtube <Text query to search in YouTube>

Here are a few YouTube search queries:

- @youtube banking
- @youtube javascript
- @youtube creating a component

Hike Search

Using Quantum IQ, you can search for any hike that is available in the hike catalog. When you type a query in the Quantum IQ chat window, you will see a list of related hikes under the HIKES tab.

Each result displays basic information of a hike such as the name, number of steps, and a **PLAY HIKE** button. You can click **PLAY HIKE** to open the hike in Kony Visualizer.

To fetch results exclusively from the hike catalog, type @Hikes < Text query to search in Hike Catalog >

For example:

- @hikes Creating a component
- @hikes Action editor
- @hikes Identity services

Translation

Quantum IQ helps you translate an application into different languages by using the Google translate API. During translation, the text in an application is converted into equivalent i18n keys in the target language. The properties of widgets that are set through the Properties panel and through code are translated.

Default locale is the language that is displayed in an app when the app is launched during run time. You can configure or modify the default locale of an app from the Project Settings of Visualizer. If the default locale is not set already, Quantum IQ updates the default locale to the language that you want to translate your app into.

Translation can be done in two ways:

- **Specify the target language:** You can specify the target language directly in your query.
- **Bot prompts the language:** You can enter your query to translate the project without specifying the target language.

Here are a few queries about translation:

- Can you translate this app to German?
- Translate this app to Spanish
- Localize to Spanish
- Can you translate this app?
- Translate my app

Visualizer Actions

Quantum IQ helps you save time by triggering small actions on Kony Visualizer, using commands in the bot. For the list of Visualizer actions that are supported on Quantum IQ, refer Visualizer actions.

Following are a few Visualizer actions.

Project Clean Up

Cleaning unused skins and actions from a Visualizer project improves the performance of Kony Visualizer at run time. Using Quantum IQ, you can clean up the unused skins and actions from Visualizer project. It also gives the count of unused or duplicate elements, and provides an option to delete them.

Some queries about project clean up are as follows:

- Clean my project
- Sanitize my project
- Project clean up
- Clean unused items

Import and Export Project (Local/ Cloud)

The bot helps you import and export projects into Kony Visualizer.

- **Import Project:** You can import the projects that are stored in the local storage or on the cloud. When you import a local project, the bot shows a list of related local projects in the file explorer. You can select the project that you want to import. When you import a cloud project, the cloud account window appears. You can then select the cloud project that you want to import.
- **Export Project:** You can export a project to the local storage or to the cloud. When you export project locally, the current Visualizer project is zipped and stored in the local storage. When you export project to cloud, the current project is shared to cloud.

Here are a few queries about importing a project:

- Import project from cloud
- Import project from local

- Import project
- Export project

Import Component (Local/ Cloud)

You can import the following types of components into Kony Visualizer:

- **Local components:** These components are imported from the local storage. When you import a local component, the bot shows you a list of components in the file explorer. You can then select the component that you want to import.
- **Cloud components:** These components are imported from Kony Marketplace. When you search for a component from cloud, the Kony Marketplace window appears. You can search for a component and import the component from the Marketplace window.

Here are a few queries on importing components:

- Import local component
- Import component
- Import component from cloud

Run Project

Using the bot, you can run your current project. The bot helps you generate a live preview by triggering the action directly on Visualizer.

Some queries about running the project are as follows:

- Run for iPhone
- Run for Android
- Run for web and Android

Build Project

Quantum IQ helps you build the current project for a required platform. You can generate the binaries by entering a build command in the bot. The bot triggers the build action directly on Visualizer.

Here are the queries on building your project for various platforms:

- Build project
- Build for Android
- Build for iPhone
- Build for Android tablet
- Build project for iPad
- Build for Android SPA
- Build my project for iPhone SPA
- Build for iPad SPA
- Build project for Android tablet SPA
- Build for Windows tablet
- Build project for Windows tablet SPA
- Build for Windows
- Build for Windows SPA
- Build for Desktop Web
- Build for Desktop Kiosk

Visualizer Actions in Quantum IQ

Quantum IQ supports the following list of Visualizer actions.

Action	Description	Query Examples
Add project to current	Imports a new project to the current project.	• add project here • import project here • import to current project • add to current project
Browse Marketplace	Opens the Kony Marketplace window.	• browse Kony Marketplace • open kony Marketplace • open Marketplace • browse Marketplace
Build and publish web	Opens the **Build and Publish Web** window.	• build and publish web • build and publish web application • build web

Build and publish native	Opens the **Build and Publish Native** window.	• build and publish native • build and publish native application • build native
Check for updates	Displays the latest version of Kony Visualizer, if any. You can click **Download** to upgrade your Kony Visualizer to the latest version.	• Check updates • Are there any new updates?
Close all	Closes all the open tabs in the canvas.	• close all tabs • close tabs
Create new form	Creates a new form for the specified platform. ***Note:*** If a platform is not mentioned, a mobile form is created.	• create a new form • create a desktop form • open a mobile form • create a new form for tablet

Create new project	Opens the **New Project** window.	• setup a new project • start a new project • begin a fresh project
Debug app	Provides you with two options for debugging an app: Debug Live Preview and Debug Native App.	• Debug • Debug android app • Debug iOS app • Debug live preview on android
Deep linking project properties	Deeplink to a property in Kony Visualizer Classic; opens the tab in which the property is present.	• change app id to <name> • set application id to <name>

Download Kony Quantum App	Opens the Kony Quantum App dialog box.	• Download Kony Quantum App • Download Preview app • Where can i find Kony Quantum App?
Edit i18n	Opens the **Configure Internationalization** window.	• Edit locales • Configure locales
Export fonts	Exports fonts	• export Font • export Fonts
Export i18n resources as JSON, CSV	Exports i18n resources as JSON and CSV.	• export I18N as json • export i18n as csv • export i18 export i18n
Export themes	Exports themes	• export theme • export themes
Import Cordova project	Imports a Cordova project into Kony Visualizer.	• Import Cordova project

Import fonts	Imports fonts into the current project.	• import font • import fonts
Import i18n resources	Imports i18n resources into the current project.	• import locale • import i18n • import i18
Import themes	Imports themes into the current project.	• import theme • import themes
Live Preview settings	Opens the **Live Preview Settings** window.	• Open Live Preview settings • Open Preview settings • Open Live Preview
Manage Cordova plugins	Opens the **Configure Cordova Plugins** window.	• Manage Cordova Plugins • Add Cordova Plugins • Remove Cordova Plugins

Manage native functions API	Opens the **Native Function Interface** window.	• open NFI settings • open native functions • Manage native functions api
Open Freeform project	Opens a specific Freeform project.	• open Freeform project <name>
Open MVC project	Opens a specific MVC project.	• open MVC project <name>
Open preferences	Opens the **Visualizer Preferences** window.	• Open Preferences • Edit Preferences
Open project	Opens a specific project.	• open project <name>
Open project settings	Opens the **Project Settings** window.	• open settings • open project settings
Open terminal	Opens a terminal tab in the lower pane of Kony Visualizer.	• enable terminal • open terminal

Open tutorials	Opens the Hikes catalog.	• open tutorials • where can i find tutorials?
Publish app to Marketplace	Opens the **Publish to Kony Marketplace** window	• publish to Marketplace • publish my app to Marketplace
Publish live preview	Opens the **Publish Live Preview** window.	• publish live preview
Refresh project	Refreshes the project.	• refresh • refresh project • refresh Visualizer
Save all	Saves the current project.	• save project • save current project
Save current	Saves the current tab or the form in the canvas.	• save current tab • save current form

Show global variable	Displays the **Variables** window.	• Manage global variable • show global variable • change global variables
Update project properties	Updates the project properties such as app id and version.	• update app id key to <name> • modify app version key to <name>

Check for Updates

Kony is constantly making improvements to Kony Visualizer, which you can install as updates.

To check for updates to Kony Visualizer, do the following:

1. Do one of the following, depending on what type of computer you have:
    - On a Mac, click the **Kony Visualizer** menu, and then click **Check for Updates**.
    - On a Windows PC, click the **Help** menu, and then click **Check for Updates**.
2. Follow the prompts of the Updates dialog box.

Collab

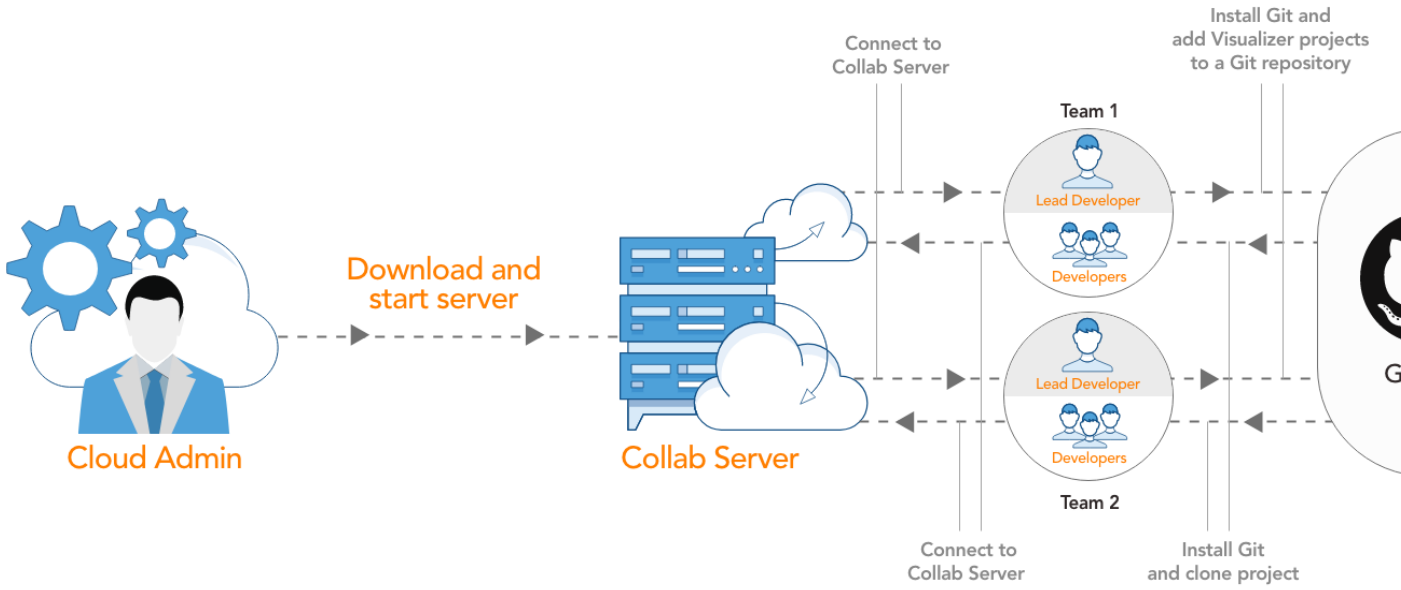
Developing applications using Kony Visualizer, as in the case of developing any software, involves collaborating with other developers. While collaboration accelerates app development, improves efficiency, and saves time; it brings its own set of issues. One of the main issues is code merge conflicts.

Kony Visualizer V9 introduces a new feature, Collab, which enables you to coordinate with the developers working on a single Kony Visualizer project while using Git as the collaboration tool.

Collab tracks the activity of developers working on the same project and displays alert messages if multiple developers work on a single file. The Collab window highlights merge conflicts that may occur when code is pushed to a remote project in the GIT repository.

There are three stages involved in the process of installing and configuring Collab:

- Download and Start the Collab Server – One-time activity usually performed by an admin in an organization.
- Install and configure Git – One-time activity performed by all the developers in a project.
- Add a Visualizer project on GIT and collaborate – Pushing a Visualizer project on Git is a one-time activity and is usually done by the project lead. The developers then clone the project and collaborate to work on their local projects.



Download and Start Collab Server

The first step in making Collab work is to download the Collab server. The administrator in an organization must download the Collab server and share the server's address with developers working on a project. All the developers working on the project must connect to the server.

Download and Configure Collab Server

In the Collab Server, you can configure the sever security to Http or Https. Follow the steps respective to the Server protocol you want to configure, to download and install the Collab Server.

Http Sever protocol

Follow these steps to download and configure the Collab Server:

1. Download the Collab server from here.
   The installer is downloaded as a zip file to your computer.
2. Extract the contents of the zip file.
3. Navigate to the folder specific to the OS you are using and open the **collabConfig.json** file with a text editor.
   i. On a Windows machine, navigate to the **CollabServer** > **CollabServer_WIN** > **collabConfig** > **collabConfig.json** file.
   ii. On a Mac machine, navigate to the **CollabServer** > **CollabServer_MACOS** > **collabConfig** > **collabConfig.json** file.
4. In the **collabConfig.json** file, configure the **Port** key to an available port number on the server.
5. Save and close the file.

Https Sever protocol

Follow these steps to download and configure the Collab Server:

1. Download the Collab server from here.
   The installer is downloaded as a zip file to your computer.
2. Extract the contents of the zip file.

> ***Note:*** Ensure you have your HTTPS server key and server certificate with you.

3. Place your HTTPS server key and server certificate in the collabConfig (specific to the OS you are using) folder.
   i. On a Windows machine, navigate to the **CollabServer** > **CollabServer_WIN** > **collabConfig**
   ii. On a Mac machine, navigate to the **CollabServer** > **CollabServer_MACOS** > **collabConfig**
4. Navigate to the folder specific to the OS you are using and open the **collabConfig.json** file with a text editor.
   i. On a Windows machine, navigate to the **CollabServer** > **CollabServer_WIN** > **collabConfig** > **collabConfig.json** file.
   ii. On a Mac machine, navigate to the **CollabServer** > **CollabServer_MACOS** > **collabConfig** > **collabConfig.json** file.
5. In the **CollabConfig.json** file, configure the **Server_protocol** key to **Https**.
6. Enter the server key and the server certificate file names for the **Server_key** and **Server_cert** keys respectively.
7. Configure the **Port** key to an available port number on the server. By default, the Port key is set to 30000.
8. Save and close the file.

Run the Collab Server

Once you have configured the Collab server, you must start the server. The process of starting the server varies depending on the OS you are using. For instructions on the Windows machine, click here. For information on Mac machine, click here.

Run the Collab Server on a Windows Machine

To run the Collab server on a Windows machine, follow these steps:

1. From the file explorer, navigate to the location where the Server is downloaded.
2. Navigate to **CollabServer** > **CollabServer_WIN**
3. Double-click the **CollabServer.exe** file.
   A terminal opens, and the server starts running.

> ***Note:*** Close the terminal to stop the server.

Run the Collab Server on a Mac Machine

To run the Collab server on a Windows machine, follow these steps:

1. Navigate to the location where the Collab Server is downloaded.
2. Copy the **CollabServer** folder.
3. From the main menu, navigate to **Go** > **Home**.
4. Paste the **CollabServer** folder in the Home directory.
5. Navigate to **CollabServer** > **CollabServer_MACOS**.
6. Double-click on the **CollabServer** file.
   A terminal opens, and the server starts running.

Your Collab server is now installed and configured. You can now start associating the server with the Visualizer project in your GIT repository.

Install and Configure Git

To install and configure Git for the first time, follow these steps:

> ***Note:*** Skip these steps if you already have GIT installed and configured.

1. Download and install Git from here.
   Ensure you have an account in Git. Create an account if you do not have one already.
2. Open the terminal.
3. Enter the following command to configure your username:

   ```
   git config --global user.name "UserName"
   ```

4. Enter the following command to configure your email address:

   ```
   git config --global user.email "emailID"
   ```

Your email address is now associated with the local commits you make to a project.

Add a Visualizer project on Git and Collaborate

Once Git is installed and configured on your system, you can add the project to a repository in Git. The project lead creates a repository in Git and adds a Visualizer project to the repository. The developers working on this project can then clone the project and push the changes to the remote project in the repository.

Create a New Repository in Git

Ensure you have created a new repository in Git. For more information on how to create a repository, refer to Github documentation.

Add Visualizer Project to Git Repository

Once a repository is created in your Git account, you can now add a Visualizer project to the repository.

To add an existing Visualizer project to the repository, follow these steps:

1. Open the Visualizer project.
2. In the **Terminal** tab of Kony Visualizer, enter the following commands to push the code to your repository:

```
cd <project name>
git init
git add –all
git commit -m “Initial Commit”
git remote add origin <web URL to where you want to raise a pull request>
git push -u origin master
```

   The project files are uploaded to the specified Git repository.

Clone the Visualizer project from the Repository

Once the Visualizer project is added to a repository in Git, developers working on the project can clone the project from the repository and work on a local version of the project.

To clone an existing project from a repository, follow these steps:

1. Open the Visualizer project.
2. In the **Terminal** tab of Kony Visualizer, enter the following commands:

```
pwd //opens current working directory is the workspace
git clone <URL of the project in the repository>
```

3. From the main menu, navigate to **Project** > **Open**, and click **Refresh**.
4. Navigate to **Project** > **Open** > **Kony Reference Architecture/Free Form JavaScript**.

5. You will see the cloned project in the list of projects that appears.
6. Select the project that you have downloaded.
   The cloned project opens.
   The developers can further work on the local project and push the changes to the remote project in Git repository.

Using Collab

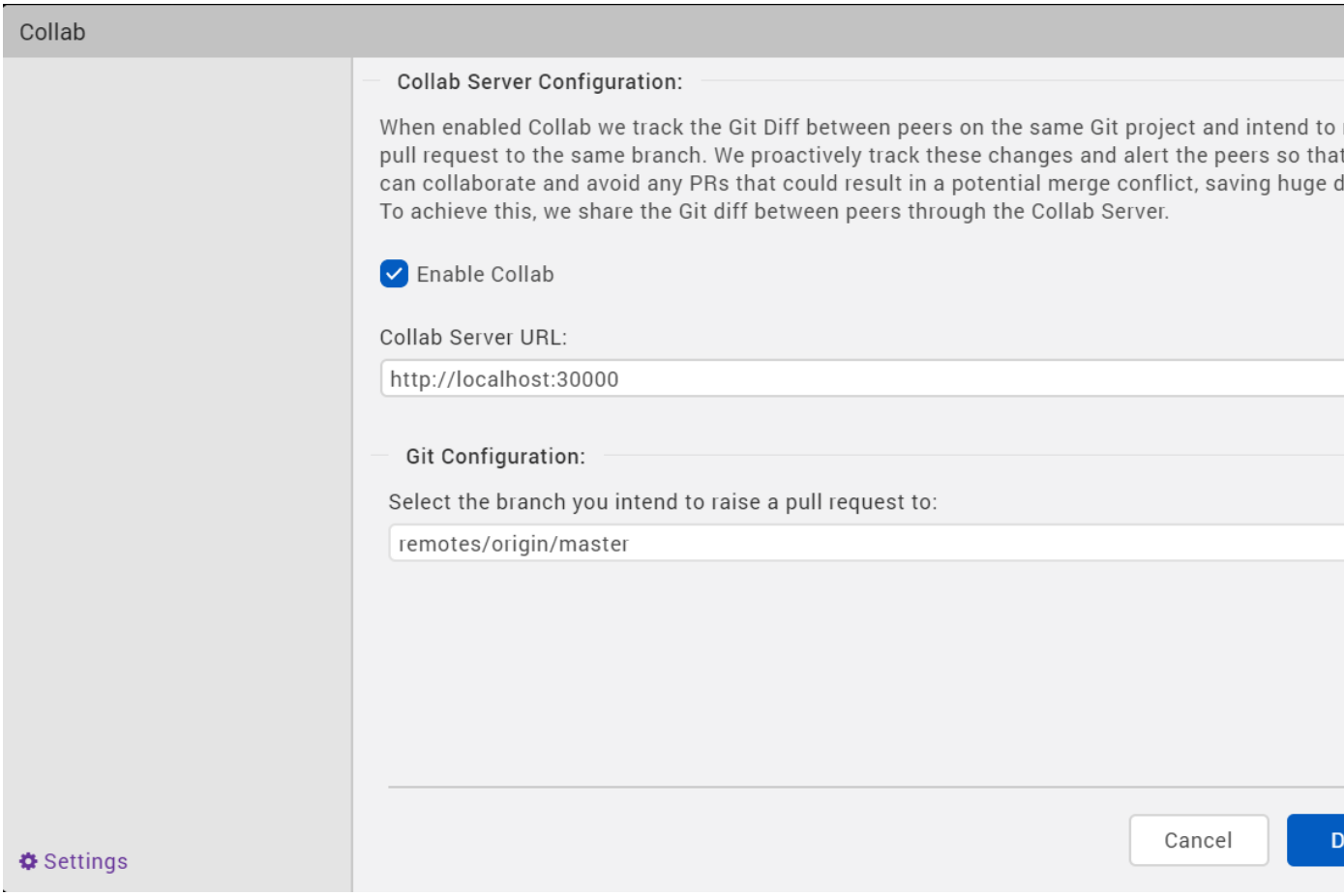
Once the Git and the Collab Server are setup, you must configure the Collab settings in Visualizer to start using Collab.

Configure the Collab Settings in Visualizer

Before you get started with using the Collab feature, you must configure the Collab Server to connect to the remote project that is hosted on the Git repository.

To configure the Collab settings in Kony Visualizer, follow these steps:

1. In Visualizer, from the **Project** menu, go to **Window** > **Collab**.
   The **Collab** window appears.



2. In the **Collab Server Configuration** section, enter the local address where the Collab server is installed. The format of the server URL is `https://<IP address of the machine on which the server is installed>:<port number>`
3. Select the **Enable Collab** check box.
4. In the **Git Configuration** section, select the remote branch to which you want to raise a pull request.

You have successfully configured the Collab settings in Visualizer.

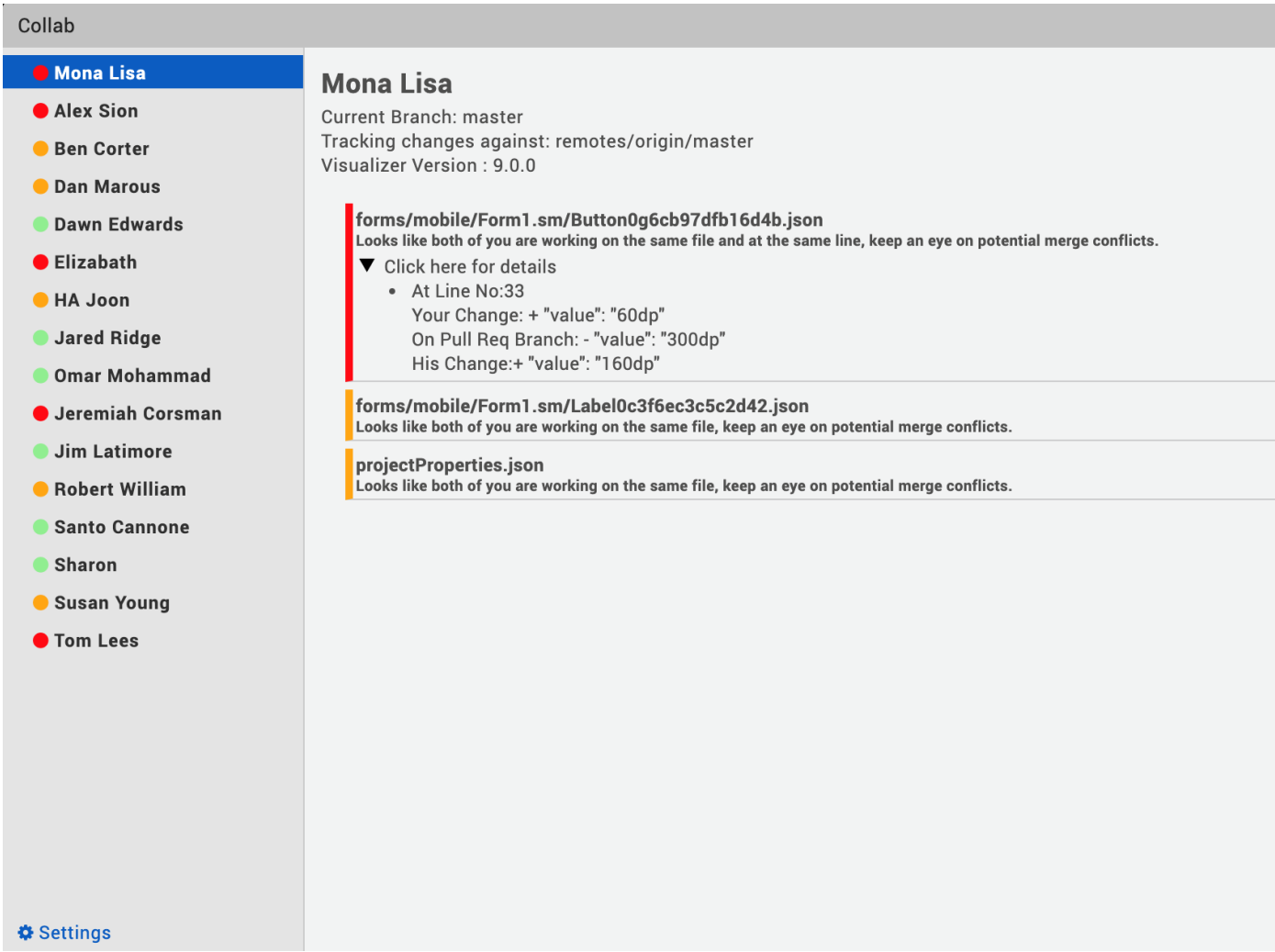
Interact with Collab

After configuring all the required settings, and developers start working on their projects, Collab starts tracking the changes that developers make to their local projects.

Collab detects potential conflicts among developers that push their code to the same branch in a Git repository and displays an alert on the screen.

To view more details about the possible merge conflicts, follow these steps:

1. In Visualizer, from the **Project** menu, go to **Window** > **Collab**.
   The **Collab** window appears.



The left panel of the window displays the list of developers working on the same project and the risk of causing a merge conflict.

A merge conflict can occur when a developer pushes the code to the same branch in the Git repository that you have been working on. The magnitude of the risk that can cause a merge conflict is represented based on the color of the badge:

- **Red:** There is at least one file that can cause merge conflict while pushing the changes into remote repository in Git.
- **Yellow:** There is at least one file in common that is edited by both developers, but there are no possible merge conflicts in it.
- **Green:** There are no files in common that are edited by both the developers.

The right panel displays the possible merger conflicts with the selected developer. The color of each potential conflict is represents the following:

- **Red:** Multiple developers are working on the same line in a project file. This will result in a merge conflict when the code is pushed to the repository.
- **Yellow:** When multiple developers work on the same file in a project but on different lines in the file. This might create a merge conflict when the code is pushed to the repository.
- **Green:** When multiple developers work on the same project but on different files or forms. This will not result in a merge conflict when the code is pushed to the repository.

2. Click on a potential conflict to view more details about the conflict.

You can now coordinate with other developers to avoid merge conflicts.

Version Control

Version control is an option available **Windows** menu in Kony Visualizer. When you click the Version Control, the default GIT GUI client is launched. If the client is not available, an error displays. You must ensure that the GIT GUI client is already configured on your developer's machine. Once the GIT GUI is open and working, you can commit, revert, push, or perform any other version control operations available in the GIT GUI client.

To initiate version control in Kony Visualizer, do the following:

1. On a Windows PC, click the Kony Visualizer **Windows** menu, and then click **Version Control**. The Git GUI displays.
2. Follow the procedure in the Git GUI to use the version control.

Designing an Application

You can use Kony Visualizer to easily create digital applications across multiple channels – phones, tablets, wearables, and desktops. You can use existing sample applications and components, available on Kony Marketplace, or build your applications from scratch.

The following topics cover some of the key tasks in planning and creating your application:

Types of Applications

Ensure You Have All the Resources You Need

Plan Your Mobile App

Create, Migrate, or Import a Project

Create Screens

Populate Screens with Widgets

"Organizing and Moving Application Elements" on page 628

Set App Lifecycle Events

Using Native Function APIs and Widgets

Add Actions

Understanding Skins and Themes

Add and Manage Images and Other Media

Preview and Collaborate on a Project

Capture Product Requirements with Review Notes

Time and Effort Savers

Types of Applications

From Kony Visualizer 7.3, you can create several different types of digital application projects.

You can create, build, and deploy applications using various approaches. At one extreme, you can use a given platform's native SDK, while at the other extreme, you can use traditional web technologies. And in between these extremes, you can use a combination of the two. The build and deployment modes available in Kony Visualizer are as follows:

- **Native.** Representing one extreme, these applications reside on the device, where all the forms and user interface (UI) definitions of the widgets are packaged along with the application.
- **Hybrid.** These applications are a variant of native applications, where the layout, rather than being rendered by native SDK widgets, is rendered using a native shell provided by the respective platforms (*WKWebView* for iPhone and *WebView* for Android). The native SDK is invoked only for native device features like Camera, Contacts, and so on. Hybrid applications are supported only on the iPhone and Android platforms.
- **Single Page Application (SPA).** With these web applications, all the HTML, CSS, and JavaScript is retrieved with a single page load. The page does not automatically reload during user interaction with the application or control the transfer to another page. Once loaded, the application requires server interaction only for data retrieval. Unlike a traditional mobile web application, the native browser on the device executes all the application logic locally, including validations, UI rules (hiding or showing fields), form transitions/navigations, and so on. In a SPA, all the application screens are expressed as a JavaScript model/CSS (unlike JSP, or ASPX in a typical Mobile Web application). This JavaScript model is downloaded to the client-side with the first page download and resides on the client for the life of the session. SPA also supports Internet Explorer 10 from release 5.5 onwards.
- **Desktop Web.** These run on a desktop web browser. The supported browsers include Firefox 3.6 and above, Internet Explorer 8 and above, Safari, and Chrome. Desktop Web applications can only be built for JavaScript projects.

The characteristics of each of the application types are summed up in the following table.

Characteristics	Application Type			
	Native	Hybrid	SPA	Desktop Web
Is available as a natively deployable package (.app , .apk, .bar, .xap, and .ipa)	Yes	Yes	No	No
Uses the native widgets, such as the title bar, application menu, ListView, TableView, Search bar, and so on.	Yes	No	No	No
Has access to native device functions like GPS, Camera, Contacts, Accelerometer, Encryption libraries (and thousands of other native SDK functions)	Yes	Yes[1][1]	Yes[2][2]	Yes[3][3]
Can be distributed through the app stores (Apple App Store, Google Play, and Windows Phone App Marketplace)	Yes	Yes	No	No

[1] Native SDK functions are accessed using JavaScript bindings.
[2] Only the APIs exposed and implemented as part of the HTML5 specification.
[3] Only GPS is supported.

[1] Native SDK functions are accessed using JavaScript bindings.
[2] Only the APIs exposed and implemented as part of the HTML5 specification.
[3] Only GPS is supported.

Characteristics	Application Type			
	Native	Hybrid	SPA	Desktop Web
UI elements (forms, images, and internationalization content) are packaged along with the application	Yes	Yes	No	No
Server access only for downloading data	Yes	Yes	Yes[1][1]	

[1] After the first page is downloaded, the application accesses the server only to fetch business data.

[1] After the first page is downloaded, the application accesses the server only to fetch business data.

Ensure You Have All the Resources You Need

The sections that follow assume that you have already set up Kony Visualizer to create and build applications for the platforms you want to deploy your application to. Doing so includes configuring Kony Visualizer, and installing platform SDKs and their emulators. If you have not yet completed those tasks, see Configuring your computer.

Plan Your Mobile App

You can think about the planning of your digital app in terms of three phases: analyzing, conceptualizing, and visualizing. In this topic we look at these, including some additional information about planning your app for wearables, such as smart watches.

Analyze What Your App Needs

Using a number of inputs such as business drivers, competitive analysis, customer feedback, and market opportunities, you want to evaluate what you want your app to do and who it's for. What are the most essential features your app requires to make it a minimally viable product? Knowing your audience is key: what distinguishes your audience from other mobile users, and what niche is available in the app market that other similar apps currently aren't filling? Develop a very detailed user profile, using the 80/20 rule: what characterizes 80% of the people who will use your app?

Conceptualize Your App

Develop a list of features for your app, and then compare that list against your detailed user profile. What's needed and what isn't? Narrow the scope of your app so that it is user-driven, easy to use, focused in its purpose, meets your business objectives, ideally fills an untapped niche, and is technically feasible for your company's IT infrastructure.

Visualize Your App

Using Kony Visualizer, you want to visualize your app's flow and hierarchy. In essence, you want to create a screen-by-screen blueprint of your entire app, ensuring that all the requirements are met and features accounted for. You're not going for the look and feel here so much as the functional flow of how the app will work. That's not to take anything away from the importance of the interface, but this

matter of aesthetics—how the appearance and behavior of the app integrates with its function, simply cannot be effectively created until you have a detailed understanding of what your app will do from top to bottom. After visualizing your app's functionality screen by screen, you may decide that you need to change the scope of your app—usually narrowing it and making it more focused and easy to use.

Capture Product Requirements

Using the Review Notes feature in Kony Visualizer, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users.

Plan Your App for Wearables

Glance-ability. Utility. Intentionality. These are the three watch words when it comes to planning your app for wearables. And as much as your app likely began as an offshoot of a desktop context, apps for wearables are adaptations of mobile apps—a phenomenon reinforced by the reality that many of the most popular wearable devices available today draw much of their functionality from pairing with the mobile device.

Since this is the case, your easiest path to a version of your app for wearables will come in adapting a subset of existing functionality in your mobile app for wearable products. But what features should you adapt? That's where the concept of glance-ability comes in.

To succeed in the wearables market, your wearable app must be glance-able; users must be able to access information or gain the benefit they're seeking quickly, easily, and with as little effort and time expended as possible. This should lead to asking yourself what features or functions of your current mobile app are glance-able.

Once you address what features of your app are glance-able, you need to consider their utility. Do they truly benefit the user, and what would it take for them to become so? What kinds of notifications will you offer and why would they be of benefit to the user? This leads to the concept of intentionality.

To create a successful wearable app, you need to be very intentional about offering the greatest amount of utility in the most glance-able way possible. This can mean conducting user surveys and usability tests, as well as that much-sought-after lightning strike of discovering a feature or implementation that users don't realize they would use, but once they do, they love it. Whatever you choose to do for your wearables app, ensure that it's glance-able, offers maximum utility, and is implemented with a lot of intentionality to ensure that it meets or exceeds the user's expectations.

Create, Migrate, or Import a Project

Whether it's an application for phones, tablets, wearables, or desktops, the process of creating an application in Kony Visualizer involves creating a *project*. You can create a project from sample applications and components, or from scratch. You can also import an existing project into your project, or migrate a project that was created in an earlier version of Kony Visualizer, or in Kony Studio.

The following topics describe how to create, import, or migrate a project:

Types of Projects

Create a Project from Sample Applications and Components

Create a Kony Reference Architecture Project

Create a Free-Form JavaScript Project

Create an Application Extension

Migrate a Project from an Earlier Version of Kony Studio or Visualizer

Import a Kony Visualizer Project

Import an Application Extension

Export a Kony Visualizer project

Integration with Design Tools

Types of Projects

Starting with Kony Visualizer 7.3, you can choose the type of custom project you want to create when you select the **New Project** command from the **File** menu (the **Project** menu in *Kony Visualizer*) to start the **New Project** wizard. In Kony Visualizer V8, you can also create a project based on pre-built sample applications and components.

In addition to creating a project from sample applications and components available on Kony Marketplace, you can create the following types of custom projects:

- **Kony Reference Architecture.** A Kony Reference Architecture project provides a framework to create a structured, modular digital application. It provides a controller to manage forms and templates as they are created. You can use RequireJS to load modules on demand. For information on creating a Kony Reference Architecture project, see Create a Kony Reference Architecture Project.
- **Free Form JavaScript.** A Free Form JavaScript project lets you create a digital application without using an application development framework. For information on creating a Free Form JavaScript project, see Create a Free Form JavaScript Project.

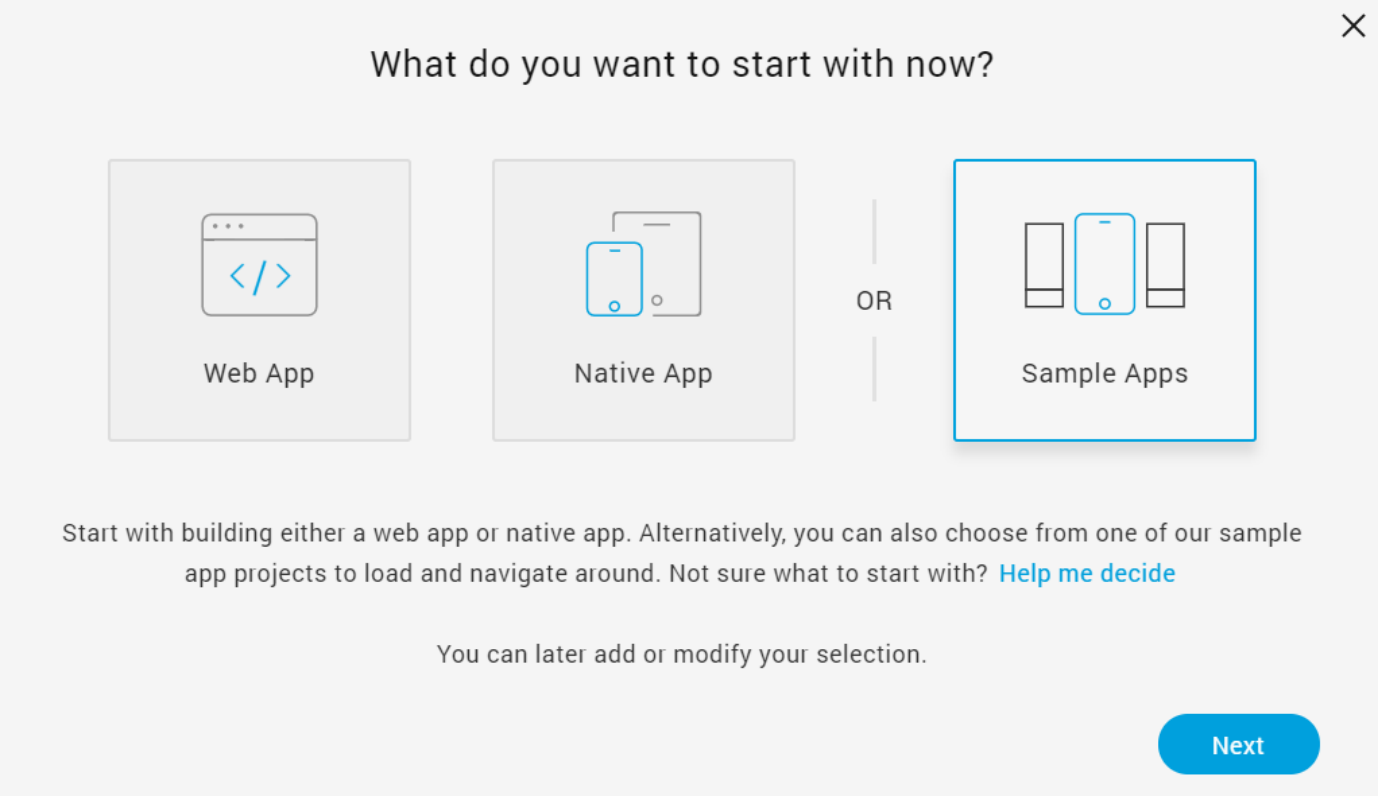
For more information on creating on project from sample applications and components, see Create a Project from Sample Applications and Components.

Create a Project from Sample Applications and Components

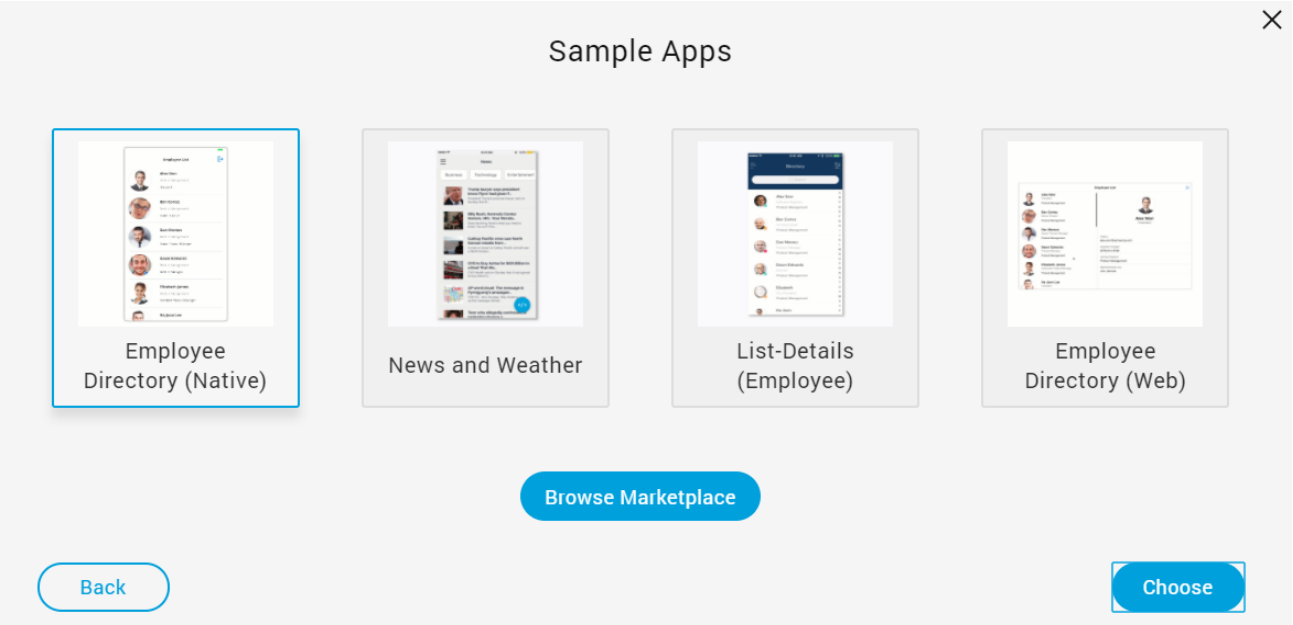
To help you easily and quickly develop your applications, Kony Visualizer provides access to a variety of sample applications and components. You can create a project based on a sample application and use that application as a template or starting point for your application, or use a rich assortment of pre-built components as building blocks for your application.

To create a project from a sample application or component, follow these steps:

1. Click the **Project** menu (**File** menu in Kony Visualizer Classic), and then click **New Project**. The **What do you want to start with now?** screen of the **New Project** wizard appears.



2. Click **Sample Apps**, and then click **Next**. The **Sample Apps** screen appears.



3. Select one of applications and components displayed on the **Sample Apps** screen, and then click **Choose**. The selected application or component is imported to your project. Alternatively, you can click **Browse Marketplace** to open the Kony Marketplace website. Kony Marketplace provides a variety of samples and components that you can add to your project.
4. Continue adding any additional components that help you build your application.

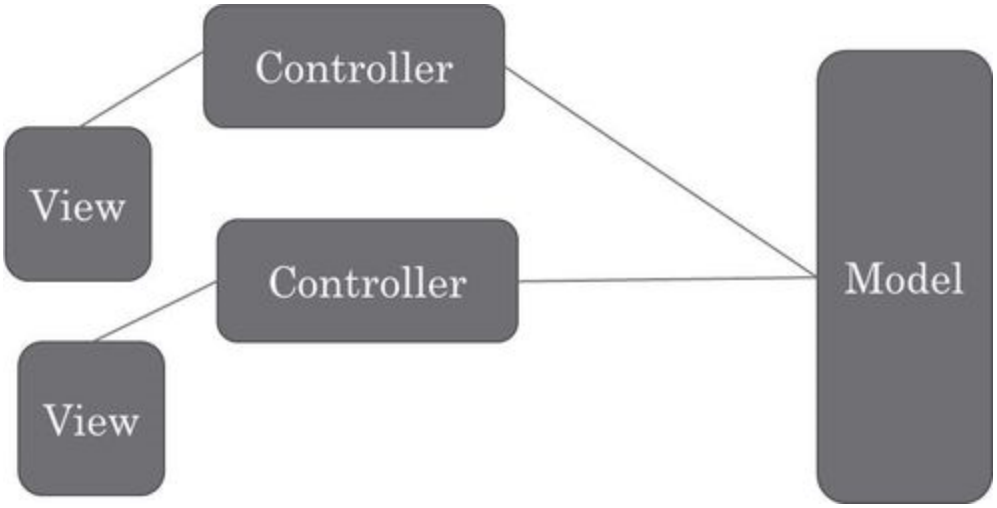
You can use components as building blocks for rapid application development without having to write all the code yourself, or create you own custom components. For more information about working with and customizing components, see Creating Applications with Components.

You can also use the **New Project** wizard to create a custom Kony Reference Architecture project or a custom Free Form JavaScript project. For information on the different types of Kony Visualizer projects, see Types of Projects.

For information about creating a custom Kony Reference Architecture project, see Create a Kony Reference Architecture Project. For information about creating a custom free form JavaScript project, see Create a Free Form JavaScript Project.

Create a Kony Reference Architecture Project

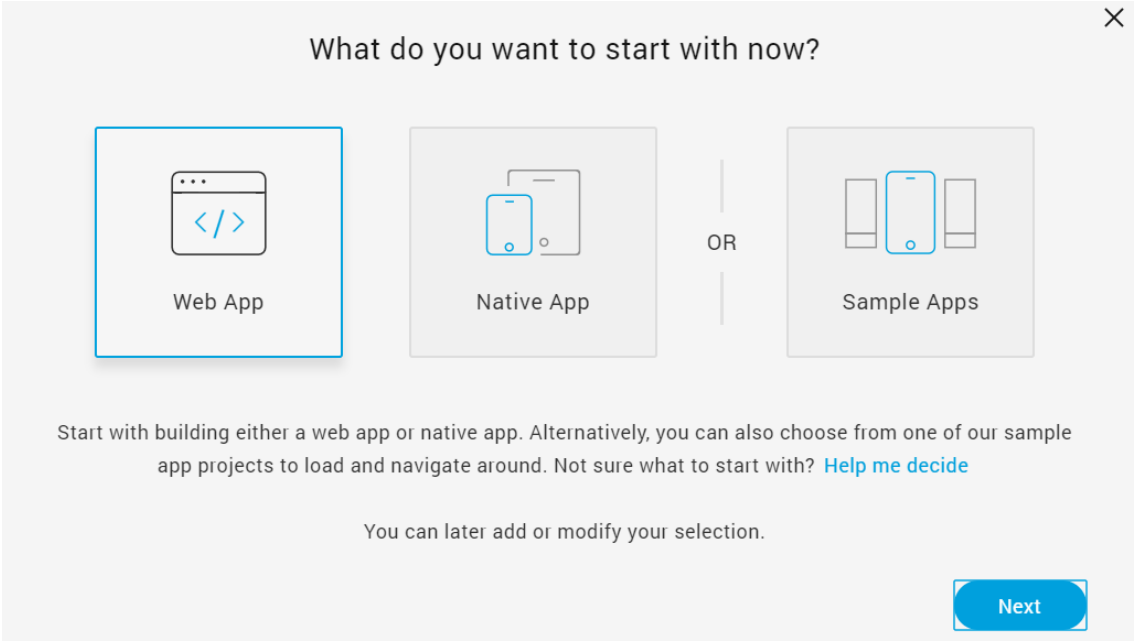
A Kony Reference Architecture project lets you create a custom application using a structured, modular framework based on the Model-View-Controller (MVC) architecture. Forms represent the *view* of the application, and there is a *controller* associated with each form that manipulates the view, handles navigation, and interacts with a statically defined *model* or dynamic object services layer.



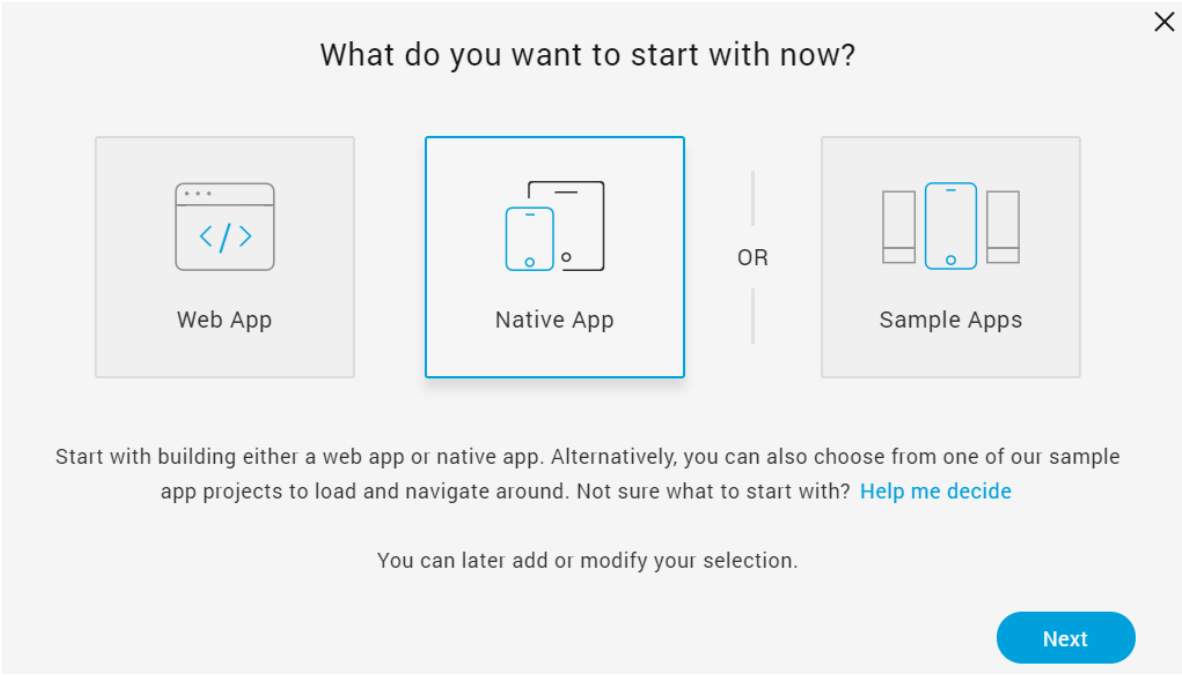
You can use the **New Project** wizard to create a custom Kony Reference Architecture project, a custom Free Form JavaScript project, or a project that is built from sample applications and components. For information on the different types of Kony Visualizer projects, see Types of Projects.

To create a new custom Kony Reference Architecture project, follow these steps:

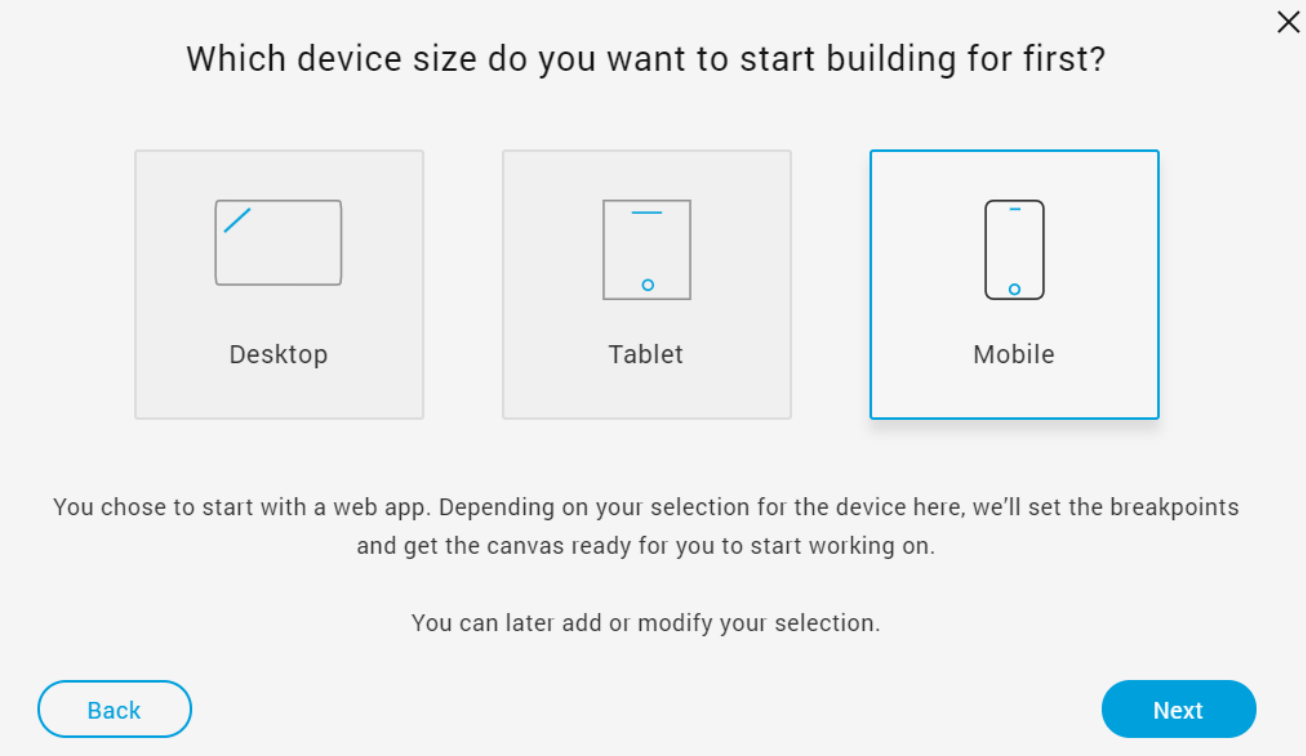
1. Click the **Project** menu (**File** menu in Kony Visualizer Classic), and then click **New Project**. The **What do you want to start with now?** screen of the **New Project** wizard appears.
2. You can choose to create any one of the following types of apps:
   - **Web App**: Click **Web App**,and then click **Next**.

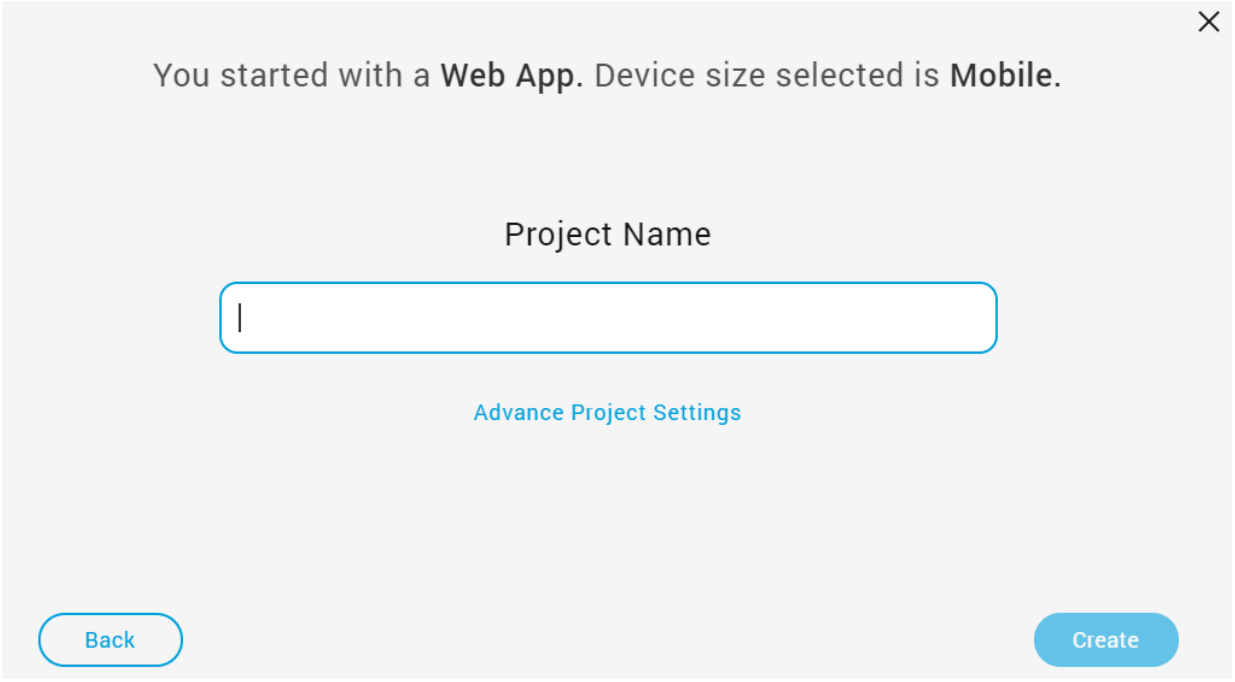


- **Native App**: Click **Native App**,and then click **Next**.

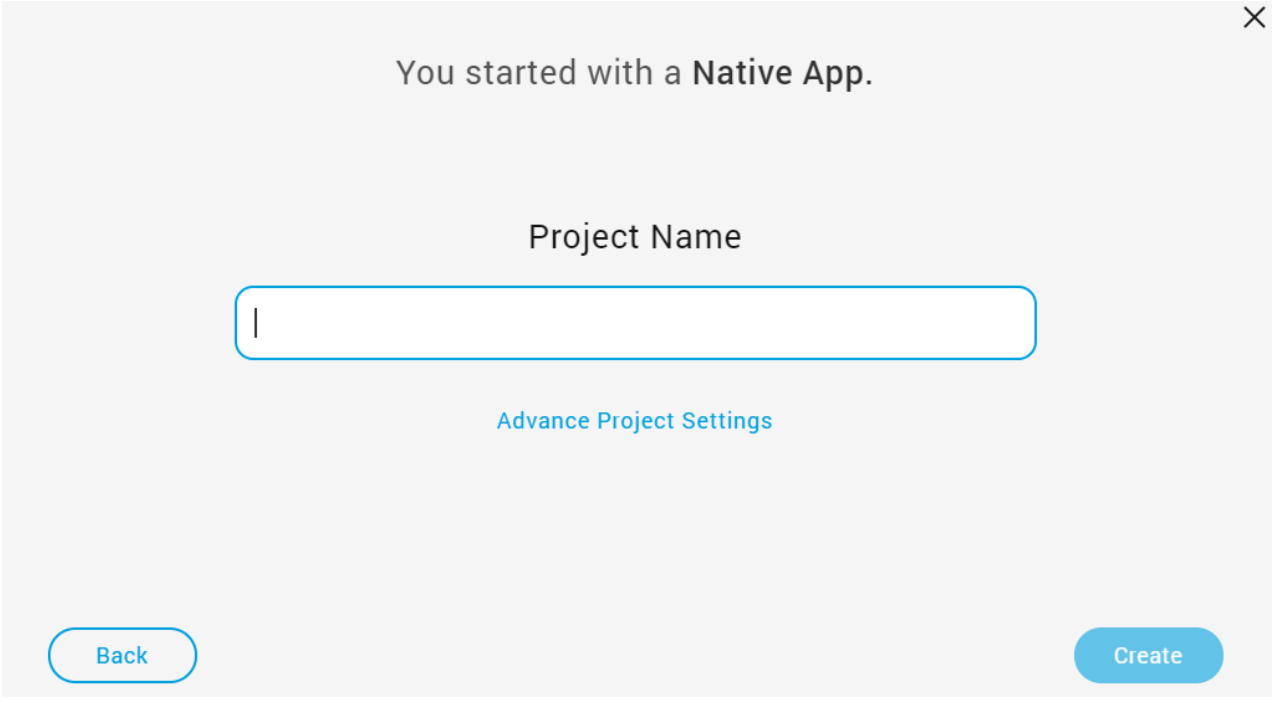


3. Perform these steps for the following types of apps:
    - **Web App**: Select the channel for which you want to create a Web app: Desktop, Tablet, or Mobile. Here, **Mobile**. Click **Advance Project Settings**.

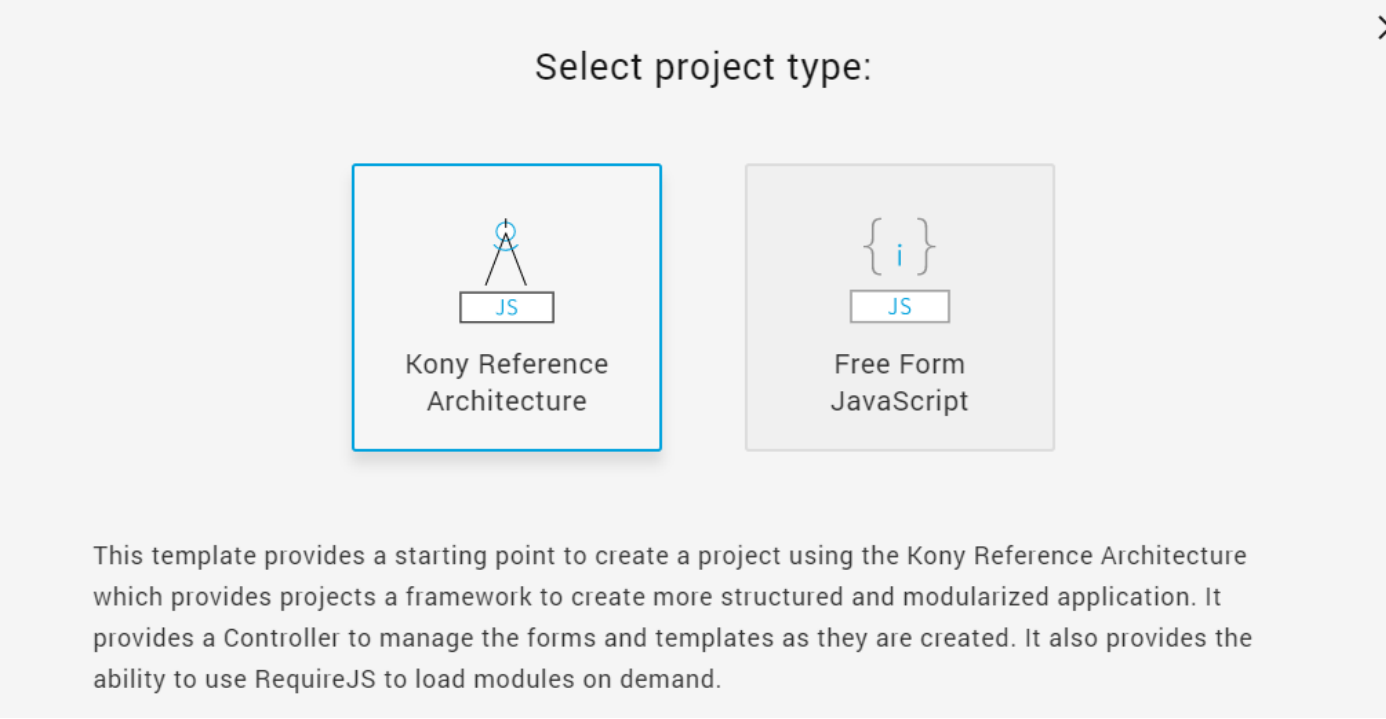




- **Native App**: Click **Advance Project Settings**.

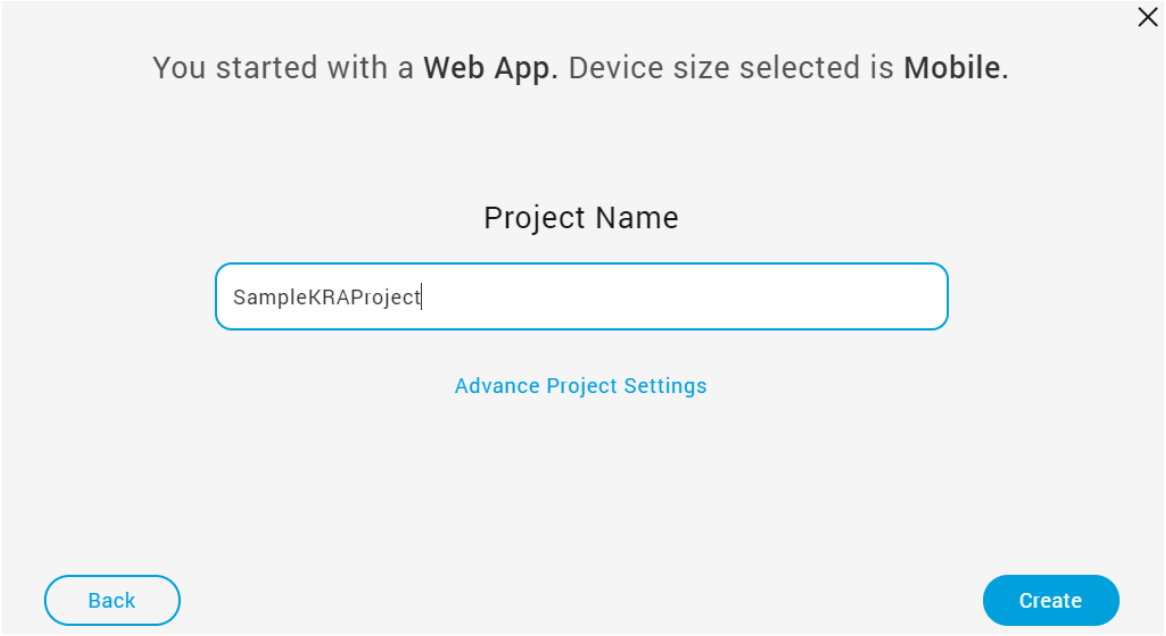


4. The **Select project type** window appears. Click the **Kony Reference Architecture** option, and then click the Close icon ×.

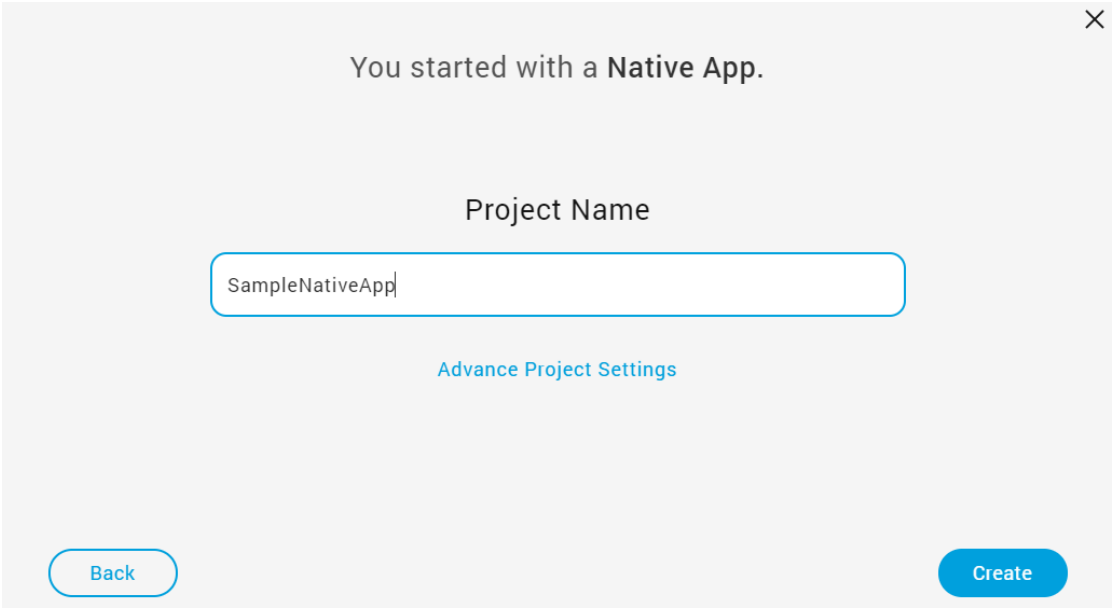


> ***Note:*** By default, the **Kony Reference Architecture** option is auto-selected in the **Select project type** window.

5. In the **Project Name** box, type the name of the project.
   - **Web App**



- Native App



Note: A project name must contain fewer than 18 characters. You cannot use any special characters or spaces in the project name.

> ***Note:*** A project name can be alphanumeric. However, the first character of a project must be an alphabet.

> ***Note:*** Do not use any of the following reserved keywords as a project name: authService, workspace, mfconsole, kpns, middleware, accounts, syncservice, syncconsole, services, admin, middleware, and appdownload.

6. Click **Create**. Kony Visualizer creates a Kony Reference Architecture project with folders for forms, popups, templates, modules, resources, actions, and services. For a Kony Reference Architecture application project, Kony Visualizer also creates a **Controllers** folder.

When you create a form or template, Kony Visualizer automatically creates associated *controller* and *controller actions* modules in the **Controllers** folder, using the form name as a prefix. When you rename the form or template, Kony Visualizer automatically updates the names of the associated modules to use the new form or template name.

For more information about the Kony Reference Architecture, see the Kony Reference Architecture SDK API Programmer's Guide

For information about creating a project from sample applications and components, see Create a Project from Sample Applications and Components. For information about creating a free form JavaScript project, see Create a Free Form JavaScript Project.

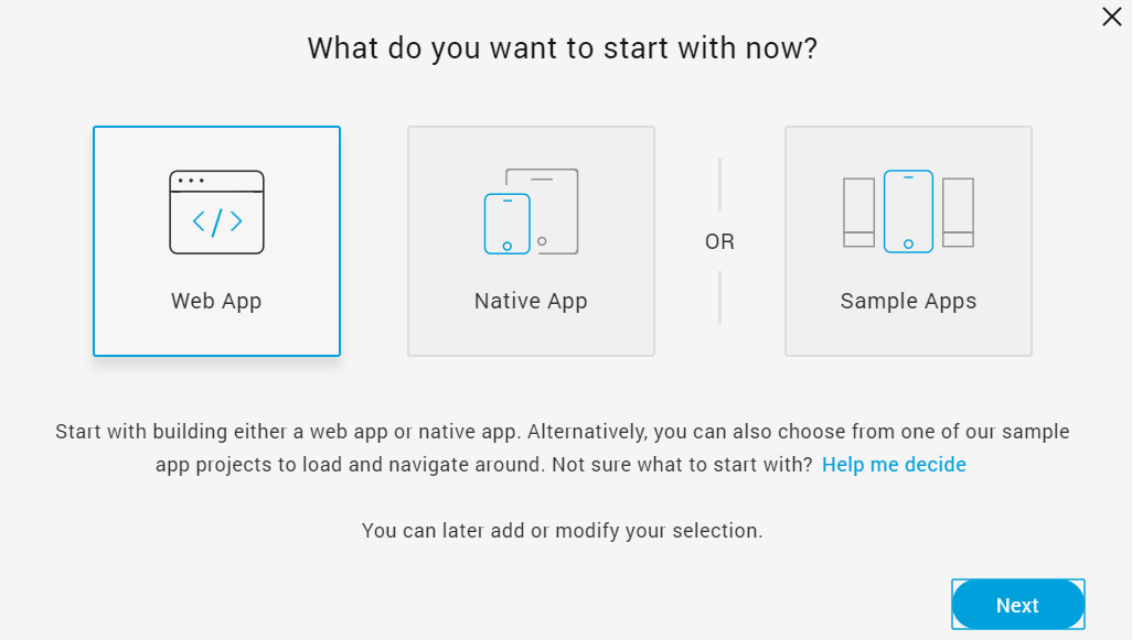
Create a Free Form JavaScript Project

A Free Form JavaScript project lets you create a custom application without using an application development framework.

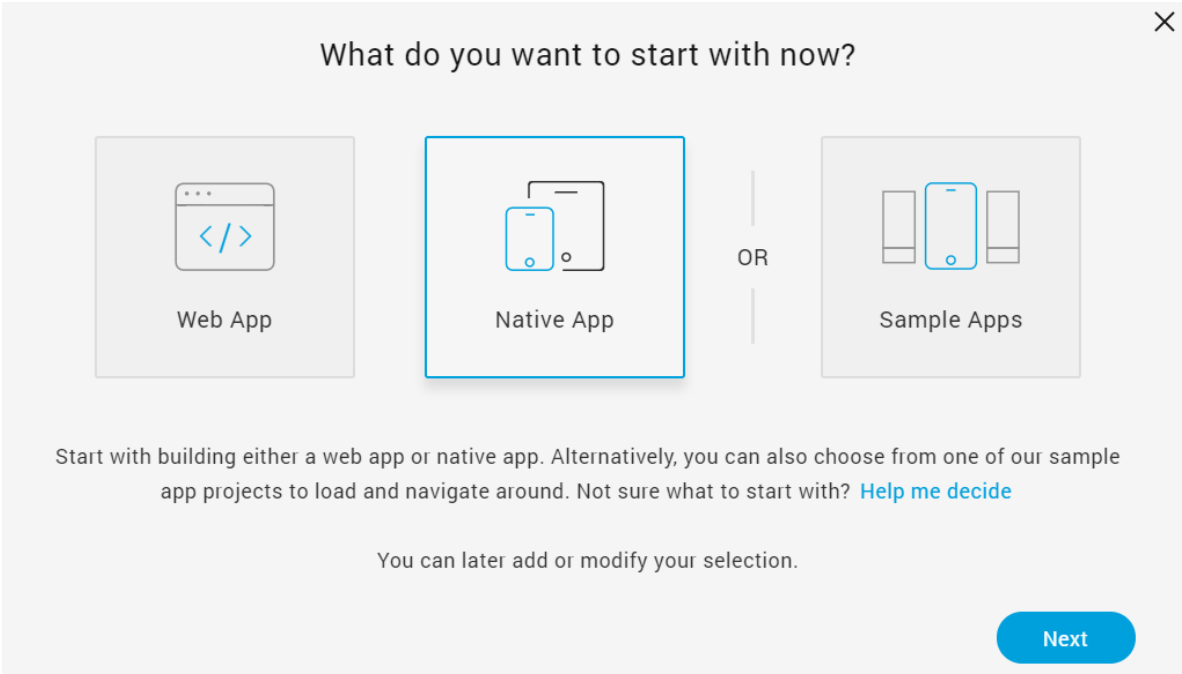
You can use the **New Project** wizard to create a custom Free Form JavaScript project, a custom Kony Reference Architecture project, or a project that is built from sample applications and components. For information on the different types of Kony Visualizer projects, see Types of Projects.

To create a new custom Free Form JavaScript project, follow these steps:

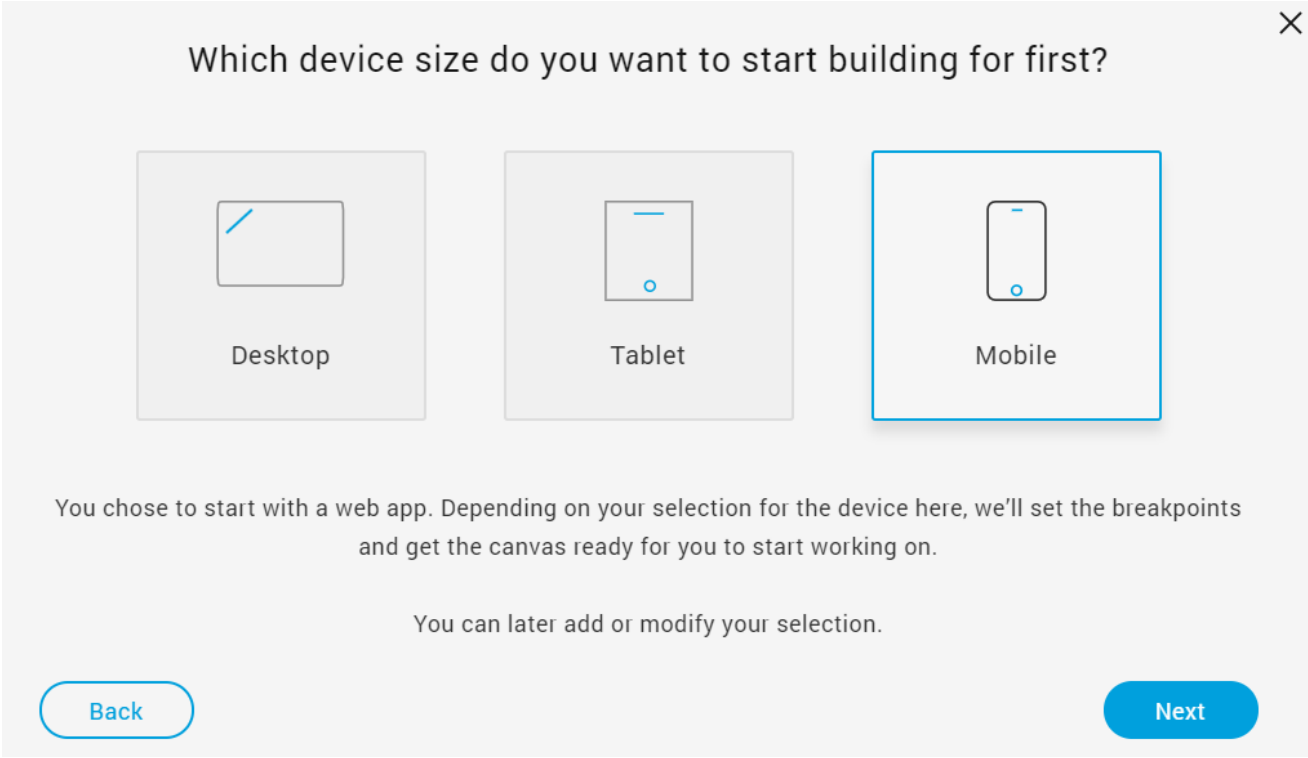
1. Click the **Project** menu (**File** menu in Kony Visualizer Classic), and then click **New Project**. The **What do you want to start with now?** screen of the **New Project** wizard appears.
2. You can choose to create any one of the following types of apps:
   - **Web App**: Click **Web App**,and then click **Next**.

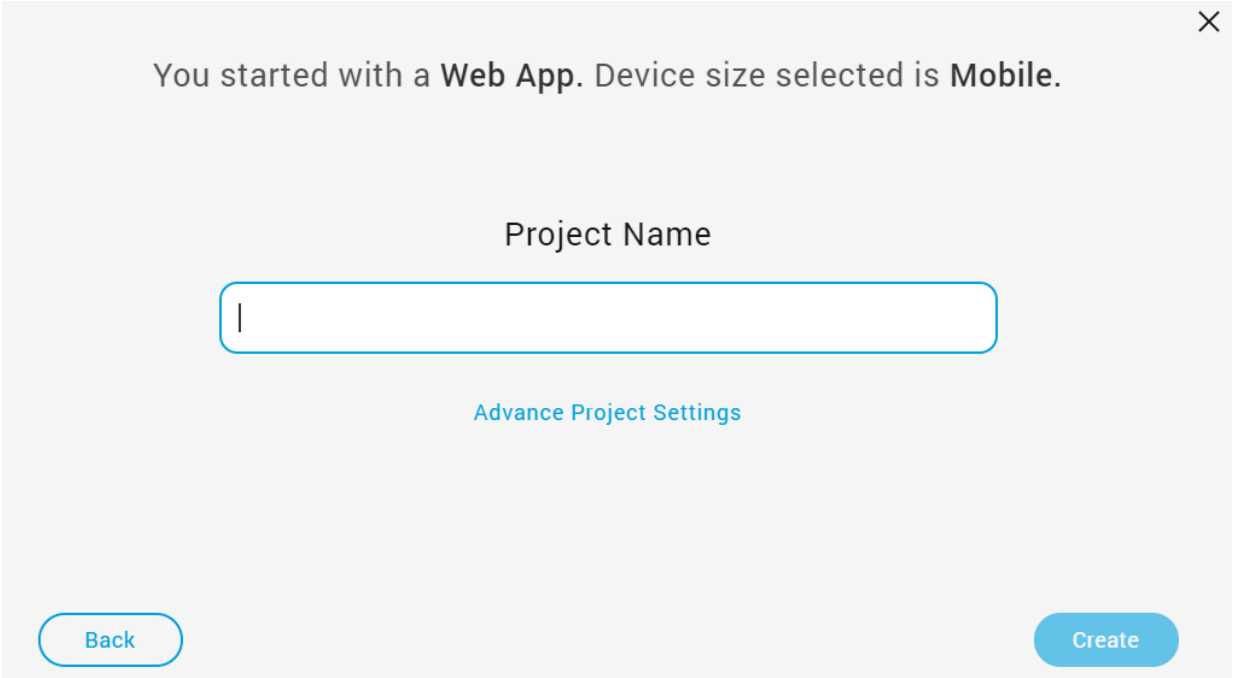


- **Native App**: Click **Native App**,and then click **Next**.

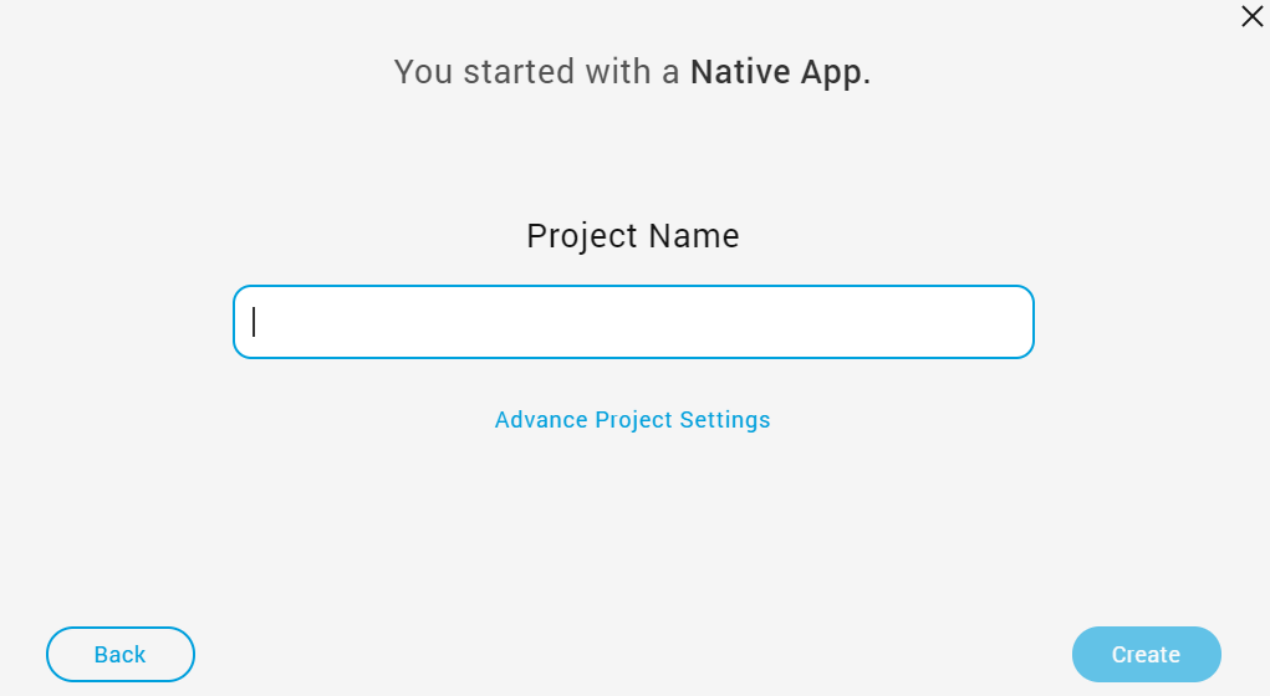


3. Perform these steps for the following types of apps:
    - **Web App**: Select the channel for which you want to create a Web app: Desktop, Tablet, or Mobile. Here, **Mobile**. Click **Advance Project Settings**.

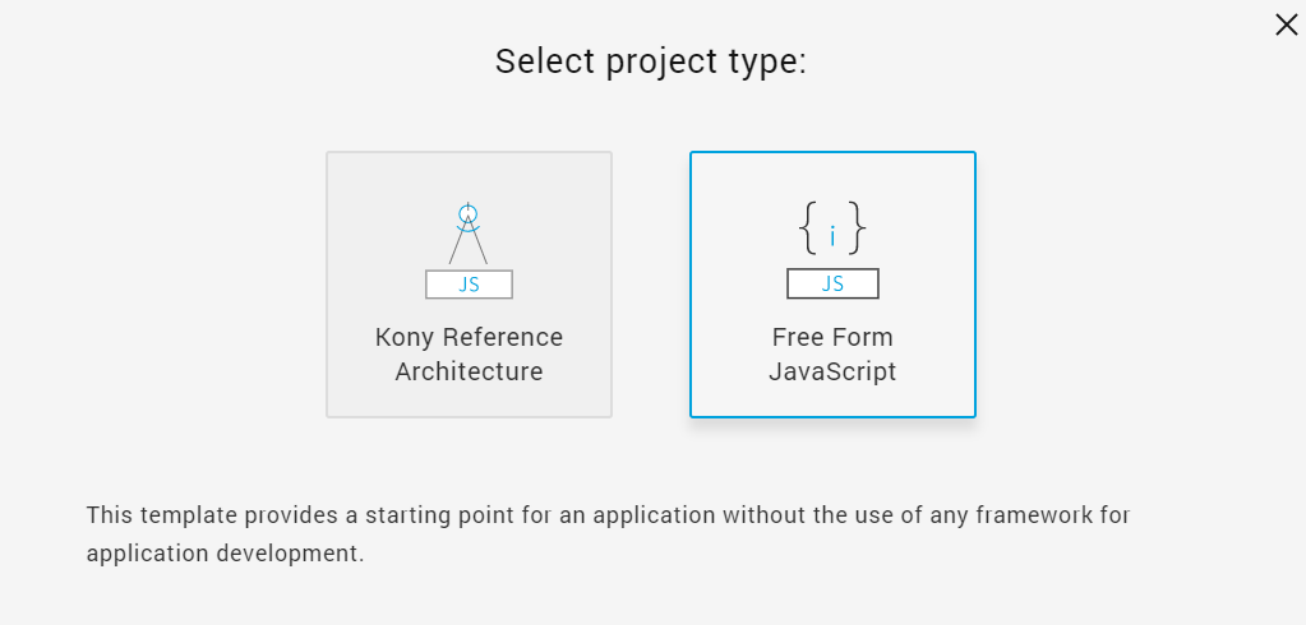




- **Native App**: Click **Advance Project Settings**.

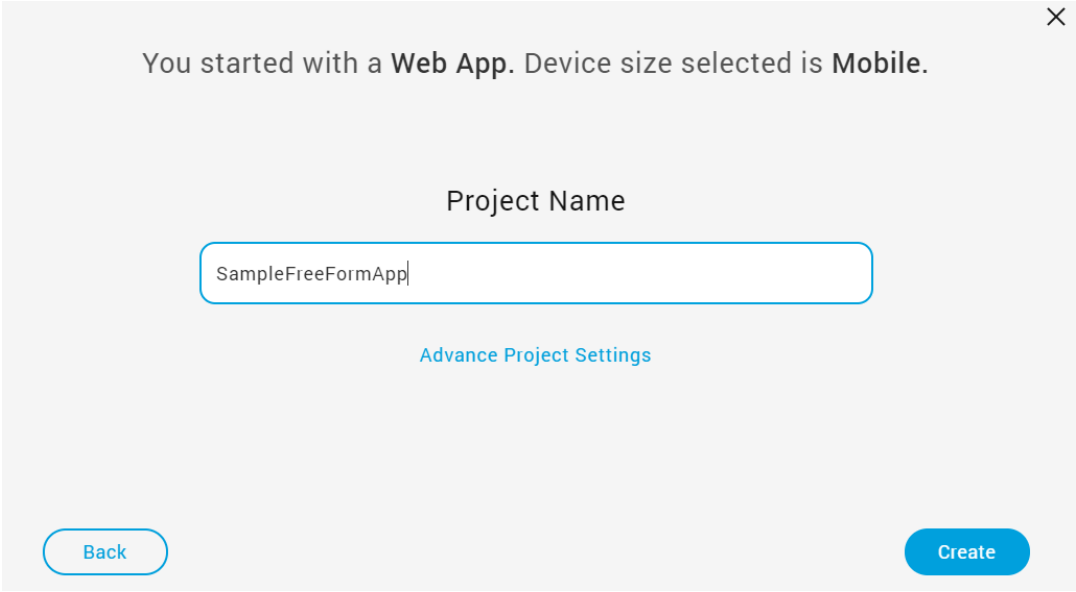


4. The **Select project type** window appears. Click the **Free Form JavaScript** option, and then click the Close icon .

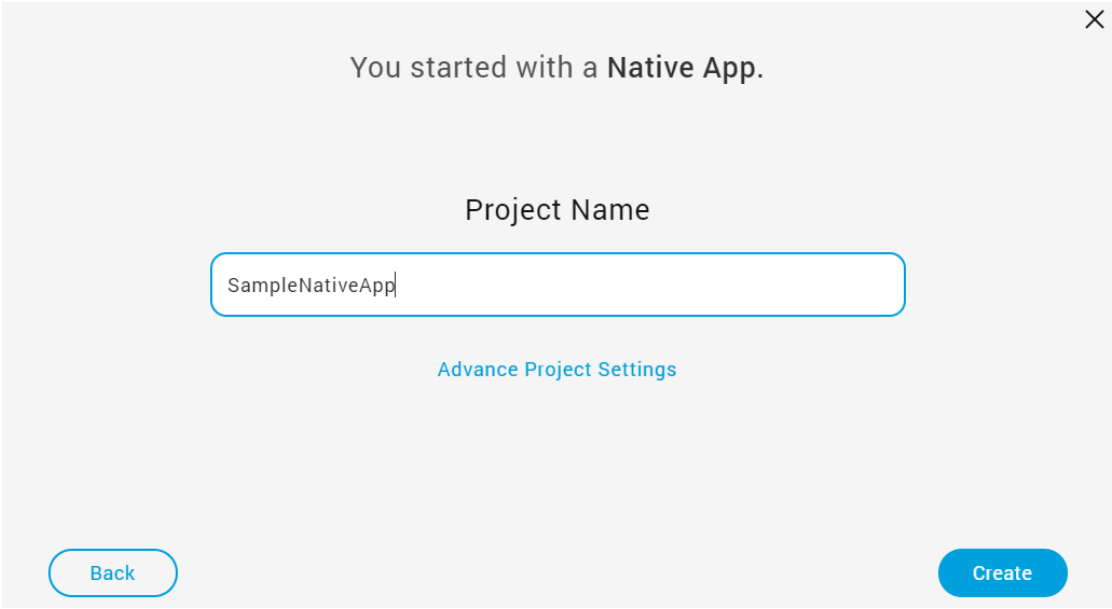


> ***Note:*** By default, the **Kony Reference Architecture** option is auto-selected in the **Select project type** window.

5. In the **Project Name** box, type the name of the project.
    - **Web App**



- Native App



Note: A project name must contain fewer than 18 characters. You cannot use any special characters or spaces in the project name.

> ***Note:*** A project name can be alphanumeric. However, the first character of a project must be an alphabet.

> ***Note:*** Do not use any of the following reserved keywords as a project name: authService, workspace, mfconsole, kpns, middleware, accounts, syncservice, syncconsole, services, admin, middleware, and appdownload.

6. Click **Create**. Kony Visualizer creates a Free Form JavaScript project with folders for forms, popups, templates, modules, resources, actions, and services.

By default, Kony Visualizer creates folders for forms, popups, templates, modules, resources, actions and services.

For information on creating a Kony Reference Architecture project, refer Create a Kony Reference Architecture Project.

For information about creating a project from sample applications and components, see Create a Project from Sample Applications and Components. For information about creating a Kony Reference Architecture project, see Create a Kony Reference Architecture Project.

Create an iOS Application Extension

An iOS application extension lets you extend custom functionality and content beyond your application, making it available to users while they are interacting with other applications or the iOS system. For example, your application could appear as a widget on the iOS Today screen, or you could add a custom interface for your application's notifications.

You create an iOS application extension in Kony Visualizer using a custom App Extension project. You can use the **New Project** wizard to create a custom iOS application extension project, a custom Kony Reference Architecture project, a custom Free Form JavaScript project, or a project that is built from sample applications and components. For information on the different types of Kony Visualizer projects, see Types of Projects.

To create a new custom iOS application extension, follow these steps:

1. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **New Project** to open the **Start a New Project** screen of the **New Project** wizard.
2. Select **Create Custom App**, and then click **Choose**. Kony Visualizer opens the **Project Type** screen of the **New Project** wizard.
3. In the **Project Name** text box, enter a name for your application extension project.
    - A project name should contain fewer than 18 characters. You cannot use any special characters or spaces in the project name.
    - A project name can be alphanumeric. However, the first character of a project must be an alphabet.
    - Do not use any of the following reserved keywords as a project name: authService, workspace, mfconsole, kpns, middleware, accounts, syncservice, syncconsole, services, admin, middleware, and appdownload.
4. Select **App Extension**, and then click **Next** to open the **Choose App Extension Type** dialog box.

    You can create any of the following types of extensions:

    - **Action Extension**. An Action extension manipulates or views content originating in a host application. For example, an Action extension might let a user edit an image in a document viewed in a text editor, or view a selected item in a different format or language.

- **iMessage**. An iMessage extension interacts with the Messages application. For example, an iMessage extension could let users create and share content, add stickers, or make payments without leaving their conversations.
- **Intents**. An Intents extension handles tasks related to supporting Siri integration with your application. Siri and Maps interact with your application through the Intents extension. The Intents extension directs Siri to the objects capable of responding to user requests.
- **Intents UI**. An Intents UI extension customizes the Siri or Maps interface after handling a task related to supporting Siri integration with your application.
- **Notification Content**. A Notification Content extension handles tasks that your application can perform in response to a delivered notification.
- **Share Extension**. A Share extension posts to a sharing web site, or shares content with others. A Share extension gives users a convenient way to share content with other entities, such as social sharing websites or upload services. For example, in an application that includes a Share button, a user could choose a Share extension that represents a social sharing website and then use it to post a comment or other content.
- **Today Extension**. A Today extension gets a quick update or performs a quick task in the Today view of Notification Center. For example, a user could check current stock prices or weather conditions, see the current day's schedule, or mark a current task item as done.

5. Select an extension type, and then click **Add** to open the **Configure App Extension** dialog box.
6. Enter a display name, and then click **Finish**.

By default, Kony Visualizer creates folders for modules and resources, and an *Info.plist* information property list file. You can use the code editor to make changes to the plist xml.

If necessary, you can import or add a native functions module to the project by selecting **Manage Native Functions** from the **Edit** menu.

For information on the iOS application extension programming interface, see App Extension API for iOS.

For information about creating a project from sample applications and components, see Create a Project from Sample Applications and Components. For information about creating a custom Kony Reference Architecture project, see Create a Kony Reference Architecture Project. For information about creating a custom free form JavaScript project, see Create a Free Form JavaScript Project.

Migrate a Project from an Earlier Version of Kony Studio or Visualizer

You can import projects into Kony Visualizer that were originally created using Kony Studio and earlier versions of Kony Visualizer.

If your project was created using a version of Kony Studio older than 6.0, you first must import it into Kony Studio 6.0 (with the latest hotfix) and then you can import it into Kony Visualizer.

> ***Important:***
>
> - If your project was originally created using Kony Visualizer 2.5 that was then imported into Kony Studio 6.5, and you import it into the latest version of Kony Visualizer, the action sequences are not imported.
> - If yours is a Kony Visualizer 2.5 or earlier project, importing it updates it so that it can take advantage of all the features of Kony Visualizer, but your earlier version of Kony Visualizer you will no longer be able to open it.

To migrate your application to Kony Visualizer, do the following:

1. If your application is built in Kony Studio versions prior to 6.0, import your application to Kony Studio 6.0 GA version (with the latest hotfix applied).
   It is recommended that you take a backup of your project before importing it into Kony Visualizer.
2. After importing to Kony Studio 6.0 GA, do the following:
   a. Right-click on the project and select **Utilities**.
   b. From the options available, select **Upgrade the project to the current version**.
   c. A confirmation message appears. Click **OK**.
3. Once the project is upgrade, export your application.
4. Import the application to Kony Visualizer. To do so, on the **File** menu (the **Project** menu in *Kony Visualizer*), click **Import**.
5. Do one of the following:
   - If your project is located in a standard project folder structure (i.e. has not been zipped up as an archive (.zip) file), click the **Browse** button for **Select project root**, navigate to the root folder of the project you want to import, and then click **OK**.
   - If you are importing an archive (.zip) file, click **Select archive file**, click its **Browse** button, navigate to the root folder of the project you want to import, and then click **OK**.
6. In the case of a Kony Visualizer project, to convert designer actions to developer actions, select **Copy 'Designer Actions' to 'Developer Actions'**. Be aware that in doing so, any developer actions that the project contains are overwritten.
7. In the case of a Kony Studio project, to import its services to Kony Fabric, select **Import services into Kony Fabric**. In Kony Visualizer, all services are managed through Kony Fabric. Importing them adds them to the Kony Fabric workspace.

> ***Important:*** If the app that you are migrating from Kony Studio or an earlier version of Kony Visualizer was configured with custom connectors, you need to migrate them to Kony Fabric as a consolidated service definition (CSD). For more information, see Migrate a Consolidated Service Definition to Kony Fabric in the *Kony Fabric Console User Guide*.

8. Click **Finish**. Kony Visualizer imports the project.

   For projects imported from Kony Studio, all deprecated widgets found in the upgrade process are deleted from the application.

   - Duplicate widgets are renamed automatically.
   - Duplicate skins are renamed automatically.
   - Reserved keywords are renamed automatically.
   - You can find all changes in a log file located in the project root folder. For example, *<ApplicationName>_upgradeLog.log*. If there were no changes in the application, the log file will not be created.
   - You must take necessary action on any of the issues in the warning section in the log file.

   > ***Important:***
   > - Ensure that you import the splash image when importing projects from Kony Studio. Importing the splash image will fix any inconsistencies related to Flex and VBox properties.
   > - Refresh of model files will take place if **Import service to Kony Fabric** is selected while migrating the app. If **Import service to Kony Fabric** is cleared at time of migration and later the import to Kony Fabric is performed then the refresh of sync client code will not happen automatically. In this case, right-click **Kony Fabric** in Project Explorer and select **Refresh Sync Client Code** option.

9. In the Migration Report dialog box, click **Save**. Locate and open the migration report, open the project in Kony Studio and, using the migration report to guide you, fix the issues. After all the

issues are fixed, export the application from Kony Studio. Then in Kony Visualizer, attempt to import the project again.

Resolve conflicts between Developer and Designer Actions

When a project is imported from Kony Visualizer Classic to Kony Visualizer V9, the actions in the project are also imported and appear in the Action editor.

If there are any conflicts in the actions due to the import, a warning symbol appears in the Action tab of the Properties panel. You can fix the conflicts by using the Resolve option in the Action tab. Conflicts arise if there are both Designer and Developer actions associated with the same action event.

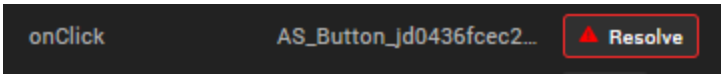
Designer actions are created in Kony Visualizer. Developer actions are created in Kony Visualizer Classic.

Note:

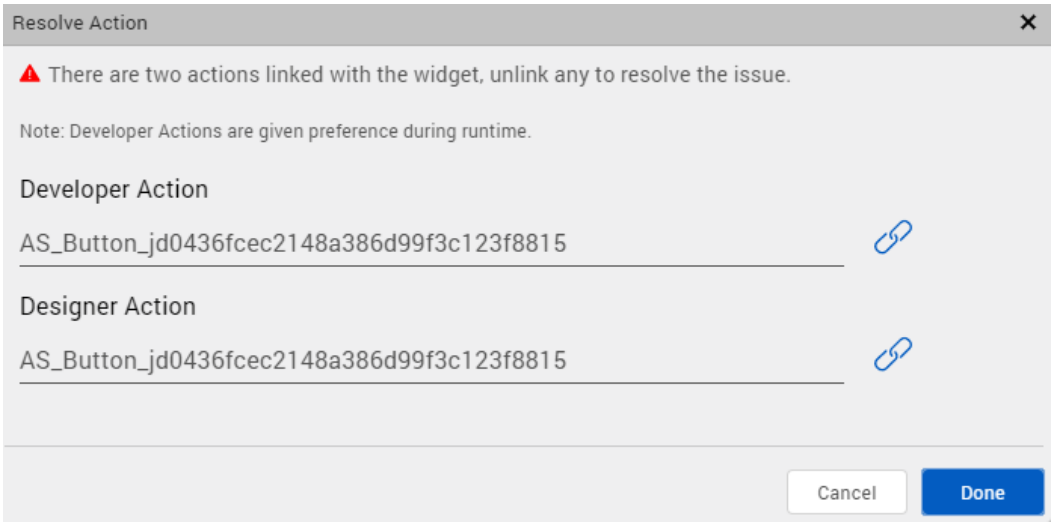
- If there are no Developer Actions, the Resolve Action pop up displays the Designer Action associated with the widget or form. This Designer Action can either be deleted or converted to a Developer action by clicking **Activate**.
- If the imported project contains only Designer Actions, all the actions are converted into Developer Actions.

To resolve conflicts between Designer and Developer actions, follow these steps:

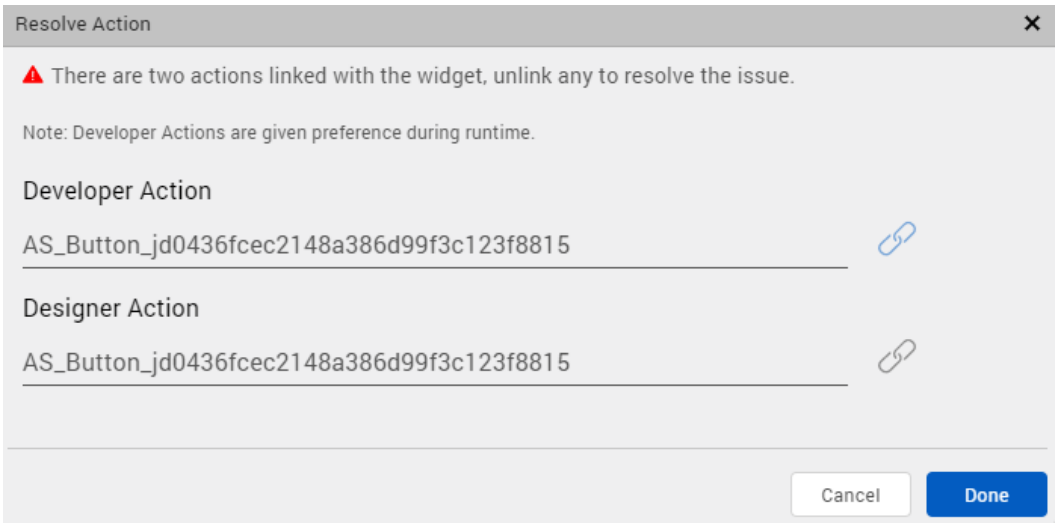
1. From the **Properties** panel of Kony Visualizer, navigate to the **Action** tab of a form or widget. For actions that are in a conflicted state, a Resolve button appears next to the action.



2. Click **Resolve**.
   Details of the Developer and Designer actions appear.



3. Click the **Unlink** icon against the action that you do not want to associate with the widget or form, and then click **Done**.



The action that you choose to retain is associated with the widget and is considered as a Developer action.

> ***Important:***
> - When you build the project, only Developer actions are associated with the widgets.

In case of conflicts, the Developer actions take precedence, and Designer actions are not considered.

- When you export an entity of a project that contains conflicting actions from Kony Visualizer, the Designer actions are not considered. The project is exported with only the Developer actions associated.

Integration with Design Tools

With the Kony Visualizer plugins, you can convert your UI designs built on third party design tools into Visualizer projects. Projects that will be converted into forms, widgets, and image assets that you can import and open in Kony Visualizer. So even if you don't begin your app design in Kony Visualizer and you're using a different application as a design environment, you can seamlessly continue the designing of your app in Kony Visualizer, picking up where you left off.

The following topics describe how to migrate a project from different applications:

Integration with Photoshop

Integration with Sketch

Integration with Adobe Photoshop

With the Kony Visualizer extension for Adobe Photoshop, you can convert your Photoshop design into forms, widgets, and image assets that you can then open in Kony Visualizer. So even if you don't begin your app design in Kony Visualizer, if you're using Photoshop as a design environment, you can seamlessly continue the designing of your app in Kony Visualizer, picking up where you left off in Photoshop.

The Kony Visualizer plugin for Photoshop is compatible with Adobe Photoshop CC and above. Please note that in converting Photoshop layers to Kony Visualizer widgets, the extension has the following limitations.

Importing a Photoshop design into Kony Visualizer involves the following tasks:

- Install the Kony Visualizer Extension for Photoshop
- Export a Photoshop Design to Kony Visualizer
- Open an Exported Photoshop Project in Kony Visualizer

Click here to watch a video on importing a Project from Adobe Photoshop to Kony Visualizer.

Install the Kony Visualizer Extension for Photoshop

To install the Kony Visualizer extension for Photoshop, do the following:

1. Depending on your development environment, download from the Kony web site one of the following stand-alone Photoshop extension installers for Kony Visualizer:

   Kony Visualizer Photoshop extension for the Mac

   Kony Visualizer Photoshop extension for Windows

2. Install the extension. The installer leads you through the process of installing the extension to the correct location on your computer so that it is available in Photoshop.

Export a Photoshop Design to Kony Visualizer

Using the Kony Visualizer extension for Photoshop, you export your Photoshop project as a Kony Visualizer app that you can then open in Kony Visualizer. In preparing a project for export, you set the global, default export options that you want, set the export options you want for individual layers (overriding the global settings for that particular layer), and indicate the **channel**[1] you want the project to be exported for.

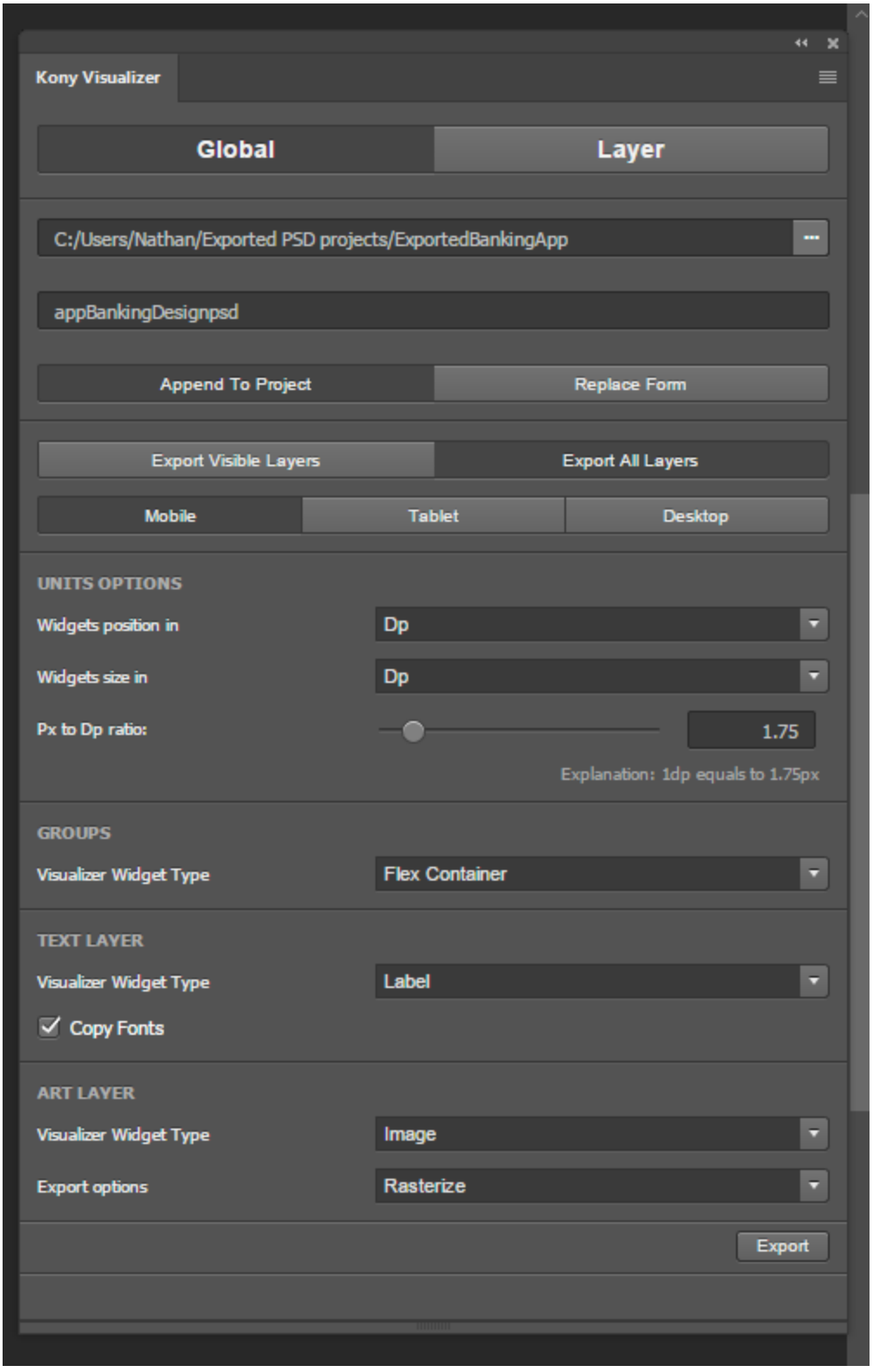
This procedure assumes that you have already installed the Kony Visualizer Photoshop extension. For more information, see Install the Kony Visualizer Extension for Photoshop.

To export a Photoshop project to Kony Visualizer, do the following:

---

[1]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

1. In Adobe Photoshop, open the project you want to export to Kony Visualizer.
2. If it is not already visible, display the Kony Visualizer extension by clicking the **Window** menu, next selecting **Extensions**, and then clicking **Kony Visualizer**.
3. From the Kony Visualizer extension, click **Global**. The extension looks like this.



4. Specify the name of your exported project by setting its path. The folder you create or select becomes the exported project's Kony Visualizer project name. To set the path, click the ellipsis button (...) of the Project Name text box, and then navigate to the path where you want to save the exported project to, creating a new folder, if necessary. When you have finished, click **OK**.

5. If you are saving the project to a folder that you have already exported to, indicate whether you want to append the project to the existing exported content as an additional form (or forms), or overwrite and replace the existing exported content. To do so, click either **Append to Project** or **Replace Form**. By default, the Kony Visualizer extension appends to any existing projects.

6. Select whether you want to export only the visible layers (as set on the Photoshop Layers tab), or export all layers regardless of whether they're visible or not.

7. Indicate the type of **channel**[1] that you want to export the project to by selecting either **Mobile**, **Tablet**, or **Desktop**. The default value is **Mobile**.

8. Set the units options for the size and position of elements. You can select from the following:

    - **Px.** Uses pixels as the unit for the dimensions and positioning of project elements for Kony Visualizer. Pixels are a fixed, physical unit and do not scale proportionately depending on the resolution of a devices screen. The rendering of an app on a low-resolution device will appear larger and more pixelated than it would on a higher-resolution device.

    - **Dp.** Uses density-independent pixels, which is based upon the physical density of a device's screen. Ths allows the dimensions of the elements of your design to scale in size depending upon the screen resolution of your target device. Because the Dp unit is relative to a 160 dpi (dots per inch) screen, one dp is equal to one pixel on a 160 dpi screen. The ratio of dp to pixels varies with the screen density of the device. To accommodate this, you can set the dp-to-pixel ratio in the Kony Visualizer extension, which is described in the next step.

[1]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

- **%.** Sizes the various elements of the project as a percentage of the parent container of the project. The horizontal value and vertical value of the parent container are designated as 100% for their respective dimensions.

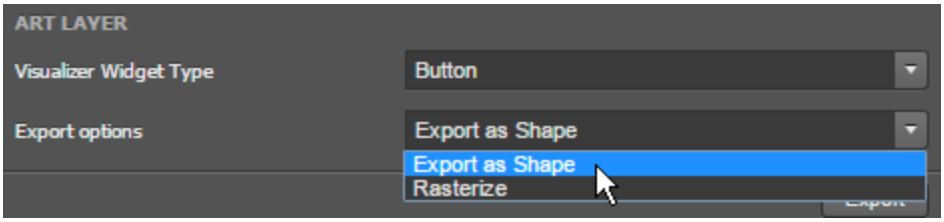
9. Set the ratio between pixels (Px) and density-independent pixels (Dp). This ratio is used only if you are using Dp as your unit of choice for element sizes or positioning. This ratio varies from device to device, depending on the physical density of pixels on the device's screen. Increasing the value of the ratio increases the number of pixels that equal one Dp. For instance, a value of 2.25 indicates that 2.25 pixels equals 1 Dp. To determine the ratio that you should use for your device, use the following formula:
   Px = DP*(The device's dpi/160)

10. Set the kind of widget you want the Photoshop elements to be converted to on a default basis for Layer Groups, Text layers, and all other layers (i.e. Art layers). To simplify the conversion process from a Photoshop project to a Kony Visualizer project, you can globally determine the default type of widget that a given type of layer gets converted to. However, if you want to override the global value for a specific layer, you can do so (as described in step 12). The Kony Visualizer extension categorizes the layers available in Photoshop into three types of layers: Layer Groups, Text layers, and all other layers (i.e. Art layers). The following table illustrates the global values that you can set for each of these three types of layers.

Photoshop Layer	Kony Visualizer Widget Options
Layer Group. Defined as any layer that has layers, project elements, or other assets nested within it.	Flex Container Flex Scroll Container
Text Layer. Any layer created using a Photoshop Type tool.	Label (default) Button Text Area Text Box
Art Layer. Defined as any layer other than a Layer Group or Text Layer. An art layer may be a shape layer, or may consist of rasterized pixels.	Image (default) Flex Container Flex Scroll Container Button

11. Select the global export option for Art layers. Depending on the Kony Visualizer widget type that you selected for Art layers, you can have such elements converted in either of the following two ways:

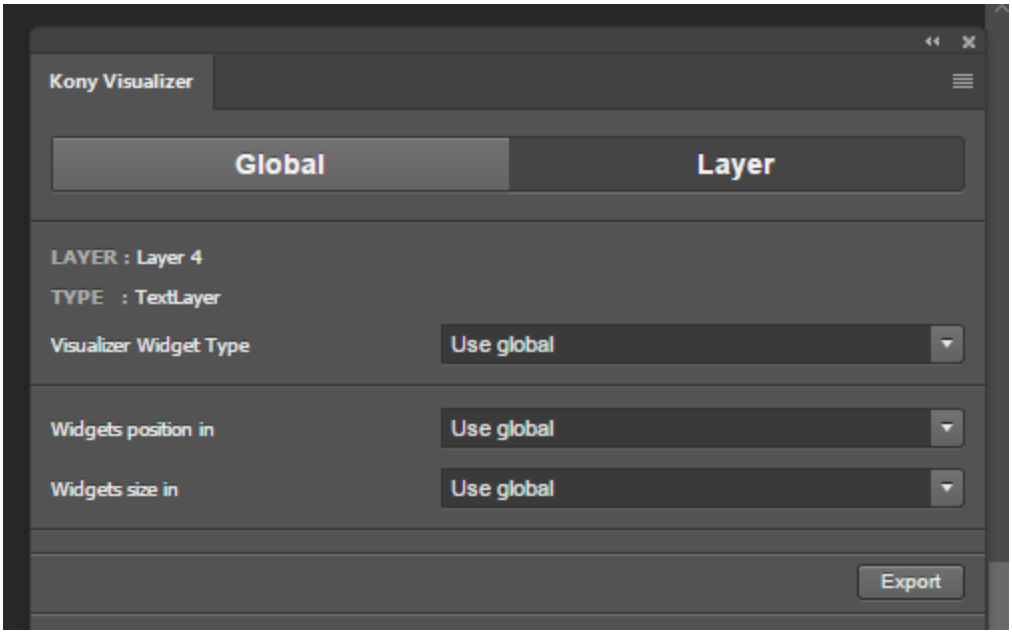
    - **Export as Shape.** The Art layers are exported as the widget type selected in the Visualizer Widget Type drop-down list, and the layer styles are converted to the equivalent styles supported under Kony Visualizer skin properties. This option is not available if the Visualizer Widget Type you select is **Image**.

    - **Rasterize.** The layer is rasterized (flattened to pixels) and cropped, and saved as a .png file. If the Visualizer Widget Type you select is either a Flex Container, a Scroll Flex Container, or a Button, the resulting converted widget uses the generated .png file as an image background in Kony Visualizer.

12. Set export options for individual layers. For any of the layers you're exporting, you can override some of the global export options. This is especially helpful if, for example, you want to convert a particular layer into a Flex Scroll Container instead of a Flex Container, or to size a given layer as a percentage of the parent container rather than using whatever unit you selected as your global export option.

    To set export options for individual layers, at the top of the Kony Visualizer extension, click **Layer**. Next, from the Photoshop Layer tab, select the layer you want to custom configure, and then select the export options you want for that layer for the Visualizer Widget Type, and the type of units you want to use for the converted widget's positioning and sizing. Select the next layer you want to customize, and repeat.



13. Once you have set all the global and layer-specific export options for your Photoshop project, click **Export**.

14. Once the exportation is finished, navigate on your computer to the folder where you exported the Photoshop project, copy the folder, and then paste it into the desired Kony Visualizer workspace folder. The default workspace folder for Kony Visualizer is:
    `C:\workspace`

Open an Exported Photoshop Project in Kony Visualizer

Once you have exported a Photoshop project using the Kony Visualizer extension and have copied it to your Kony Visualizer workspace, you can open it in Kony Visualizer.

To open an exported Photoshop project in Kony Visualizer, do the following:

1. If Kony Visualizer is already open, on the **File** menu (the **Project** menu in *Kony Visualizer*), click **Refresh**. Otherwise, launch Kony Visualizer.

2. Do one of the following:

- On the Kony Visualizer launch screen, if Kony Visualizer is pointing to the workspace where you pasted the project, Kony Visualizer lists your exported Photoshop project among recent projects. Click it.
- On the **File** menu (the **Project** menu in *Kony Visualizer*), point to **Open**, and then select from the list of available projects the project you exported from Photoshop.

3. On the Project Explorer, click the **Project** tab, open the **channel**[1] that you converted the Photoshop project to, open **Forms**, and then double-click any of the forms listed.

Limitations

The following Photoshop layer styles have no equivalent Kony Visualizer skin property and are ignored during the export process:

- Bevel and Emboss
- Satin
- Inner Glow
- Pattern Overlay

Best Practices

To optimize the conversion of Photoshop elements into Kony Visualizer elements, we recommend that you create your Photoshop project using the following best practices.

- Always use vectors / shape layers instead of pixels, since these will translate directly into Kony Visualizer widgets.
- Instead of pixels, use layer styles for gradients, color overlays, shadows, and borders, since these will all translate directly to Kony Visualizer widgets and skins.

[1]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

- Only export images as needed, and avoid exporting .png files for anything other than icons or actual images and photos.
- Do not rasterize text layers, since these can be exported as labels, text boxes, rich text, or button widgets directly into Kony Visualizer.

Integration with Sketch

With the Kony Visualizer plugin for Sketch app, you can convert your Sketch design into forms, widgets, and image assets that you can then open in Kony Visualizer. So even if you don't begin your app design in Kony Visualizer, if you're using Sketch as a design environment, you can seamlessly continue the designing of your app in Kony Visualizer, picking up where you left off in Sketch.

The Kony Visualizer plugin for Sketch is compatible with Sketch Version 60 and later. This plugin is supported in Kony Visualizer version V8 SP4 and later.

Importing a Sketch design into Kony Visualizer involves the following tasks:

- Install the Kony Visualizer Plugin for Sketch
- Export a Sketch Design to Kony Visualizer
- Open an Exported Sketch Project in Kony Visualizer

Click here to watch a video on importing a Project from Sketch to Kony Visualizer.

Install the Kony Visualizer Plugin for Sketch

Applicable for *Kony Visualizer on Mac devices.*

To install the Kony Visualizer plugin for Sketch, do the following:

1. Download the Sketch plugin installer for Kony Visualizer from the Kony community site.
   The installer is downloaded as a zip file to your computer.
2. Extract the contents of the zip file.
3. Double-click the **konyvisualizer.sketchplugin** app plugin.
   The app is installed to the correct location on your computer so that it is available in Sketch.

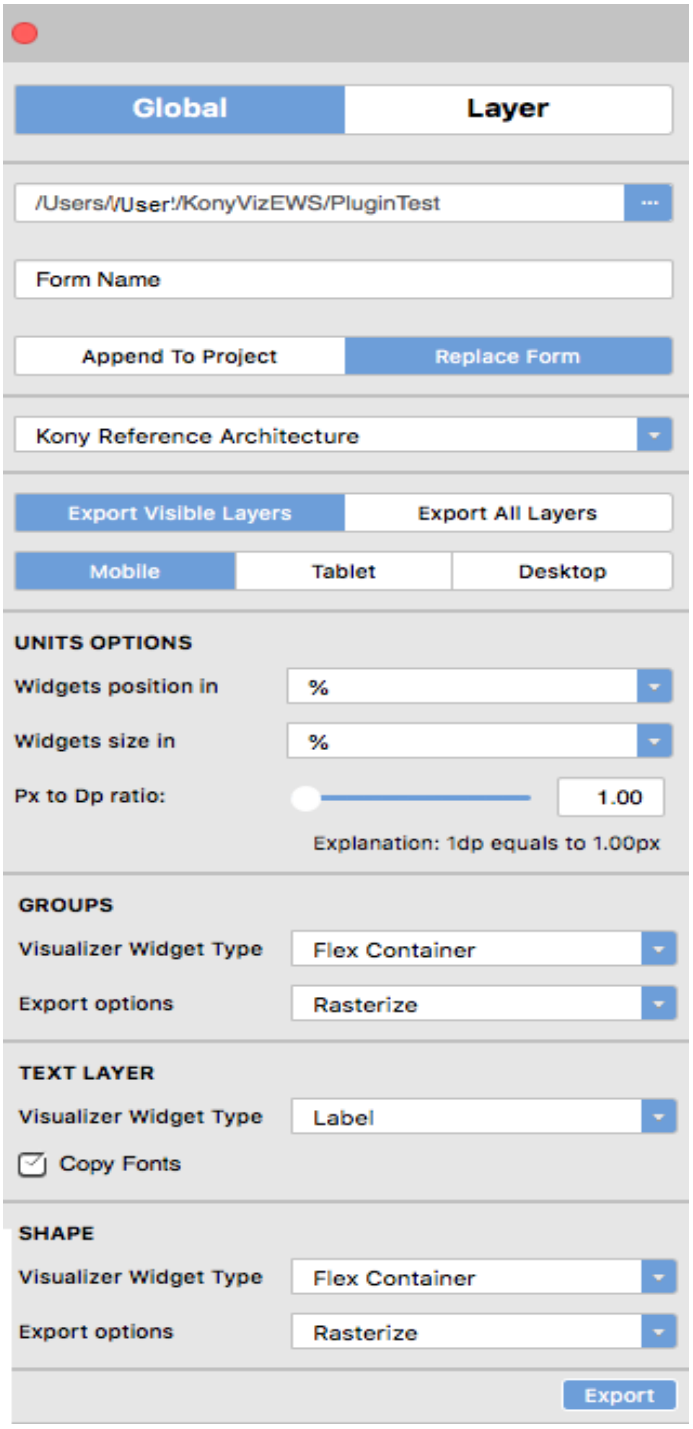
Export a Sketch Design to Kony Visualizer

Using the Kony Visualizer plugin for Sketch , you export your Sketch project as a Kony Visualizer Project that you can then open in Kony Visualizer. In preparing a project for export, you set the global, default export options, set the export options you want for individual layers (overriding the global settings for that particular layer), and indicate the channel you want the project to be exported for.

This procedure assumes that you have already installed the Kony Visualizer Sketch Plugin. For more information, see Install the Kony Visualizer Plugin for Sketch.

To export a Sketch project to Kony Visualizer, do the following:

1. In Sketch, open the project you want to export to Kony Visualizer.
2. Open the Kony Visualizer plugin by clicking the **Plugins** menu, next click **Kony Visualizer**> **Export Panel**.
3. From the Kony Visualizer plugin, click **Global**. The plugin looks like this.

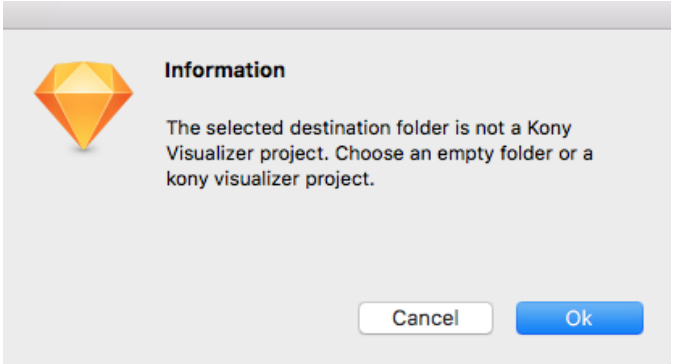


4. Specify the name of your exported project by setting its path. The folder you create or select becomes the exported project's Kony Visualizer project name. To set the path, click the ellipsis

button (...) of the Project Name text box, and then navigate to the path where you want to save the exported project to, creating a new folder, if necessary.
When you have finished, click **OK** .

> ***Note:*** You must set the export destination of the Sketch project to a Kony Visualizer project folder or to an empty folder. You cannot export a Sketch project to a non-empty folder.

From Kony Visualizer version V8 SP3 onwards, you will be shown this dialog.



5. If you are saving the project to a folder that you have already exported to, indicate whether you want to append the project to the existing exported content as an additional form (or forms), or overwrite and replace the existing exported content. To do so, click either **Append to Project** or **Replace Form** . By default, the Kony Visualizer plugin appends to any existing projects.
6. Select whether you want to export as Free Form or Kony Reference Arch project. If an existing project is selected, then this will get auto populated.
7. Select whether you want to export only the visible layers (as set on the Sketch), or export all layers regardless of whether they're visible or not.
8. Indicate the type of channel that you want to export the project to by selecting either **Mobile** , **Tablet**, or **Desktop** . The default value is **Mobile** .
9. Set the units options for the size and position of elements. You can select from the following:
    - **Px.** Uses pixels as the unit for the dimensions and positioning of project elements for Kony Visualizer. Pixels are a fixed, physical unit and do not scale proportionately

depending on the resolution of a devices screen. The rendering of an app on a low-resolution device will appear larger and more pixelated than it would on a higher-resolution device.

- **Dp.** Uses density-independent pixels, which is based upon the physical density of a device's screen. This allows the dimensions of the elements of your design to scale in size depending upon the screen resolution of your target device. Because the Dp unit is relative to a 160 dpi (dots per inch) screen, one dp is equal to one pixel on a 160 dpi screen. The ratio of dp to pixels varies with the screen density of the device. To accommodate this, you can set the dp-to-pixel ratio in the Kony Visualizer plugin, which is described in the next step.

- **%.** Sizes the various elements of the project as a percentage of the parent container of the project. The horizontal value and vertical value of the parent container are designated as 100% for their respective dimensions.

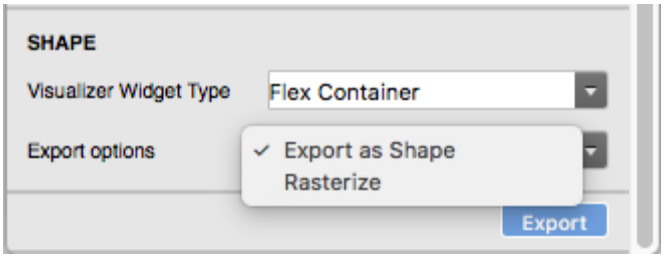
10. Set the ratio between pixels (Px) and density-independent pixels (Dp). This ratio is used only if you are using Dp as your unit of choice for element sizes or positioning. This ratio varies from device to device, depending on the physical density of pixels on the device's screen. Increasing the value of the ratio increases the number of pixels that equal one Dp. For instance, a value of 2.25 indicates that 2.25 pixels equals 1 Dp. To determine the ratio that you should use for your device, use the following formula:
    Px = DP*(The device's dpi/160)

11. Set the kind of widget you want the Sketch layers to be converted to on a default basis for Layer Groups, Text layers, and all other layers (i.e. Shape layers). To simplify the conversion process from a Sketch project to a Kony Visualizer project, you can globally determine the default type of widget that a given type of layer gets converted to. However, if you want to override the global value for a specific layer, you can do so (as described in step 13). The Kony Visualizer plugin categorizes the layers available in Sketch into three types of layers: Layer Groups, Text layers, and all other layers (i.e. Shape layers). The following table illustrates the global values that you can set for each of these three types of layers.

Sketch Layer	Kony Visualizer Widget Options
Group Layer. Defined as any layer that has layers, project elements, or other assets nested within it. You can mask group of shapes into a Group Layer which then can be rasterized.	Flex Container Flex Scroll Container
Text Layer. Any layer created using a Sketch Text.	Label (default) Button Text Area Text Box
Shape Layer. Defined as any layer other than a Layer Group or Text Layer. An Shape layer may consist of rasterized pixels.	Image (default) Flex Container Flex Scroll Container Button

12. Select the global export option for Group and Shape layers. Depending on the Kony Visualizer widget type that you selected for Shape layers, you can have such elements converted in either of the following two ways:

    - **Export as Shape.** The Shape layers are exported as the widget type selected in the Visualizer Widget Type drop-down list, and the layer styles are converted to the equivalent styles supported under Kony Visualizer skin properties. This option is not available if the Visualizer Widget Type you select is **Image**.



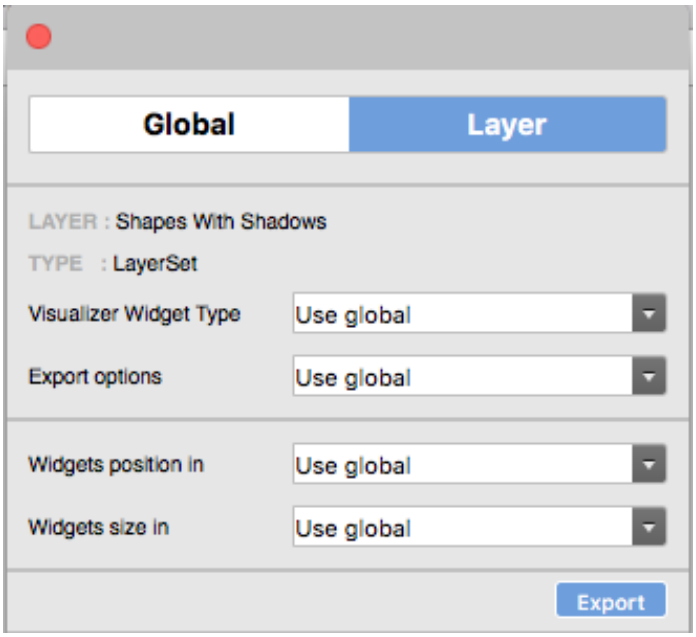
    - **Rasterize.** The layer is rasterized (flattened to pixels) and cropped, and saved as a .png file. If the Visualizer Widget Type you select is either a Flex Container, a Scroll Flex

Container, or a Button, the resulting converted widget uses the generated .png file as an image background in Kony Visualizer. And corresponding SVG files are also generated in resources/svg folder in Kony project. Which developer can use to get platform specific vector drawable.

13. Set export options for individual layers. For any of the layers you're exporting, you can override some of the global export options. This is especially helpful if, for example, you want to convert a particular layer into a Flex Scroll Container instead of a Flex Container, or to size a given layer as a percentage of the parent container rather than using whatever unit you selected as your global export option.

    To set export options for individual layers, at the top of the Kony Visualizer plugin, click **Layer**. Next, from the Sketch Layer tab, select the layer you want to custom configure, and then select the export options you want for that layer for the Visualizer Widget Type, and the type of units you want to use for the converted widget's positioning and sizing. Select the next layer you want to customize, and repeat.

    > ***Note:*** There will be no layer specific setting for symbols(user widgets in Kony Visualizer).



14. Once you have set all the global and layer-specific export options for your layers in a Artboard or a group, you will need to select the Artboards you want to export. You have the following options:
    - Export an Artboard or a group.
    - Export multiple Artboards(Export Limit: 15 Artboards)
    - Export a page(Export Limit: 10 Artboards)

    > ***Note:*** The Groups present in the page will not get exported, only the Artboards will.

15. From Kony Visualizer version V8 SP3 onwards, after you click on Export, you will be asked if you want to export the project as a zip folder or as a Kony Visualizer project folder. You can choose to generate a zip folder to be able to import the Sketch project in Kony Visualizer.

16. Once the exportation is finished, you will be able to view the exported project in the Visualizer Project whose path you had mentioned initially.
    If you did not export the Sketch project directly to a Visualizer project, then navigate on your computer to the folder where you exported the Sketch project, copy the folder, and then paste it into the desired Kony Visualizer workspace folder. The default workspace folder for Kony Visualizer is:
    `/Users/Username/KonyVizEWS/`

Open an Exported Sketch Project in Kony Visualizer

Once you have exported a Sketch project using the Kony Visualizer plugin and have copied it to your Kony Visualizer workspace, you can open it in Kony Visualizer.

To open an exported Sketch project in Kony Visualizer, do the following:

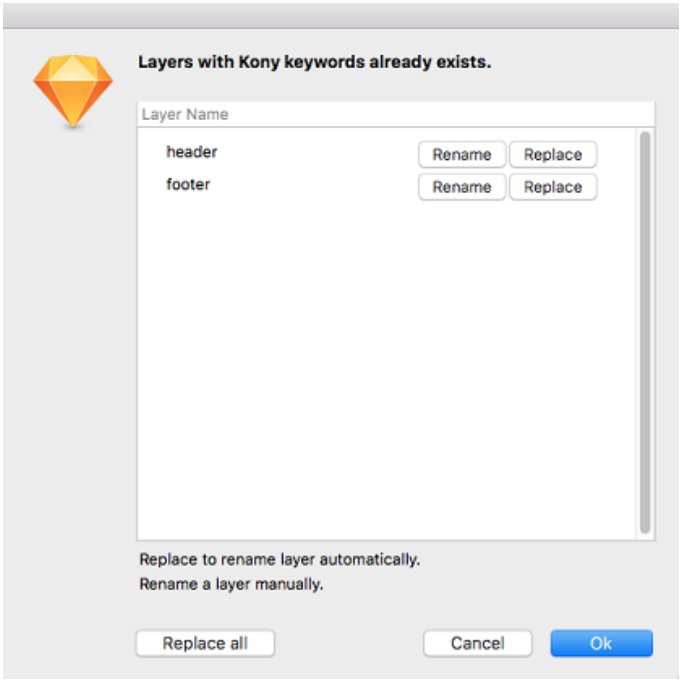
1. If Kony Visualizer is already open, on the **File** menu (the **Project** menu in Kony Visualizer), click **Refresh**. Otherwise, launch Kony Visualizer.

2. Do one of the following:

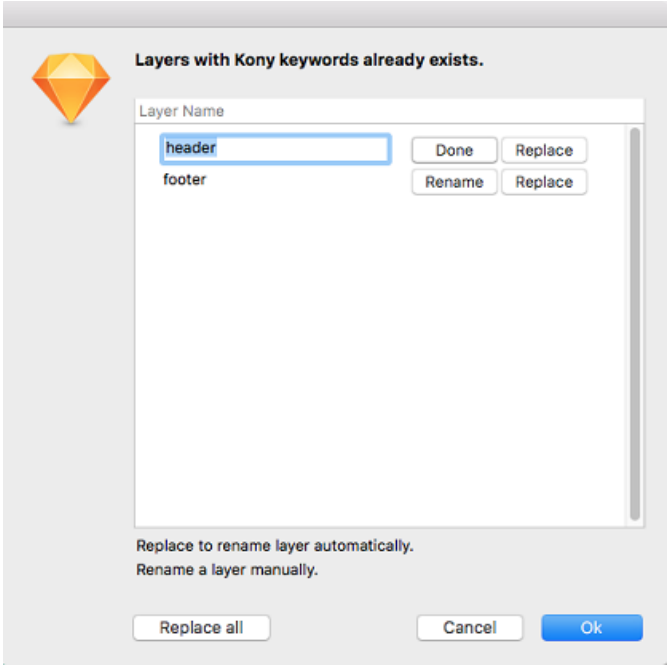
- On the Kony Visualizer launch screen, if Kony Visualizer is pointing to the workspace where you pasted the project, Kony Visualizer lists your exported Sketch project among recent projects. Click it.
- On the **File** menu (the **Project** menu in Kony Visualizer), point to **Open**, and then select from the list of available projects the project you exported from Sketch.

3. On the Project Explorer, click the **Project** tab, open the channel that you converted the Sketch project to, open **Forms**, and then double-click any of the forms listed. The form appears on the Visualizer workspace.

   Ensure that your widget name does not conflict with another widget. Ensure that the widget name is also not a Kony Visualizer keyword. If your widget name conflicts with another widget name, Visualizer displays a message. Using the options on the screen, you can rename or replace your widget name.



4. Click on **Replace all** to rename all the layers automatically. Click on **Replace** to rename a specific layer automatically. or,
   Click on **Rename** and type a different name for a specific layer. Click **Done** to save the change.



5. Click **Ok** to save the changes.

Limitations

While converting Sketch layers to Kony Visualizer widgets, the plugin has the following limitations:

- **Export** is supported for Group and ArtBoard Layer only, otherwise will throw an error.
- You cannot export a Sketch document or a Sketch Page directly. You can only export a Artboard or a Group Layer.
- In case of **Fills**, **Borders** and **Shadows** only the first enabled entry is considered during **Export**.
- Multiple symbols with the same name cannot be exported.
- After you export symbols to a Visualizer project in one instance and then try to export additional symbols to the same project at a later instance, the previously exported symbols are ignored. This is because symbols are exported as components to the Visualizer project.

Best Practices

To optimize the conversion of Sketch elements into Kony Visualizer elements, we recommend that you create your Sketch project using the following best practices.

- Only export images as needed, and avoid exporting .png files for anything other than icons or actual images and photos.
- Avoid using keywords that visualizer uses to avoid any issues while importing the exported project in Kony Visualizer.
- Do not rasterize text layers, since these can be exported as labels, text boxes, rich text, or button widgets directly into Kony Visualizer.
- Make sure you have the fonts installed for the current user at the following path : **~/Library/Fonts/**.
- SVG files are generated in resources/svg folder in Kony project. Which developer can use to get platform specific vector drawable.
- Instead of exporting each shape as image, it is recommended to **Mask** group of related shapes into a **Group Layer** and then export as image.
- Use appropriate size mask, when masking shapes to a Group Layer. Otherwise images dimensions and size will be large.
- Use ArtBoard Layer to define required Form, instead of Group Layer.
- Avoid empty groups and shapes as these will increase unused widgets in Kony Form. You can make them invisible in Sketch if required.

Import a Kony Visualizer Project

Applies to *Kony Visualizer*.

You can import a project that is located locally on your computer or you can import a project from a cloud account to Kony Visualizer.

For more information, click any of the following sections:

Import a Local Kony Visualizer Project

Import a Kony Visualizer Project from a Cloud Account

Important:

If the project you are importing includes Cordova content, ensure the following:

- The Cordova content needs to be in the following location:
  `<ProjectName>/web/cordova`
- Within the Cordova folder, the content must be structured in the following way:

```
config.xml
hooks
www
www/css
www/img
www/index.html
www/js
```

Import a Local Kony Visualizer Project

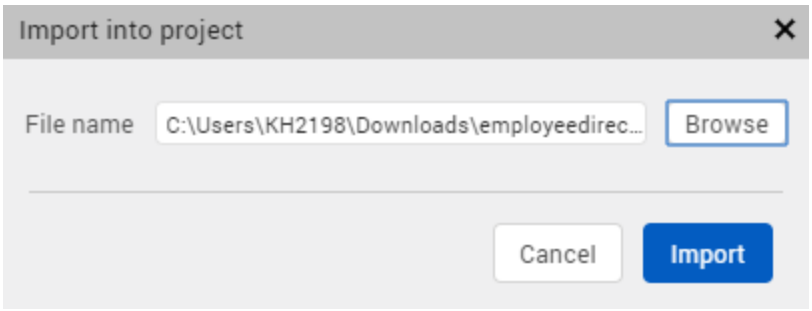
When you import a local Kony Visualizer project, you are essentially locating an archive (.zip) file, which Kony Visualizer then extracts into the proper location of your workspace.

Important: If your project was created using a version of Kony Studio older than 6.0, you first must import it into Kony Studio 6.0 (with the latest hotfix) and then you can import it into Kony Visualizer.

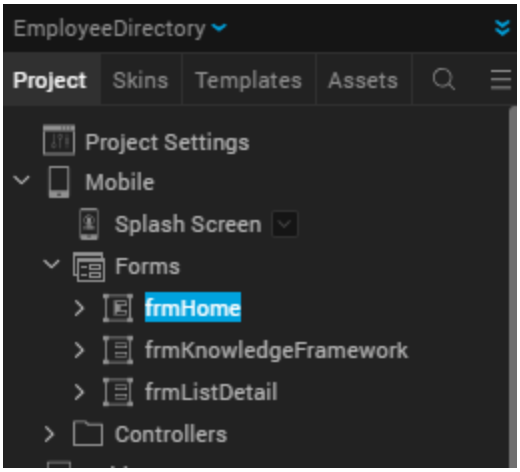
To import a local Kony Visualizer project, follow these steps:

1. On the **Project** menu, point to **Import**, and then point to **Local Project**.
2. You can do any of the following:
   - Click **Open as New Project** to import the local project as a new Kony Visualizer project.
   - Click **Add to Current Project** to import the local project to the current Kony Visualizer project.

   The **Import into project** dialog box appears.

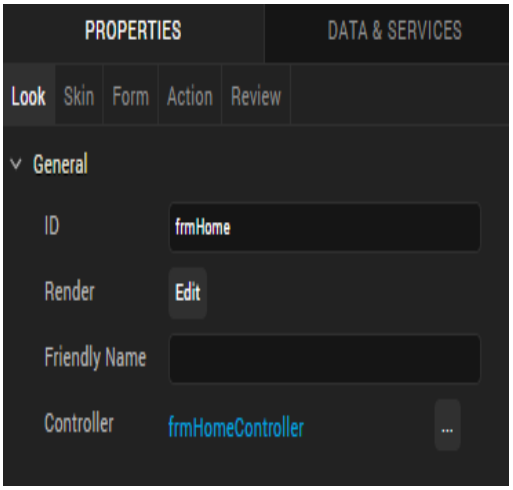


3. Click **Browse** to locate the archive file that you want to import, select it, and then click **Import**. The project is imported to your new or current project. On successful import of the local project, Kony Visualizer does the following:
   - The startup form is highlighted in the Project Explorer. Here, **Employee Directory** is the imported sample app and **frmHome** is the startup form.



   - The startup form of the imported project is displayed on the Project Canvas.

- The **Form** tab is auto-selected and displayed on the **Properties** panel.



> ***Note:*** If a project with the same name exists already, the **Conflict** dialog box appears, asking you if you want to overwrite the existing project. Click **OK** to overwrite the existing project. Click **Cancel** to end the import process.

Import a Kony Visualizer Project from a Cloud Account

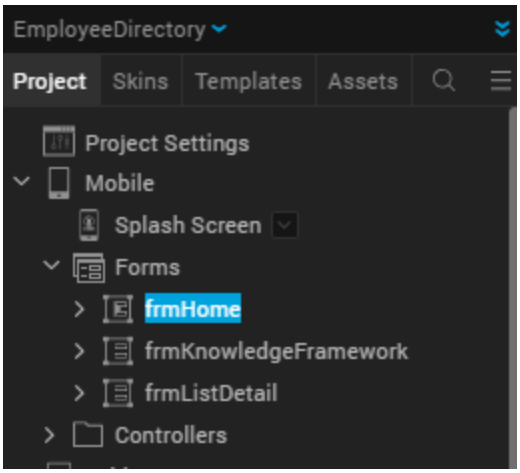
It is assumed that you have access to the cloud account where the project is located. For access, contact someone with admin privileges to the cloud account. They can invite you by referring to Invite Users to a Cloud Account.

To import a Kony Visualizer project from a cloud account, follow these steps:

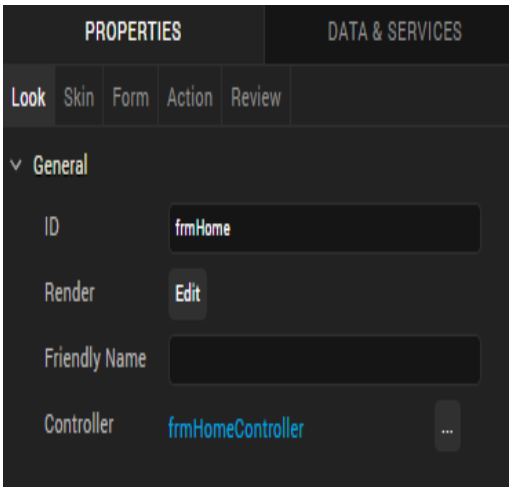
1. In Kony Visualizer, if you are not already signed in, sign in to your Kony account.
2. On the **Project** menu, point to **Import**, and then click **Cloud Project**. The **Import Project** dialog box appears.
3. Scroll through the list of available projects, and then click **Import** for the project you want to download.

4. In the list of projects, hover over the project you want to import. The project's field highlights, and an Import button appears along the right-edge of the project's field. Click **Import**. The project is imported to your current project. On successful import of the local project, Kony Visualizer does the following:

    - The startup form is highlighted in the Project Explorer. Here, **Employee Directory** is the imported sample app and **frmHome** is the startup form.



    - The startup form of the imported project is displayed on the Project Canvas.
    - The **Form** tab is auto-selected and displayed on the **Properties** panel.



> ***Note:*** If a project with the same name exists already, a dialog box appears, asking if you want to overwrite the existing project or rename the project you're importing with a different name. Click **Overwrite** to overwrite the existing project. Click **Rename** to open a dialog box where you can enter a new name for the app that you are importing. Then you can either click **OK** to proceed with the import process from the cloud or click **Cancel** to end the import process without downloading the project from the cloud.

Import an Application Extension

You can add custom functionality and content to your application by importing an application extension into your project. For information on creating an application extension, see Create an iOS Application Extension.

To import an application extension:

1. On the **File** menu (the **Project** menu in Kony Visualizer), select **Import**. Kony Visualizer displays a dropdown list of items that can be imported.
2. From the dropdown list, select **App Extensions**. Kony Visualizer displays a list of available extensions.
3. Select the extension you want to import.

Kony Visualizer adds the extension to an *app-extensions* folder and creates:

- An *Info.plist* information property list file.
- A *javascript* folder for associated JavaScript modules.
- A *resources* folder for any associated resources. To import a resource, right-click the resources folder and select **Import Resources**.

To configure the extension, right-click the extension and select **Configure App Extension**. To manage native function APIs, right-click the extension and select **Manage Native Function API(s)**.

> ***Note:*** Any changes made to the extension's code or resources after importing the extension into a project are local to that project.

To enable or disable an application extension, right-click the extension in the Project pane. If the extension is currently enabled, select **Disable** to disable it. If the extension is currently disabled, select **Enable** to enable it. Disabled applications extensions are not compiled or added to the application when you do a build.

To delete an application extension, right-click the extension in the Project pane and select **Delete**.

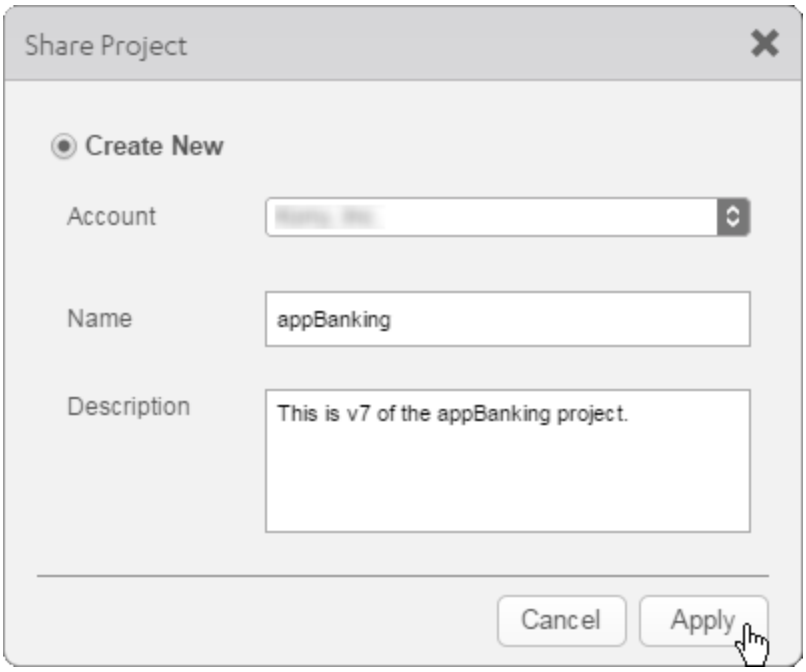
Share a Project on the Cloud

By publishing your project to the cloud, your stakeholders can download the complete project, make changes, and then upload the modified project back to the cloud.

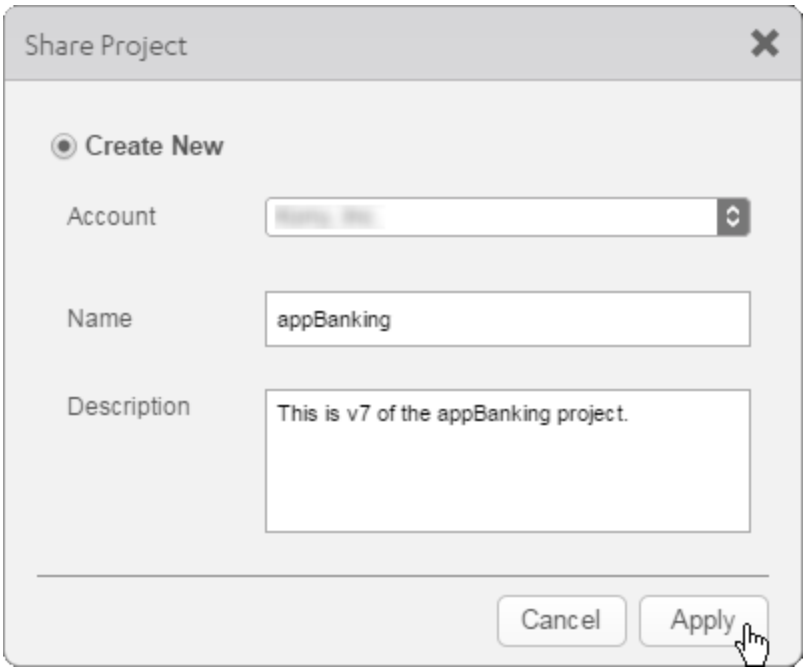
To publish your project to the cloud, do the following:

1. If you are not currently logged in to your Kony account, do so now by clicking **Login** in the top right corner of the Kony Visualizer window. The Kony Account sign-in window opens.
2. Enter your email and password credentials for your Kony account, and then click **Continue**. Once you are signed in, your account name displays in the top right corner of the Kony Visualizer window.
3. Do one of the following, depending on the edition of Kony Visualizer you are using:
    - For Kony Visualizer Classic, On the **File** menu, point to **Export**, and then click **Cloud**.
    - For Kony Visualizer, on the **Publish** menu, click **Project**. From Kony Visualizer V8 SP4 onwards, to publish a Project to the Cloud, refer to Generate a Cloud Preview.

The Share Project dialog box displays.



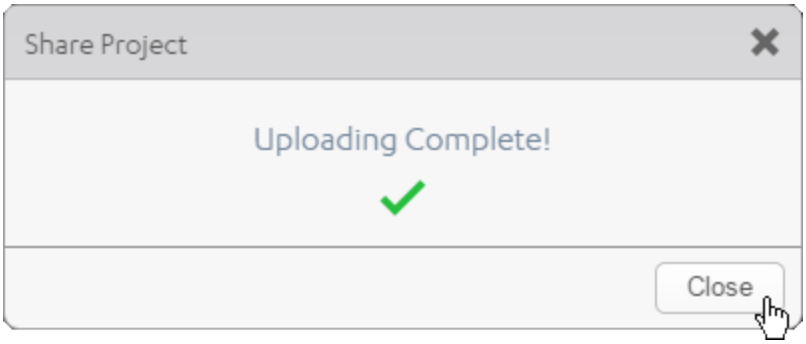
4. On the **Publish** menu, click **Project**. The Share Project dialog box displays.



5. From the **Account** drop-down list, select the cloud account to which you want the project published.

   > ***Note:*** If your profile is linked to a single account, the Share Project dialog box does not contain a list for the Account field.

6. In the **Name** text box, the project name is populated automatically. To change this, enter the name that you want people to see when they log in to the cloud account to download the project.
7. If you want, in the **Description** text box, type a brief description of your project, such as a version number or something about what's new in this iteration.
8. Click **Apply**. The upload begins.
9. Depending on the size of your project and the speed of your internet connection, uploading the project may take some time. After the project is uploaded, Kony Visualizer displays a confirmation message. Click **Close** to complete the process of sharing the project.



The maximum allowed file size is 256 MB (in .zip format).

Now that the project is uploaded to the cloud, users can download and view it using the import feature in Kony Visualizer. For more information, see Import a Kony Visualizer Project from a Cloud Account.

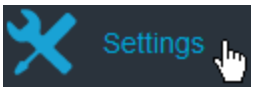
Invite Users to a Cloud Account

Publishing a project to a cloud account presumes necessitates that stakeholders have access to it. To grant access, you invite users to the cloud account.

To invite users to a cloud account, do the following:

.As an admin of a cloud, you can invite users by following the below steps:

1. Visit http://manage.kony.com and enter your Kony account credentials.
2. After validating your login credentials, click the **Settings** icon along the left edge of the page.

3. Click **Invite User**.
4. In the **Invite User** dialog box,
   - Enter the email address of the user you're inviting.
   - Select an account role to be assigned to the user. Available roles include: Owner, Admin, Billing, and Member.
   - From the existing cloud accounts, select the type of access to be provided to the user. Selecting the **Full Access** option results in adding the user to the cloud account.

   The user receives an email invitation to join the cloud account, complete with instructions on how to do so.

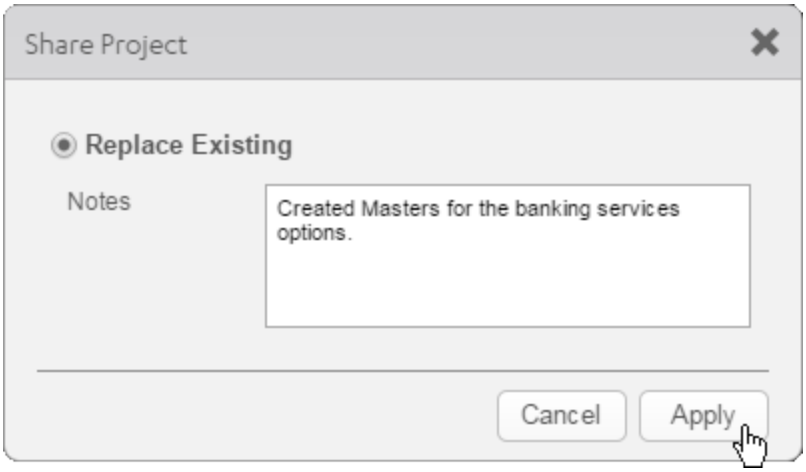
Modifying a Shared Project

This procedure assumes that you have already imported the shared project into Kony Visualizer. For more information, see Import a Kony Visualizer Project from a Cloud Account.

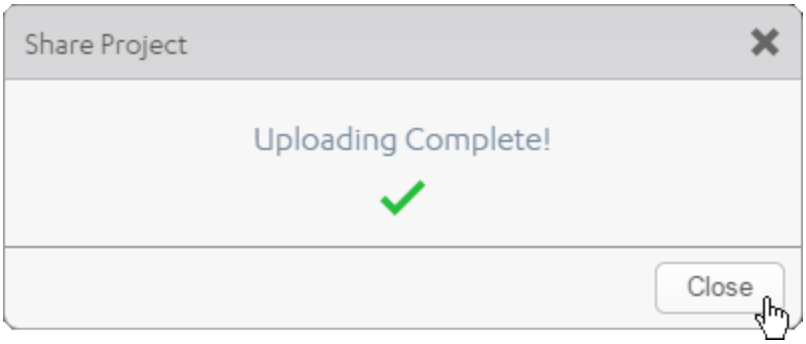
To modify a shared project, do the following:

1. In Kony Visualizer, if you are not already logged in, log in to your Kony account.
2. Open the shared project in Kony Visualizer.
3. Make the necessary changes to the project and save it. To make it easier for stakeholders to recognize the changes you have made, use the Review tab on the Properties Editor to add notes. For more information, see Capture Product Requirements with Review Notes.

4. On the **Publish** menu, click **Project**.
5. Kony Visualizer prompts you to confirm that you want to replace the existing project on the cloud. Click **Apply** to upload the modified project.



6. Depending on the size of your project and the speed of your internet connection, uploading the project may take some time. After the project is uploaded, Kony Visualizer displays a confirmation message. Click **Close** to complete the process of sharing the project.



Shared Project Limitations

You can re-upload a project to the same cloud that you imported it from. For example, a user shares a project on a cloud account **A**. You can import this project (assuming you have the necessary permissions) and make changes to the project, and then upload it back to the same cloud account **A**. But you cannot share this project on any other cloud account even if you have access to multiple accounts (like cloud account **B**, **C** and **D**).

Hence, it is always advisable that an admin create a cloud account specific to a project and invite all the stakeholders of the project to this account. This ensures that the project is available for all to share and modify.

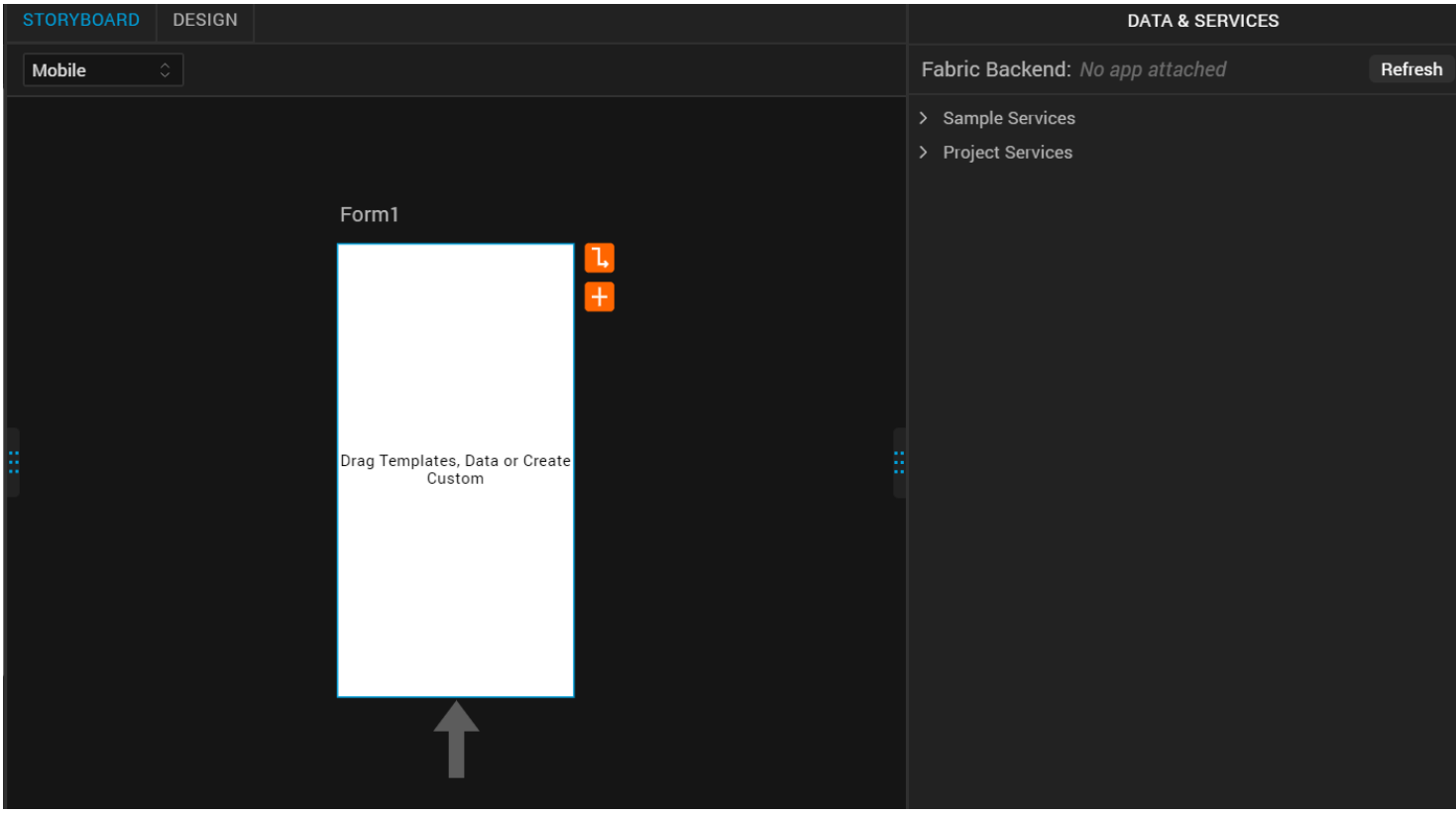
Create the Storyboard of Your App

From Kony Visualizer V9, while designing your app, you can develop a storyboard by using **Storyboard** view. Storyboard view in Kony Visualizer is a progressive way for you to quickly get the overall picture of an application. You can also use this feature to define simple navigation actions between various forms in a project.

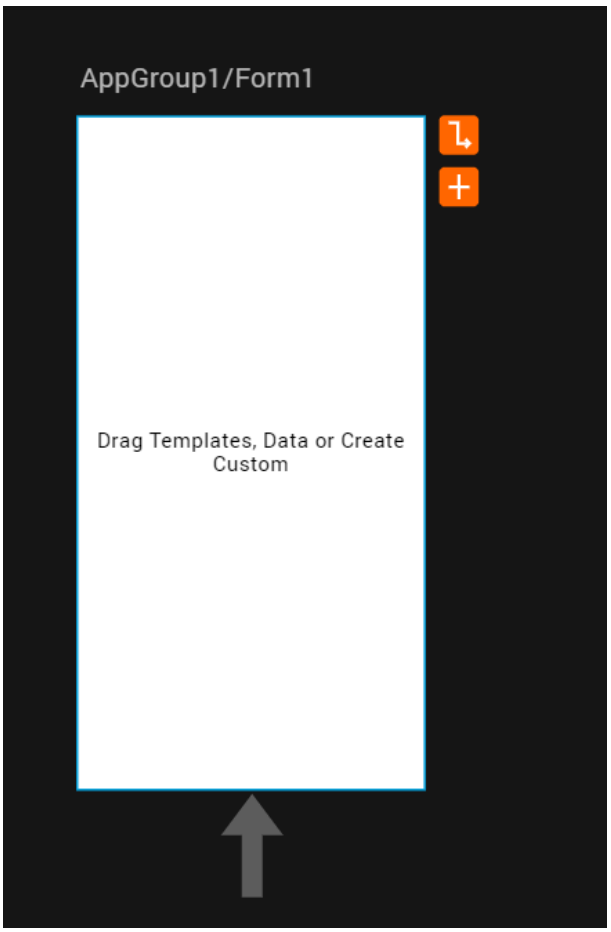
When you create a project in Kony Visualizer, the Storyboard view is displayed by default, containing one form (as shown in the image). The arrow at the bottom of the form (here, *Form1*) indicates that this form is the startup screen or landing page of the app.

> ***Note:*** If you are in Storyboard view and want to open a form in Design view, double-click the form.

> ***Note:*** In Storyboard view, only the Data & Services panel is displayed. If you want to access the Properties panel, click the **Design** view tab on the project canvas.



If a form is associated to an App Group, the name of the App Group is displayed by default with the form name in Storyboard view. Here, *Form1* is associated to *AppGroup1*.



This topic covers the following sections related to the Storyboard view feature:

- Benefits of Using Storyboard View
- Rename a Form
- Add a Widget to a Form
- Add a New Form with Navigation Link
- Add a Navigation Link between Two Forms

Benefits of Using Storyboard View

- Provides a more global, summarized view of a Kony Visualizer application.
- Reduces the number of assets that you have to manage at any given time for large Kony Visualizer projects.
- Simplifies the creation of basic navigation for clicks within apps at the beginning of the application design and development cycle.

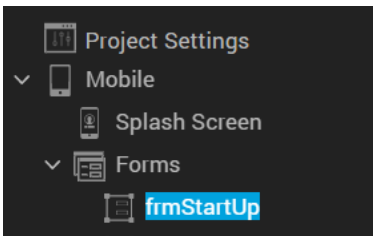
Rename a Form

To rename a form in Storyboard view, follow these steps:

1. In Storyboard view, click the **Form1** text. A box appears in which you can enter the new name for Form1.

> ***Note:*** In Storyboard view, it is not possible to rename the App Group to which a form is associated; you can only rename a form.

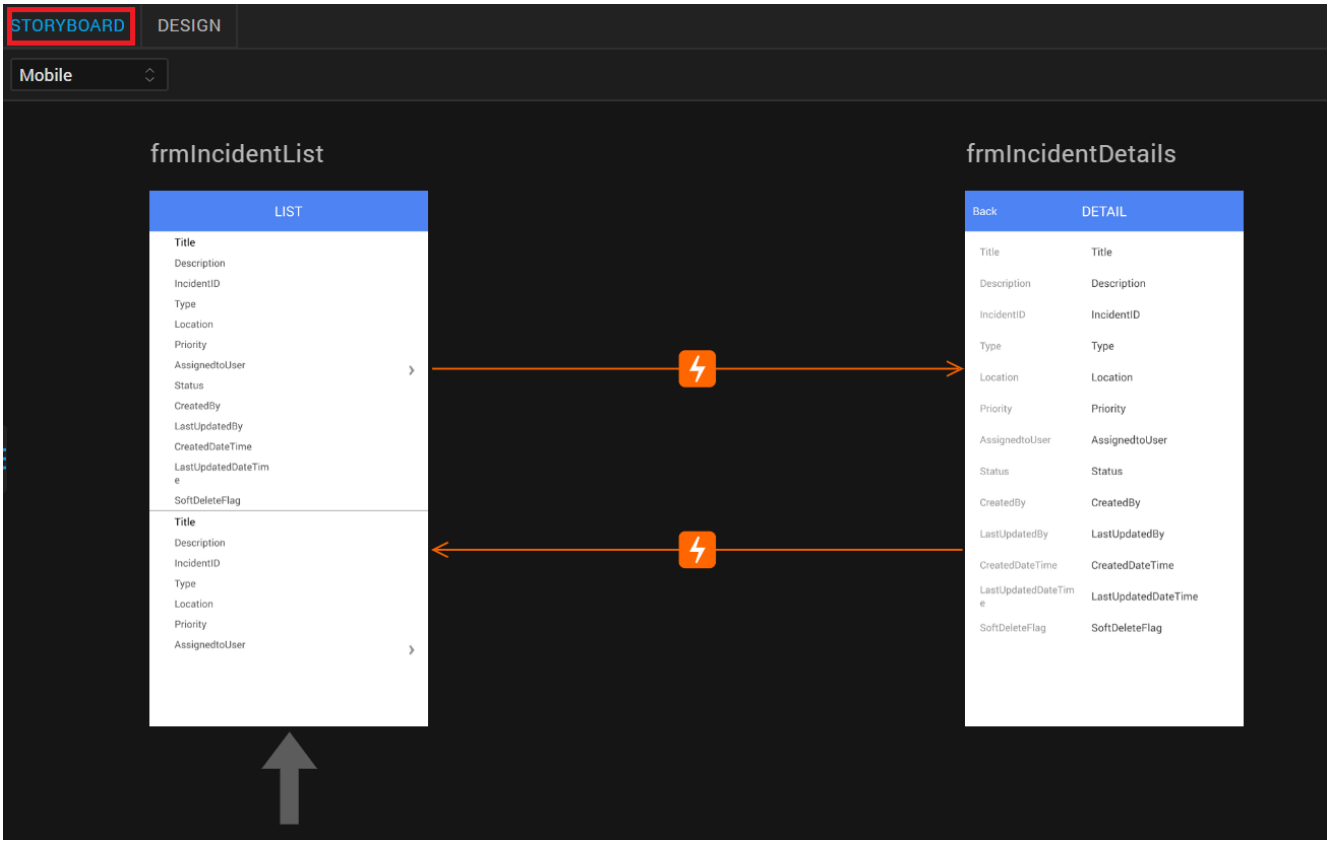
2. Type the required text in the box as the field name (here, *frmStartUp*), and press Enter. The form is displayed with the updated name in both Storyboard view and Design view.



Add a Widget to a Form

You can drag and drop any item or widget from the Default Library to a form in Storyboard view.

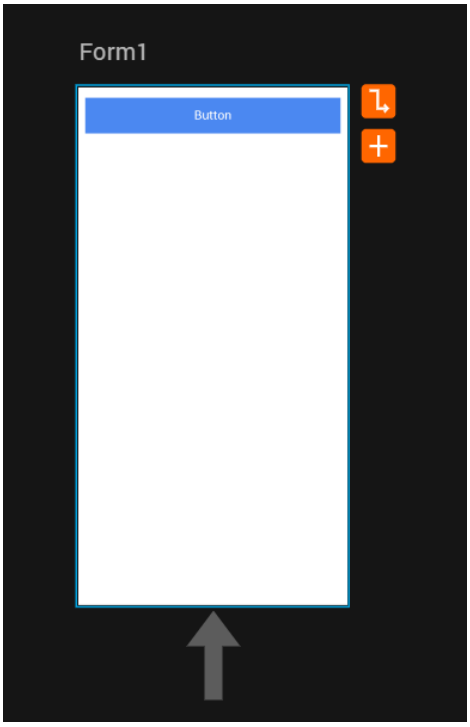
It is also possible to drag and drop services from the **Data & Services** panel onto the Storyboard view. When you drag and drop a method, object, or service anywhere on the Storyboard view canvas, the relevant forms and associated hard navigation links between the forms are automatically generated.



In the following scenario, we have added a SquaredButton widget to the form:

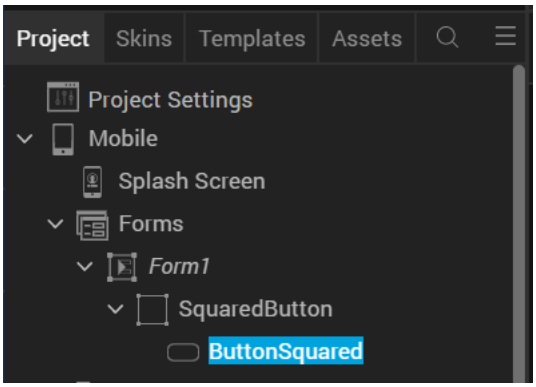
1. In Storyboard view, under **Default Library**, expand **Buttons**. The list of available Button widgets appears.

2. Drag and drop the **SquaredButton** widget to **Form1**.



3. If you want to delete or create a copy of the button, right-click the button on the project canvas and click **Delete** or **Duplicate** respectively.

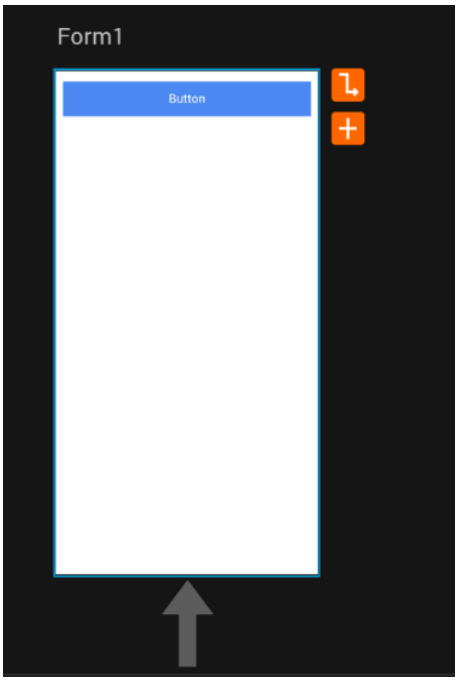
4. To view the details of the button that you added to the form, click the **Design** view tab on the project canvas. You can select the button on the Project Explorer, and then view or modify its details on the Properties panel.



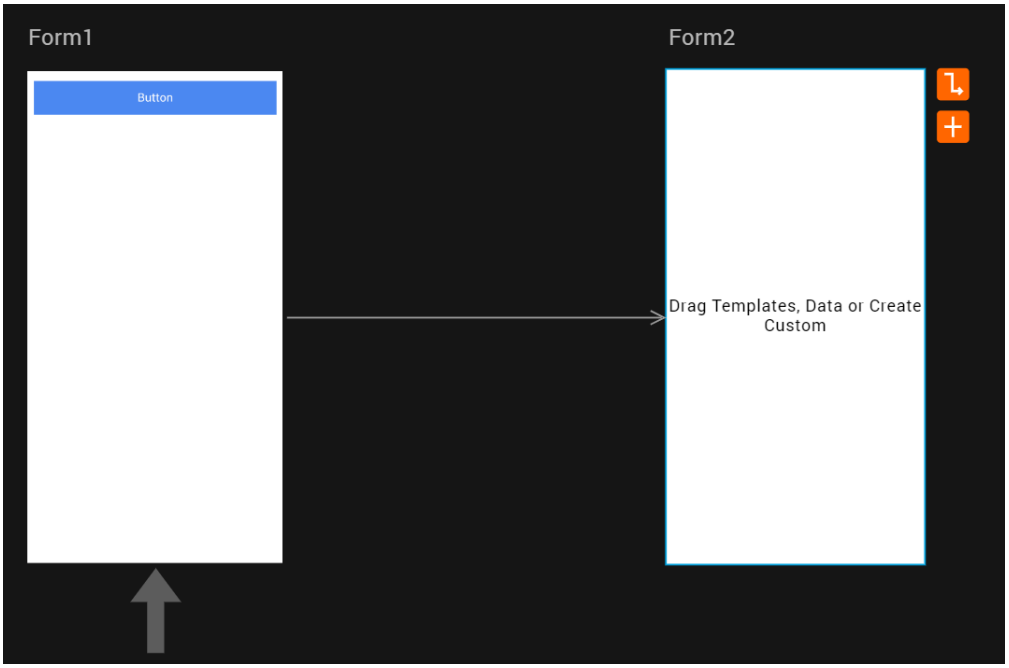
Add a New Form with Navigation Link

To add a new form with a navigation link to the existing form in Storyboard view, follow these steps:

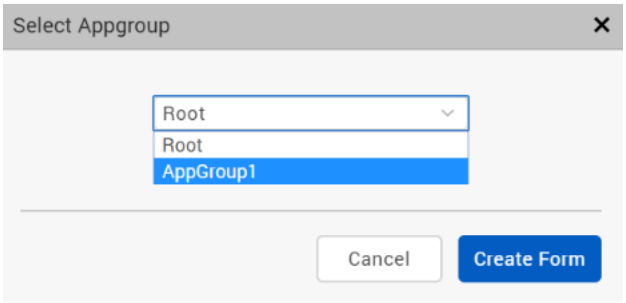
1. In Storyboard view, click **Form1**. Two icons appear beside the form.



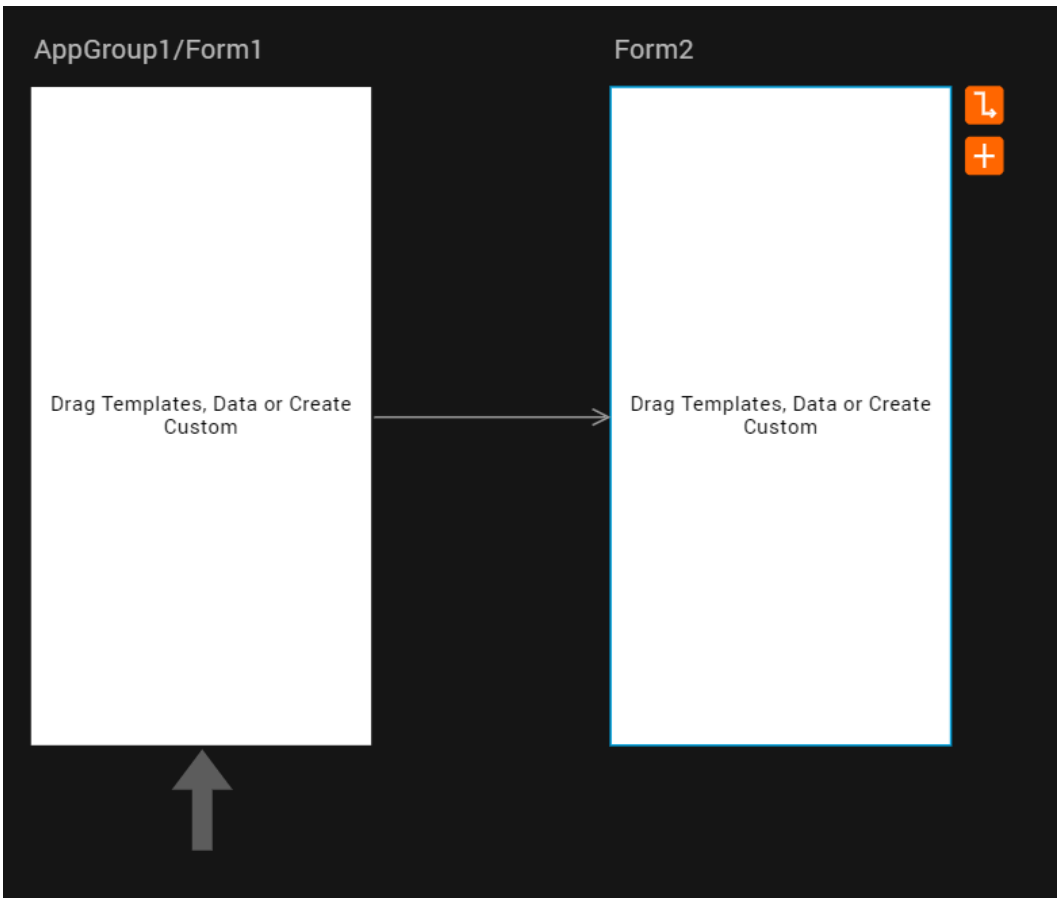
2. Click the Add icon  . A new form, named **Form2** by default, appears with a navigation link connecting to Form1.



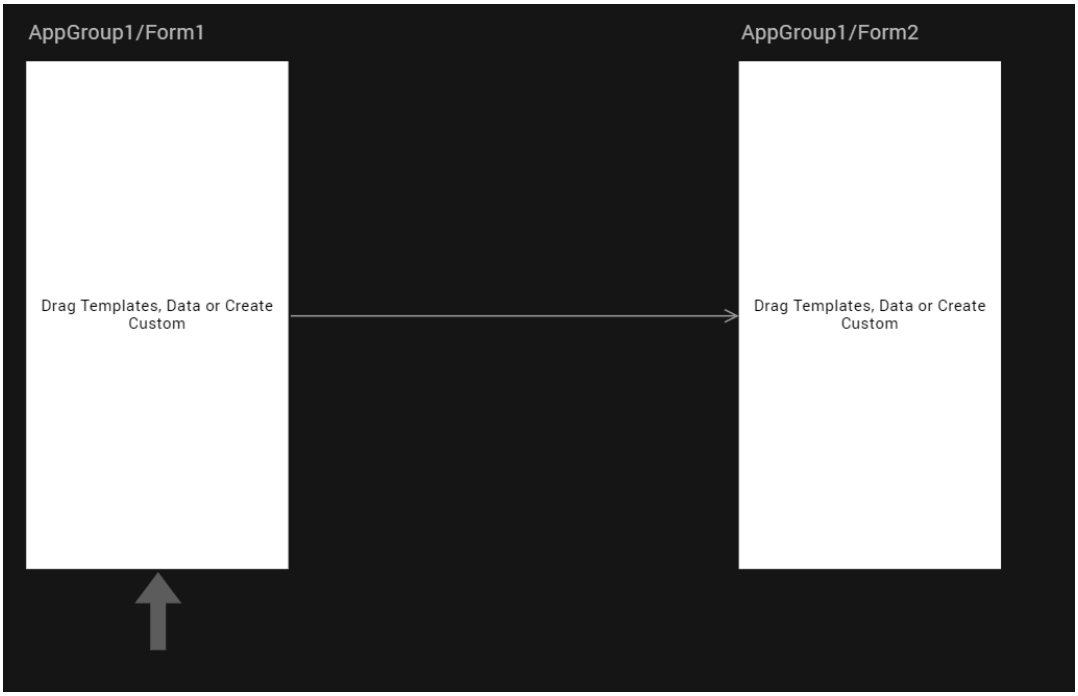
If an App Group exists under the same channel, when you click the Add icon  , the **Select Appgroup** dialog box appears.



If you choose **Root** and then click **Create Form**, a new form named **Form2** appears with a navigation link connecting to Form1. In the Project Explorer, Form2 is displayed under **Mobile** > **Forms**.

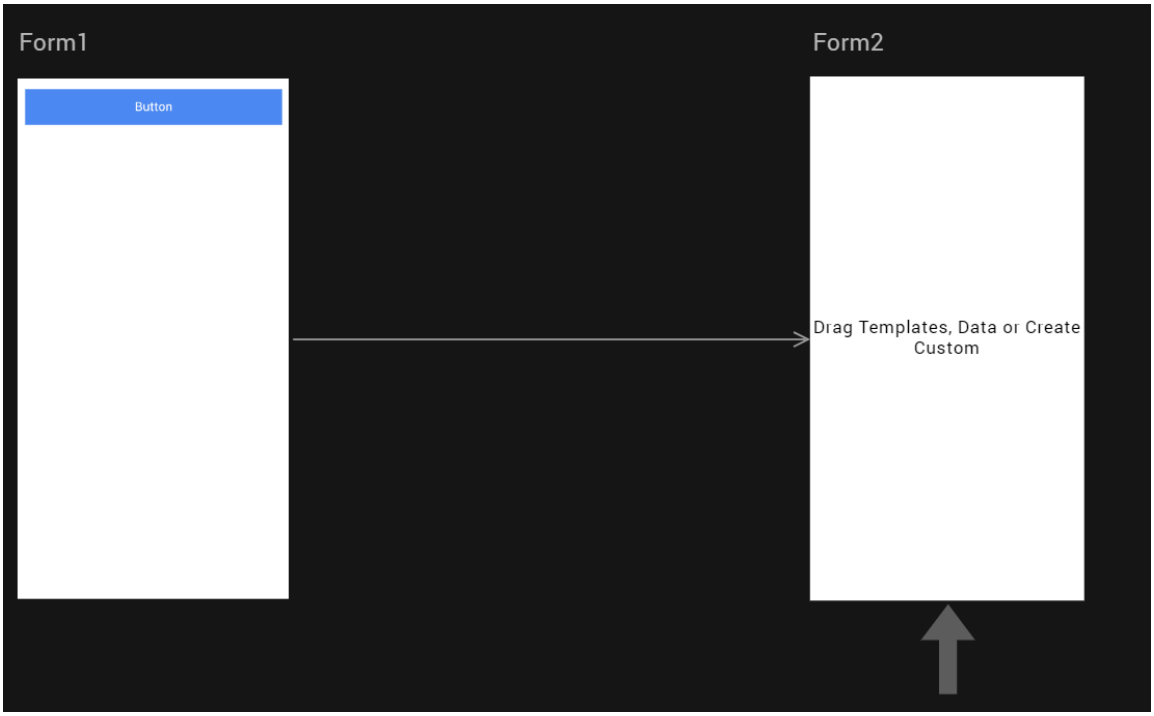


If you choose **AppGroup1** and then click **Create Form**, a new form **AppGroup1/Form2** appears with a navigation link connecting to Form1. In the Project Explorer, Form2 is displayed under **Mobile** > **Forms** > **AppGroup1**.



> ***Note:*** This type of navigation link is called as a soft link. It is just a visual representation of the navigation action between two forms; there is no actual navigation action involved between the forms through code.

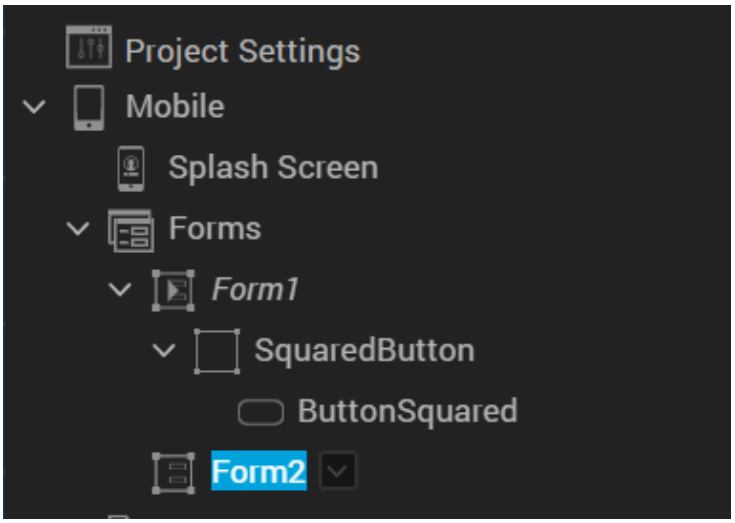
3. You can perform any of the following actions on Form2:
    - **Mark as Startup**: Right-click Form2, and then click **Mark as Startup**. Form2 is selected as the landing page of the app, and the arrow that indicates this appears under Form2.



- **Duplicate**: Right-click Form2, and then click **Duplicate**. A copy of Form 2 (named *CopyForm2* by default) is created without a navigation link, and appears beside it.
- **Delete**: Right-click Form2, and then click **Delete**. Form2 and its navigation link to Form1 are removed.

4. If you want to just delete the navigation link between Form1 and Form2, click the navigation link and press the **Delete** key from your keyboard.

5. To view the details of Form2, click the **Design** view tab on the project canvas. You can select **Form2** on the Project Explorer, and then view or modify its details on the Properties panel.



Add a Navigation Link between Two Forms

In this section, we will discuss how to create a navigation link between two forms in Storyboard view. There are two types of navigation links: **hard link** and **soft link**.

A *hard* link is an actual navigation action between two forms through code. You can create this type of link only in Design view.

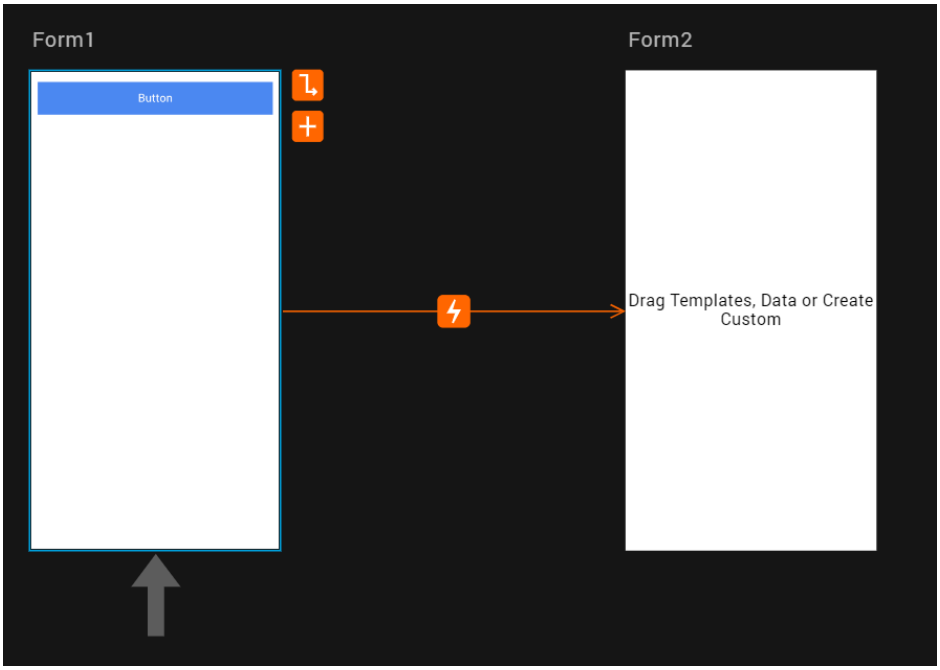
A *soft* link is just a visual representation of a navigation action between two forms in the storyboard of your app. There is no actual navigation action involved through code between the forms. You can create this type of link only in Storyboard view.

To add a hard link between two forms in Design view, follow these steps:

1. In Design view, click **Form1** > **SquaredButton**. We will take the example used earlier to create the navigation from Form1 to Form2, on the click of the button in Form1.
2. Go to the **Properties** panel > **Action**, and then click **Edit** for the **onClick** action. The Action Editor appears with the Diagram View by default.
3. In either the Diagram View, Design View, or Code View of the Action Editor, add the **Navigate to Form** action.

4. Select **Form2** as the form to be navigated to, and then click **Ok**. The onClick navigation action from Form1 to Form2 is added.

5. Go to Storyboard View. The hard navigation link between Form1 and Form2 is displayed.



> ***Note:*** If a soft navigation link already exists between two forms and later you define a hard navigation link between those two forms, the hard navigation link replaces the soft navigation one.

> ***Note:*** You cannot delete a hard navigation link; only soft navigation links can be deleted.

6. If you want to view or modify the details of the navigation action between the two forms in the Action Editor, click the Open Action icon  . The Action Editor opens with details of the navigation action.

> ***Note:*** If there are multiple navigation actions between two forms, when you click the Open Action icon  , all the created actions are displayed. You can click the required navigation action to view or modify the details in the Action Editor.

To add a soft link between two forms in Storyboard view, follow these steps:

1. In Storyboard view, click **Form2**. Two icons appear beside the form.
2. Click the Add icon  . A new form, named **Form3** by default, appears with a navigation link connecting to Form2.
3. To add a visual navigation link from Form1 to Form3, click **Form1** and then click the Draw Navigation Link icon  .

   > ***Note:*** You can only create soft navigation links between forms that do not have any hard navigation links between them.

4. Drag the navigation link from Form1 to Form3. The visual representation of a navigation action between two forms is created.
   If you want to delete the navigation link between Form1 and Form3, click the navigation link and press the **Delete** key from your keyboard.

> ***Note:*** A soft link is just a visual representation of the navigation action between two forms; there is no actual navigation action involved between the forms through code.

Export and Import Resources

With Kony Visualizer, you can export resources used within a project and then import them into other projects, saving time and effort. You can export and import the following:

Internationalization Resources

Export i18n resources

Import i18n resources

Themes

Export and Import Themes

Fonts

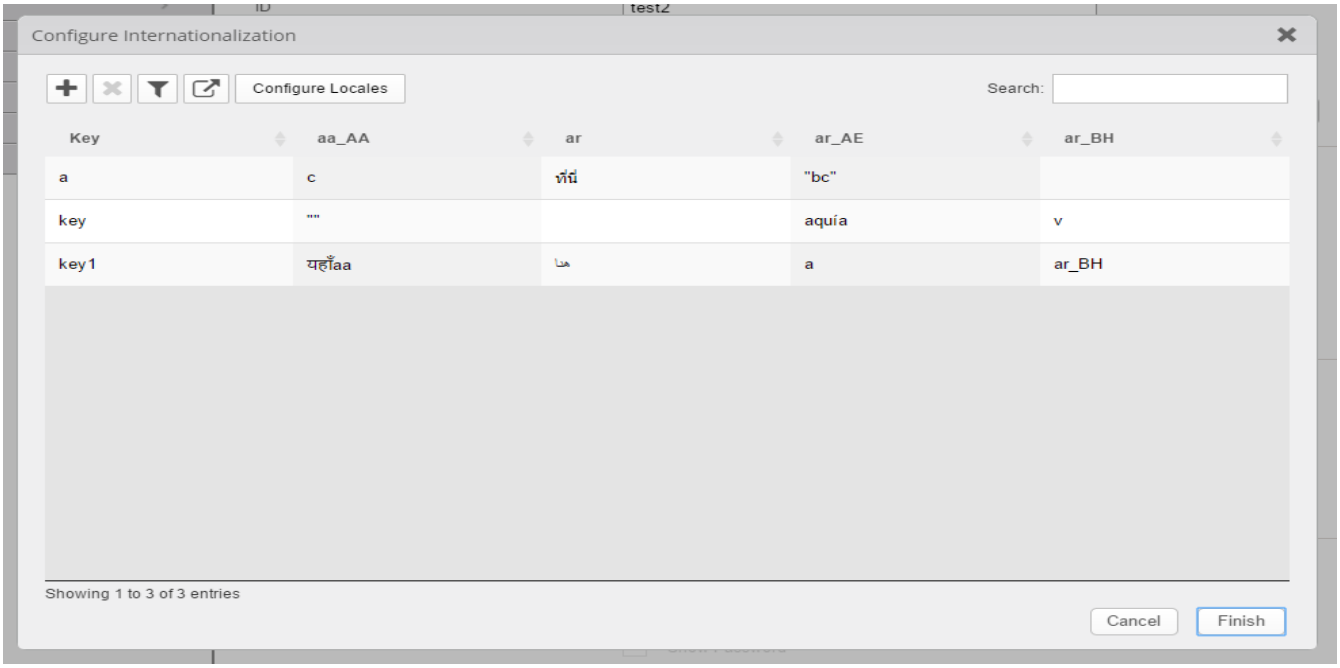
Export and Import Fonts

Application Extensions

Export and Import Application Extensions

Internationalizing (i18n) Application Content

The process of designing or developing an application in such a way that it supports various languages and regions is referred to as *Internationalization*, also known as *i18n*. In Kony Visualizer, the i18n features make it possible to build your app for various language locales without making changes to the application code or logic. By setting up the locales that you want to support and then adding what amounts to a vocabulary list for each locale, your app can become functional in multiple languages. Each entry that you make is called a key, and defines a given term in each of the locales that you want your application to support.



The following topics lead you through the process of implementing i18n functionality in an app:

Create Locales

Add i18n Content for Each Locale

Assign an i18n Key to a Widget

Search for An i18n Key

View a Locale Using the Preview App

In addition, you can also export a project's i18n settings and import them into another project.

Export Internationalization Settings

Import Internationalization Settings

Updating i18N Keys on Android Applications

Create Locales

To create locales, do the following:

1. In Kony Visualizer, on the Project (Edit menu for Kony Visualizer Classic) menu, click **Internationalization (i18n)**. The Configure Internationalization window appears.
2. Click **Open Locales**.
3. On the Predefined tab, select the checkboxes of the locales that you want your app to support. If you do not see the locale you want, click the Custom tab, and then add your own locale, giving it values for Language, Country, and Locale. For these values, use the patterns found in the predefined locales as examples to guide you.

   > ***Notes:***
   > - The Locale field must use the format aa_AA or aa. For example, jp_JP, or jp would be a valid locale name.
   > - The Country field can only contain letters.
   > - The Language field must begin with a letter.

4. Once you have created your locales, click **OK**. The Configure Internationalization dialog box now has a column for each locale you created. For every locale you configure, a corresponding empty folder is created in your workspace using the following path:
   `<workspace>\<application>\resources\common`
   You can add images to this folder which you can use in the project for corresponding locale.
5. Set the default locale by selecting the locale you want from the Default Locale drop-down list.
6. Click **Finish**.
   With your locales chosen, you can now define i18n keys for each locale.

Add i18n Content for Each Locale

When it comes to adding i18n content to your application, you have a couple of options:

- You can add i18n content within Kony Visualizer
  - By adding i18n keys one after the other manually, or
  - By adding all the i18n keys at once from a resource bundle

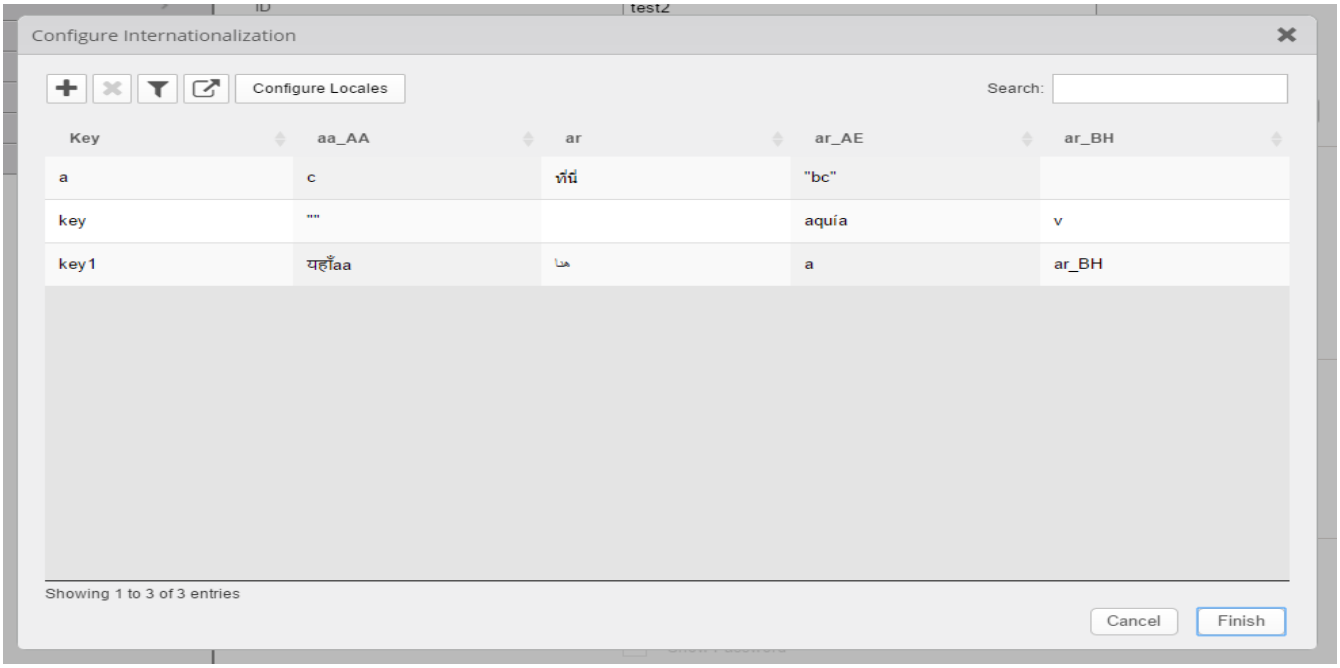
- You can add i18n content programmatically from the code using APIs.

Add i18n Keys Manually

This procedure assumes that you have already created the locales that you want your app to support. For more information, see Create Locales.

To add i18n keys manually, do the following:

1. In Kony Visualizer, on the **Edit** menu (the **Project** menu for Kony Visualizer), click **Internationalization (i18n)**. The Configure Internationalization dialog box displays.
2. In the Configure Internationalization dialog box, an initial row for an i18n key and the locales you want your app to support is displayed. In the Key field, type a name for your key, such as *key001*.
3. In each locale's field for the key you just named, enter the word or phrase for the term you're defining. For instance, if you're entering the Spanish equivalent for the English phrase *Sign In*, you would enter *Iniciar Sesión*.
4. Add a new row for each key you create by clicking the green plus sign button.
5. Repeat steps 3 and 4 for each new key you want to create.



6. Click **Finish**.

Add i18n Programmatically Using APIs

For information on how to use APIs to programmatically add i18n functionality to an app, see the Kony Visualizer API Developer's Guide.

Assign an i18n Key to a Widget

To assign an i18n key to a widget, do the following:

1. Using either the Project Explorer or the Visualizer Canvas, select a widget.

   The text of a widget uses the default language of the project.

2. On the Look tab of the Properties pane, select the key you want from the **Text i18n Key** drop-down list, or using the search icon, search and filter for the key you want.

> ***Note:*** For Group Widgets (such as ComboBox, DataGrid, and CheckBoxGroup) and Segment Widget, you assign i18n keys in the Master Data.

Search for An i18n Key

When adding editing, or deleting i18n keys, you can make it easier to locate particular keys by searching for them.

To search for an i18n key, do the following:

1. In Kony Visualizer, on the **Edit** menu (the **Project** menu for Kony Visualizer), click **Internationalization (i18n)**. The Configure Internationalization dialog box displays.
2. In the **Search** text box, type text that matches the key you're looking for. Keys that contain the text you type are listed in the table of i18n keys.

View a Locale Using the Preview App

To get a sense for what the user will experience when viewing an app in a particular locale, you can use the Kony Visualizer preview feature. The preview feature builds a preview version of the app, posts it to the cloud, and then gives you a publish code that you—or anyone you give the code to—can use to view the app on a device, such as a smartphone. The device simply needs to have the Kony Visualizer Preview App installed, which is available free from Kony on your device's app store.

To view a locale using the Preview App, do the following:

1. Set the default locale of the app to the locale that you want to view using Functional Preview. To do so, on the Project (Edit menu for Kony Visualizer Classic) menu, click **Internationalization (i18n)**, and then select the locale you want from the Default Locale drop-down list.
2. Follow the instructions provided in the topic, Preview an App on a Device.

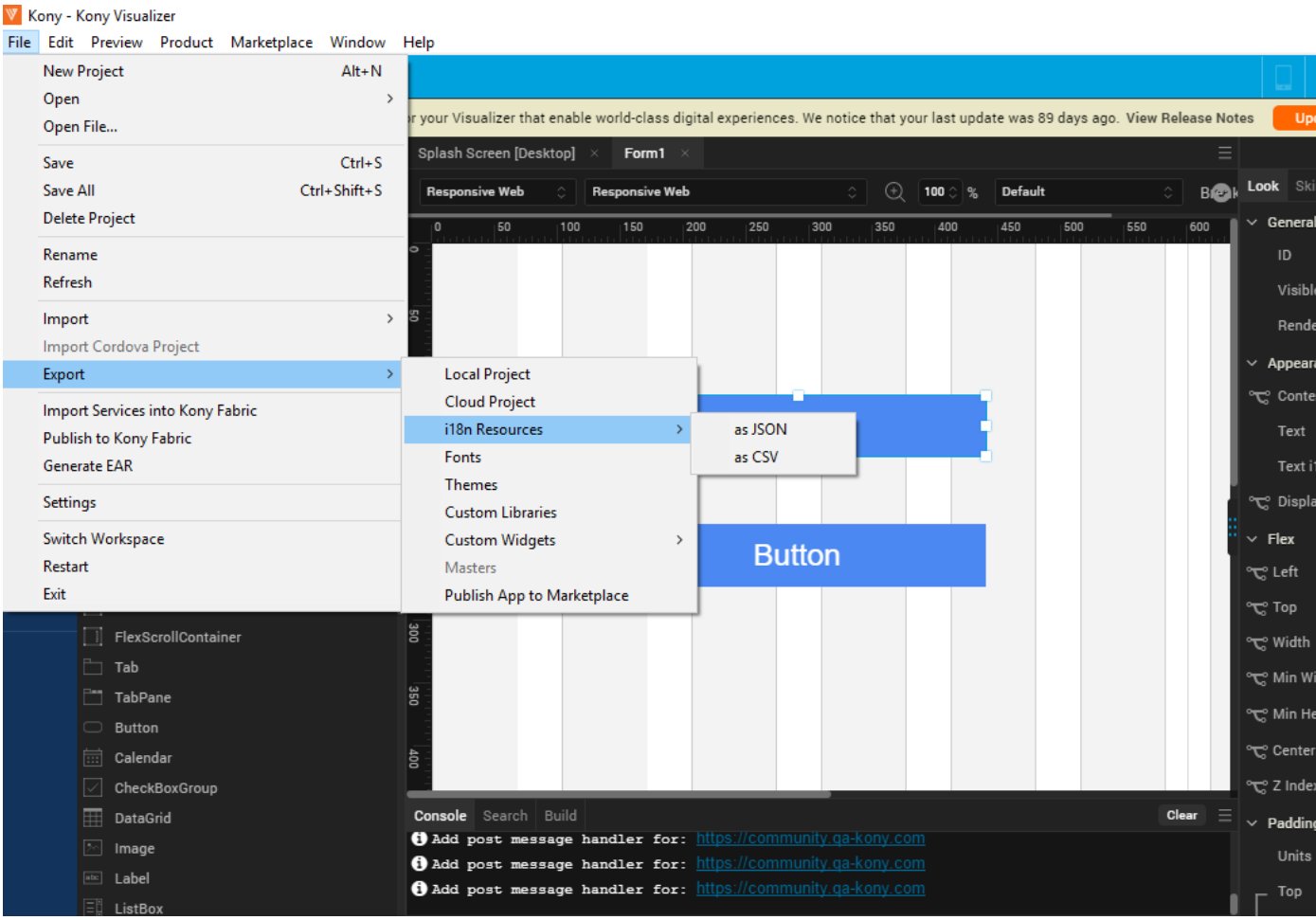
Export Internationalization Settings

You can export your i18n settings so that you can import them for use in a different Kony Visualizer project. From V8 SP4 onwards, you can export your i18n settings either as a JSON or as a CSV file format.

To export i18n settings, you can perform any of the following actions.

From the File Menu

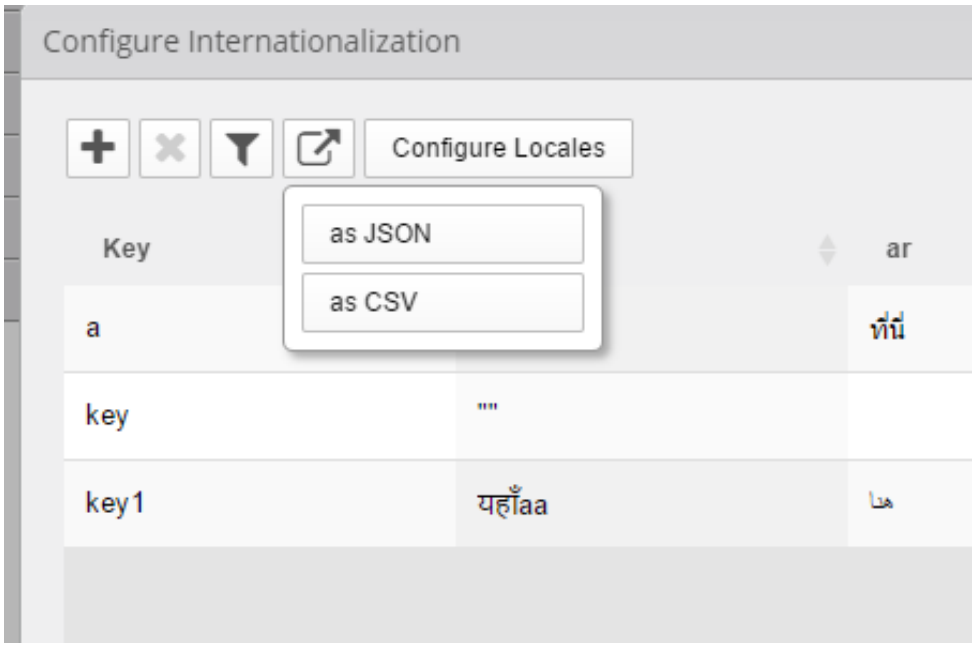
1. In Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer) , point to **Export**, and then click **i18n resources**.



2. Select the file format as either **JSON** or **CSV**. The Save As dialog box appears.
3. Navigate to the folder where you want to save the i18n settings, and then click **Save**. Kony Visualizer exports the i18n resources as a zipped file.

From the Edit Menu

1. In Kony Visualizer, on the Edit menu (the Edit menu in Kony Visualizer) menu, click **Internationalization (i18n)**. The Configure Internationalization window appears.
2. Click the **export** icon beside the Configure Locales button.
3. Select the file format as either **JSON** or **CSV**. The Save As dialog box appears.
4. Navigate to the folder where you want to save the i18n settings, and then click **Save**. Kony Visualizer exports the i18n resources as a zipped file.



Import Internationalization Settings

If you have the exported i18n settings from another Kony Visualizer project, you can import them into the project you're working on. From V8 SP4 onwards, you can import i18n settings either as a JSON or as a CSV file format.

To import i18n settings, do the following:

1. In Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer) , point to **Import**, and then click **i18n resources**.
2. In the Open dialog box, navigate to the zipped file that contains the i18n settings you want to import, select the file, and then click **Open**.

Important Considerations

The following are important considerations when importing i18n keys:

- If the current project contains i18n keys, the imported keys are added to the existing i18n file.
- If the name of an existing key is the same as an imported key, then the imported key values overwrite the existing key values.
- If the name of an existing key is the same as an imported key but the locales are different, then only those locales that are common are overwritten.
- If the imported i18n keys have locales that are not defined in the current project, then Kony Visualizer activates these locales in the current project, and then assigns the imported i18n values.

Resource Bundle

A resource bundle is a properties file that contains all the i18n keys, along with their values. It is locale-specific, that is, you will have one resource bundle per locale. To use resource bundles, you design your application to access the locale of choice at run-time, depending on the language the user chooses to display the app in. When the user chooses a different language, the resource bundle for

that locale is implemented. A resource bundle follows the naming convention of `<language_Country>.properties`. For example,

- For the United States English locale, the resource bundle is `en_US.properties`
- For the French Canadian locale, the resource bundle is `fr_CA.properties`.

Updating i18N Keys on Android Applications

If your Android app is published even once, when you modify the app (including i18N locales, values), you must ensure to change the version of the app from project settings. Modifying the Version Number and Version Code are mandatory steps for the changes to reflect in the app.

To update the Version Number, do the following:

1. From the **File** menu, navigate to **Project Settings**.
   The **Project Settings** window opens.
2. In the **Application** tab, update the version number to the next version. For example, if the **Version** is 1.0.0, modify it to 1.0.1
3. Click **Finish**.

To update the Version Code, do the following:

1. From the **File** menu, navigate to **Project Settings**.
   The **Project Settings** window opens.
2. Navigate to **Native** > **Android**.
   The Android tab contents appear. By default, the Mobile/Tablet tab displays.
3. Update the entry in the **Version Code**. For example, if the **Version Code** is 1.0, modify it to 1.1
4. Click **Finish**.

Export and Import Themes

To export a theme, do the following:

1. On the **File** menu (the **Project** menu in Kony Visualizer), point to **Export**, and then click **Themes**.
2. Select the skins you want to export.
3. Click **Finish**. The Save As dialog box displays.
4. Enter the file name of the .zip file into which you want the skins to be bundled, and then navigate to the folder where you want it to be saved.
5. Click **Save**. The skins you selected are bundled as a .zip file, and a status box informs you that the export was successful.

To import a theme, do the following:

1. On the **File** menu (the **Project** menu in Kony Visualizer), point to **Import**, and then click **Themes**.
2. Navigate to the .zip file of the theme you want to import, select it, and then click **Open**. The Import Themes dialog box opens.
3. Select the skins you want to import, and then click **Next**.
4. If a skin that is being imported has a name identical to a skin that exists in the project, you have an opportunity to rename the skin you are importing. If you do not rename the skin, it will overwrite the existing skin. Click **Next**.

   The skins you selected are imported, and a status box informs you that the import was successful.

Export and Import Fonts

Export Fonts from Kony Visualizer

To export a font file from Kony Visualizer, follow these steps:

1. On the **File** menu (the **Project** menu in Kony Visualizer), point to **Export**, and then click **Fonts**.

   > ***Note:*** If Kony Visualizer does not find any custom fonts in the project to export, it posts a warning to the Console to this effect, and takes no further action related to exporting.

2. Enter the file name of the .zip file into which you want the fonts to be bundled, and then navigate to the folder where you want it to be saved.
3. Click **Save**. The fonts you selected are bundled as a .zip file, and a status box informs you that the export was successful.

Import Fonts to Kony Visualizer

Earlier, you had to individually import each `.ttf` font file for every channel in Kony Visualizer. From V8 SP4 Fixpack 20 onwards, you can now import any font file directly to your Kony Visualizer project, and the font is automatically made available for all the channels. You also have the option to import a font file for one specific channel at a time, and the imported font is then available for that channel only.

In addition, when you try to import a font, Kony Visualizer verifies the name of your font file and the original font family name. If there is a mismatch between the names, an alert appears. This alert indicates that the font file has been renamed to the actual font name, for a glitch-free functioning of the font within Visualizer.

To import a font file to Kony Visualizer, follow these steps:

1. In Kony Visualizer, go to **Project Explorer** > **Assets**.
2. Depending on your requirement, do any one of the following:
   - Import a font for all channels:
     a. Right-click **Fonts**, and then click **Import Fonts**. Your local File Explorer window appears.
   - Import a font for specific channels:
     a. Expand **Fonts**. The list of available channels to which you can import fonts, appears.
     b. Right-click a particular channel, and then click **Import Fonts**. Your local File Explorer window appears.
3. Browse for and select the required font(s), and then click **Open**.

> ***Note:*** If the selected font file is already available in Kony Visualizer, a **Conflict** dialog box appears indicating that the font exists for certain or all channels. Click **Override** to import and replace the existing font file; alternatively, click **Skip** to abort the import process.

> ***Note:*** If you have modified the original name of the font file that you are trying to import, Kony Visualizer automatically renames the file to the actual font name while importing it.

4. Once the font file is imported successfully to Kony Visualizer, the **Import Fonts** dialog box appears displaying an appropriate success message. Click **Close**.
5. You can now use the imported font for the required channels in Kony Visualizer.

Create Screens

In Kony Visualizer, you build screens using forms that you populate with widgets such as buttons, text fields, labels, and the like. Any given screen that the user sees is the result of a form, and your application can consist of as many or as few forms as it needs. To have the app move from one form to another, you associate that action with a particular widget, such as a button.

You can use a form across multiple **platforms**[1], and you can adapt a form created for one **channel**[2] for use on other channels. For information on creating forms for Apple Watch, see Apple Watch.

> ***Note:*** In Kony Studio 6.5, you could port a project from one platform to another, and to channels within the target platform. However, issues would arise due to the wide variations in form factors from one platform and channel to another. In Kony Visualizer, universality across platforms and channels is achieved through the use of components and masters; platform porting is not supported in Kony Visualizer. For more information, see Working with Components, Kony Marketplace, and Masters. .

To add a form to your application, do the following:

1. On the Project Explorer, click the **Project** tab.
2. Expand the channel you want to create the form for, either Mobile, Tablet, Watch, or Desktop.
3. Hover over **Forms**, click its context menu arrow, select **New Form**, and then click **Flex Form**. A new form is added to the **Project** tab, and becomes the form of focus on the Visualizer Canvas.

   > ***Note:*** Do not select **VBox Form** from the **New Form** menu. This is used only for legacy applications that were created in Kony Studio 6.0 and earlier.

[1]The operating system of a given device. iOS and Android are examples of platforms.
[2]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

4. To rename the form, on the **Look** tab of the Properties Pane, type the new name in the ID text box. The form name cannot have any spaces.

Apple Watch

The Apple Watch is a smartwatch that runs on the WatchOS platform, and is paired to an iPhone (5 or above) to perform functions such as calling, texting, and receiving notifications. Kony Visualizer supports Apple Watch 42mm and Apple Watch 38mm.

The following are the main features of a watch:

- Forms
- Notifications
- Glances
- Reviews

Forms

A form is the starting point for any application, and is the top most container. A form can contain any number of widgets but cannot contain another form. For more information regarding Watch widgets, see Add Watch Widgets.

Your Watch app can consist of multiple forms, and you can move from one form to another using a couple of different methods. You can use an action sequence, such as assigning the Navigate to Form action to a button so that, when it's clicked, it displays a different form.

Add a Form to a Watch

To add a form, follow these steps:

1. From the **Project** tab of the **Project Explorer**, expand the **Watch** folder. Right-click **Form** > **New Form**. This results in creating a new form with a default name assigned to it.
2. You may choose to rename the form by clicking the down-arrow next to the form and then clicking **Rename**. The name of the form becomes editable.
3. To make a form function as the app's startup form (that is, the first form to appear after the application is loaded), click the down-arrow next the form, and then click **Mark as Startup**.

The following widgets are supported on a form:

Group	Button	Date	Image2
Label	Line	Slider	Timer
Map	Segment2	Switch	

Add a Widget to a Form

To add a widget to a form, do the following:

To add a widget to a form, follow these steps:

1. From the Widget Library, drag the required widget and drop it on a form on the Visualizer Canvas.
2. If you want, rename the widget.
3. Save the form.

Duplicate a Form

When you want to have a form that is similar to an existing form, you can duplicate the required form and reuse it.

To duplicate a form, do the following:

1. Right-click the form you want to duplicate, and then click **Duplicate**.

   A duplicate form is created.

2. If you want, rename the form.

Delete a Form

When you no longer require a form, you can delete it.

To delete a form, do the following:

1. Right-click the form you want to delete, and then click **Delete**.
2. A deletion confirmation message appears. Click **OK** to delete the form.

Open the Assets' Folder

To navigate to the location where the form assets are available on your computer, do the following:

1. Right-click the required form and click **Resources Location**.

   The Windows Explorer where the form assets are located appears.

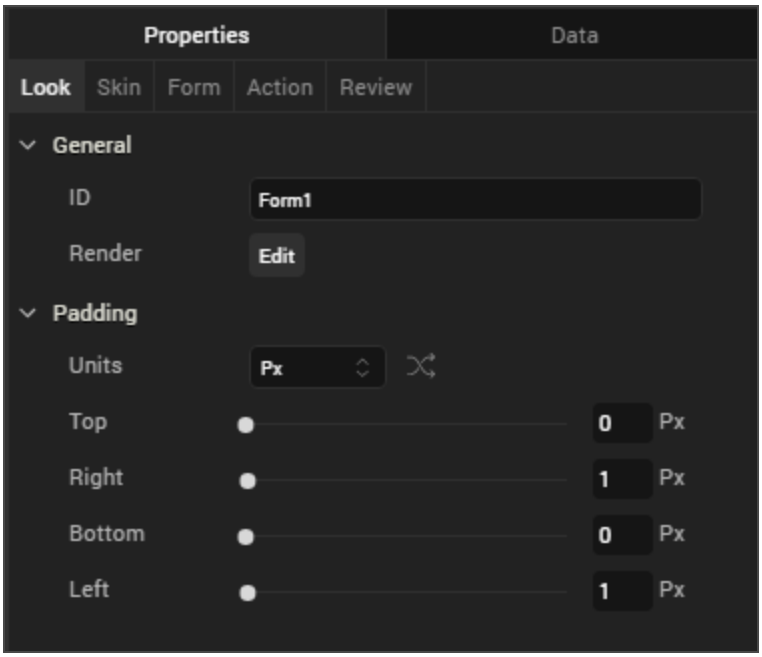
Form Properties

You can customize the following properties of a form:

- Look
- Skin
- Form
- Action
- Review

Look

On the Look tab, you set properties that control the way a form appears on an app. The following are the major properties you can condition for a form:



1. **ID**: It is the unique identifier of the form. When a form is added, system assigns a unique ID (or name). You can rename the form ID, if required.
2. **Render**: This property defines if a form should appear on a specific platform. By default, a form is rendered for all the platforms.

   If you do not want to render a form for a specific platform, click the **Edit** button against the **Render** to open the **Fork Platforms** dialog box. Clear a platform check box for which you do not want to render the form.
3. **Padding**: Defines the space between the content of the form and the form boundaries. The following padding options are available for the form.

Property	Definition	Action
Top	Top padding	Move the slider to adjust the top padding of the widget.
Bottom	Bottom padding	Move the slider to adjust the bottom padding of the widget.
Left	Left padding	Move the slider to adjust the left padding of the widget.
Right	Right padding	Move the slider to adjust the right padding of the widget.

Skin

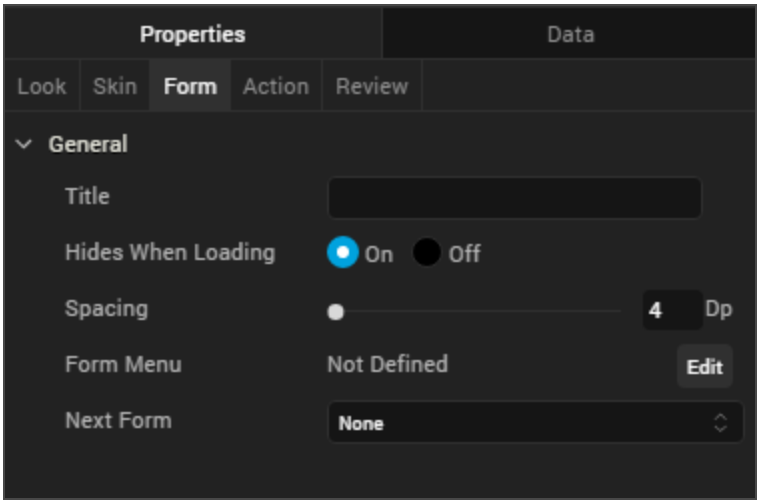
On this tab, you define properties such as background color, borders, and shadows for a form. For a Watch form, you can define a skin its **Normal** state.

For more information on skin properties, see Understanding Skins and Themes.

> ***Note:*** Custom themes developed in Kony Visualizer are not supported on Apple Watch. Apple Watch applications support the default theme of Kony Visualizer.

Form

On this tab, you set properties that are common for all the platforms and also, set properties that are unique to a platform.



Title

Use this property to set the title of a form.

Hides When Loading

This property specifies if the form should be hidden while loading the application.

Spacing

This property defines the amount of space between the widgets in a notification. The spacing values are assigned in Dp format and can take values in the range of 0-100 Dp.

Next Form

From the **Next Form** list, click the form that should appear after the current form.

Action

On this tab, you define the events that are executed when an action is run. For a notification, you can run the following actions:

- onInit: This action allows you initialize your interface controller.
- onWillActivate: This action lets you know that your interface will soon be visible to the user. Use this method only to make small changes to your interface. For example, you might use this method to update a label based on new data.
- onDidDeactivate: This action allows you to clean up your interface and put it into a quiescent state. For example, use this method to invalidate timers and stop animations.
- onAwake: This action is called to let you know that the interface controller's contents are onscreen.
- localNotificationActionHandler: This action delivers a local notification payload and a user-selected action to the interface controller.
- remoteNotificationActionHandler: This action delivers a remote notification payload and a user-selected action to the interface controller.

For more information on using these actions, see the topic, Add Actions.

Review

This tab can be used to add Notes and Comments to the selected form.

You can search for the notes written about a form by typing its name in the **Search**box.

You can expand, contract, add, and delete notes.

Notifications

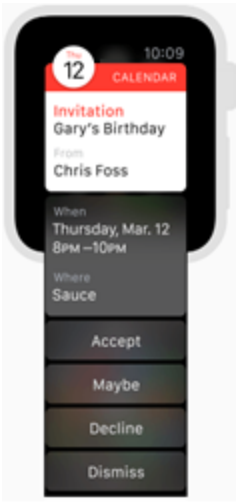
A notification informs a user about information received on their Apple Watch, such as a message, an upcoming appointment, or new data. These notifications appear in the form of an alert message or a badge on the app icon. Depending on the app settings, you may also hear a voice alert when you receive a notification. The iOS platform supports both local notifications and remote notifications (or push notifications.)

A local notification is scheduled by an app and delivered on the same device. Whereas, a remote notification is sent by your server to the Apple Push Notification service (APNS) and from there the notification is delivered to your device.

The notifications you receive are either short look notification or long look notifications. To customize a

- Short-Look Notification uses a static notification.
- Long-Look Notification you can use both static notification and dynamic notification.

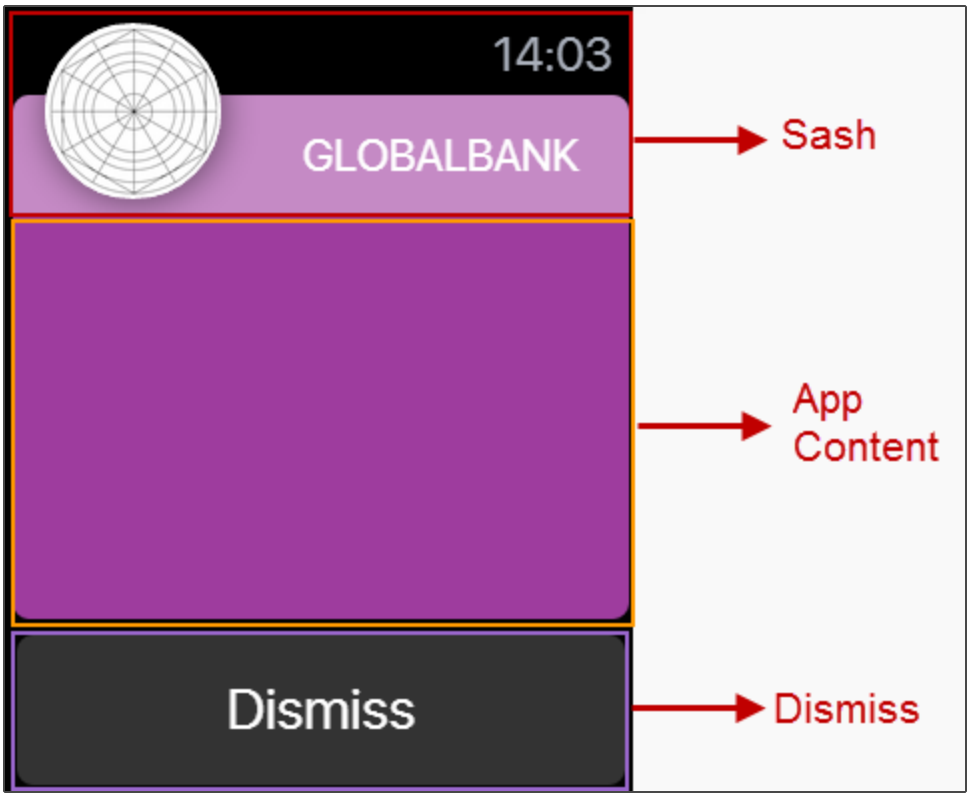
The following example shows how a notification appears on your watch.

Short-Look Notification	Long-Look Notification
	

You may receive a Short-Look notification instead of a Long-Look notification due to the following reasons:

- Battery Saver is on.
- You have exceed your carrier's data plan limits.
- Data Sense is enabled and you have exceeded your specified limit.
- Dynamic notification not configured.

A notification consist of following panes:



- **Sash**: It is the top pane of a notification. In Sash, the app icon (you can upload the icon from the Project Settings > Apple Watch OS > Long-Look Notification Icon) and project title are displayed.
- **App Content**: The central pane of a notification. Here, you can design a notification using the Kony Visualizer widgets.
- **Dismiss**: The bottom pane of a notification. This pane contains system defined **Dismiss** button only. You cannot delete this button or add another widget.

Add a Static Notification

In Kony Visualizer, initially when you add a notification, it implies that you are adding a **static** notification. Only after you add a static notification, you can add a dynamic notification.

To add a static notification, follow these steps:

1. Expand the **Watch** channel and hover on **Notifications** to display a drop-down arrow.
2. Click the drop-down arrow and click **New Notification**. A new static notification is created. The system assigns a unique name to this notification.
3. Rename the notification (if required) by clicking the drop-down arrow next to the notification and then clicking **Rename**. The notification name becomes an editable text field. Rename the notification and then press **Enter** on your keyboard.

Add a Dynamic Notification

You can add a dynamic notification only after you have added a static notification. For each static notification, you can add only a single dynamic notification.

To add a static notification, follow these steps:

1. Hover on a static notification to display a drop-down arrow.
2. Click the drop-down arrow and then click **Create Dynamic Notification**.

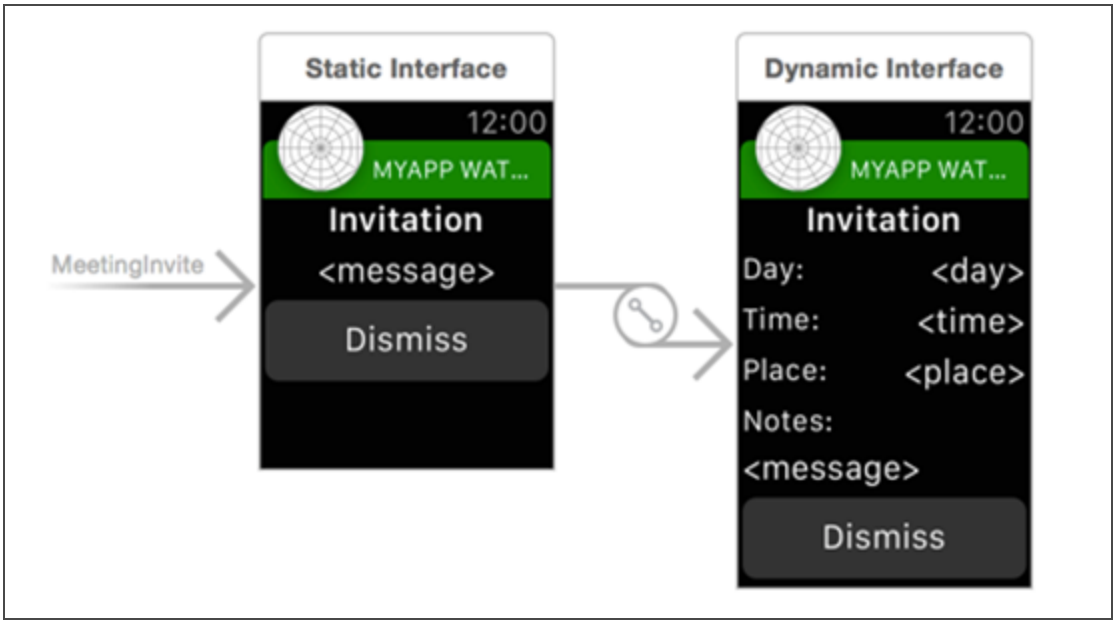
   A new dynamic notification is created with the same name as static notification and appended with the word **[Dynamic]**. For example, if a static notification is named **watchNotification**, its dynamic notification is named as **watchNotification [Dynamic]**.

   > ***Note:*** If a static notification is renamed, its dynamic notification is also renamed. The dynamic notification follows the naming convention **Name of the static notification + [Dynamic]**.

Difference between a Static Notification and Dynamic Notification

- A static notification displays the notification's alert message and any static images and text are configured during design time.

- A dynamic notification allows you to customize the appearance of your notification's content. This notification is optional when designing a notification scene.



Notification Properties

The notification properties contain the following tabs:

1. Look
2. Skin
3. Static Notification (available only on Static Notifications)
4. Action
5. Review

Look

On the Look tab, you set properties that control the way a notification appears on a watch.

ID

Denotes the name of a notification. When a notification is added to a watch, a unique name is assigned to the notification. You can rename a notification by entering a new name in the **ID** box.

> ***Note:*** You can also rename a notification from the Project Explorer by right-clicking a notification, and then clicking **Rename**.

Padding

Defines the space between the content of the notification and the notification boundaries. You can use this option to assign the top, left, right, and bottom distance between the notification content and the notification boundaries.

Property	Definition	Action
Top	Top padding	Move the slider to adjust the top padding of a notification.
Bottom	Bottom padding	Move the slider to adjust the bottom padding of a notification.
Left	Left padding	Move the slider to adjust the left padding of a notification.
Right	Right padding	Move the slider to adjust the right padding of a notification.

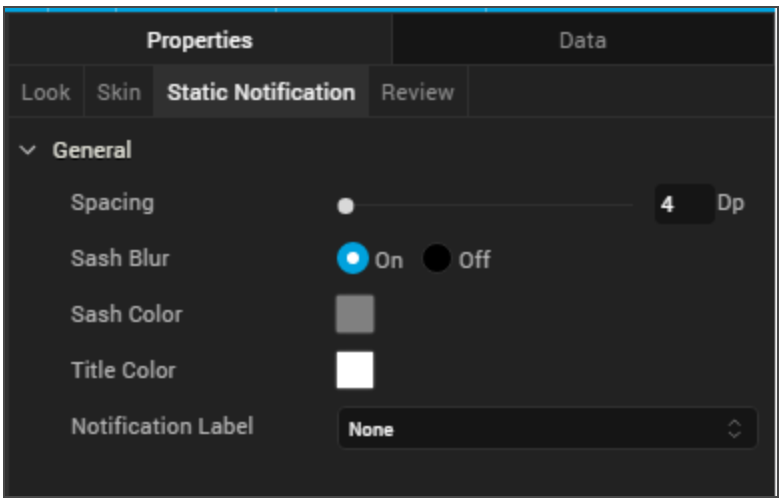
> ***Notes:***
> - Click  to apply uniform margins across all the padding boundaries of a notification.

Skin

A skin defines the background color that you apply to a **Normal** state of a notification. For more information on skins, see Understanding Skins and Themes.

Static Notification

This section list the properties that you can control for static notifications.



Spacing

This property defines the amount of space between the widgets in a notification. The spacing values are assigned in Dp format and can take values in the range of 0-100 Dp.

Sash Blur

The **Sash Blur** controls the background color of Sash area. You can either have the background color of a notification as the background color of the Sash area or choose a unique background color for the Sash area. To assign a background color, do one of the following:

- **On**: When you select **On** button next to **Sash Blur**, the Sash area background color is same as that of its notification's background color but with a blur.
- **Off**: When you choose **Off** button next to **Sash Blur**, the Sash area background color is controlled by **Sash Color**.
    - **Sash Color**: Click on the color palette icon next to **Sash Color** to open the color palette. Choose a required color for use as Sash area background color.

Title Color

Controls the font color of the title in the Sash area.

Click on the color palette icon next to **Title Color** to open the color palette. Choose a required color for use as font color for the title text.

Notification Label

A notification label controls the text that appear on a short notification. The **Notification Label** list displays all the **Label** widgets added to App Content pane. Click on a **Label** widget to use a notification label.

Action

On this tab, you define the events that are executed when an action is run. For a notification, you can run the following actions:

- onInit: This action allows you initialize your interface controller.
- onWillActivate: This action lets you know that your interface will soon be visible to the user. Use this method only to make small changes to your interface. For example, you might use this method to update a label based on new data.
- onDidDeactivate: This action allows you to clean up your interface and put it into a quiescent state. For example, use this method to invalidate timers and stop animations.
- onAwake: This action is called to let you know that the interface controller's contents are onscreen.

Glances

A glance is a supplemental way for a user to view important information from the app. When you click on a glance of an app, the relevant app appears on your screen. For example. a glance might inform you of an upcoming meeting. Each glance is divided into Upper Glance and Lower Glance. The Upper Glance covers the top half of a watch and the lower glance covers the bottom half of a watch. Kony Visualizer supports only one Glance for each app. It is not mandatory for an app to have a glance.

To add a Glance in Kony Visualizer, do the following:

1. Expand the **Watch** channel and hover on **Glance** to display a drop-down arrow.
2. Click the drop-down arrow and click **New Glance**. A new Glance is created. The system assigns a unique name to this Glance.
3. If you want, rename the notification by clicking the drop-down arrow next to the Glance and then clicking **Rename**. The Glance name becomes an editable text field. Rename the Glance and then press **Enter** on your keyboard.

> ***Note:*** When a Glance is added to a Watch, the Upper Glance and Lower Glance are created automatically.

The following widgets are supported on a Glance:

Group	Date	Image2	Label
Line	Timer	Map	

Look

On the Look tab, you set properties that control the way a Glance appears on an app. The following are the major properties you can condition on this tab:

- Specify a name for the Glance.
- Choose the platforms on which a widget should be rendered.

Action

On this tab, you define the events that are executed when an action is run. For a notification, you can run the following actions:

- onInit: This action allows you initialize your interface controller.

- onWillActivate: This action lets you know that your interface will soon be visible to the user. Use this method only to make small changes to your interface. For example, you might use this method to update a label based on new data.
- onDidDeactivate: This action allows you to clean up your interface and put it into a quiescent state. For example, use this method to invalidate timers and stop animations.
- onAwake: This action is called to let you know that the interface controller's contents are onscreen.

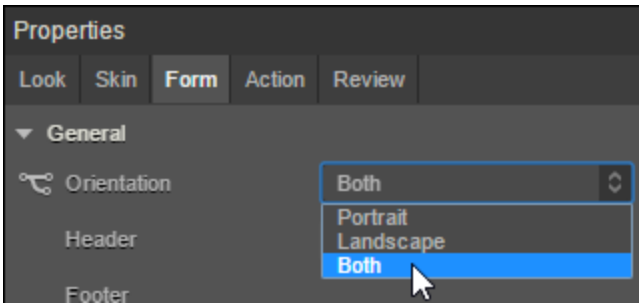
For more information on using these actions, see Add Actions.

Configure a Screen for Both Portrait and Landscape (Beta)

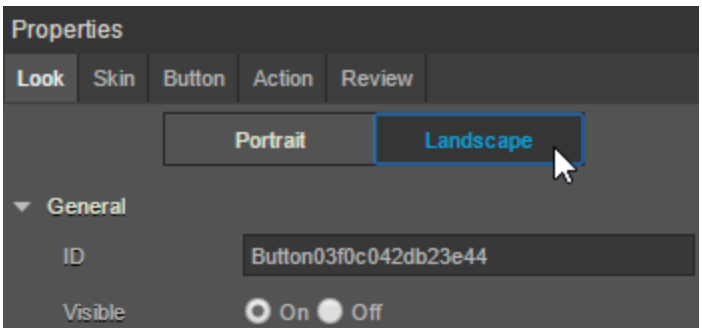
With Kony Visualizer, you can create a screen with properties for both portrait and landscape orientations without having to write any code. You set these properties at the form level. Once the app is published and is installed, when users rotate their device, a screen designed to support both orientations takes on the properties you set for that specific orientation.

To configure a screen for both portrait and landscape orientations, do the following:

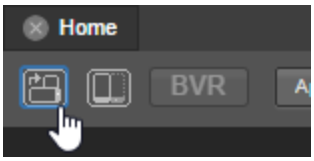
1. On the Project Explorer, click the **Project** tab.
2. Expand **Forms**, and then click the form whose orientation you want to configure.
3. On the **Form** tab of the Properties pane, from the **Orientation** drop-down list, click **Both**.



4. Click a child widget on the form, and then click its **Look** tab on the Properties pane. Two buttons are located at the top of the tab: Portrait and Landscape.



5. Click the orientation button that you want to configure the widget for.
6. Change the orientation of the screen on the Visualizer Canvas by clicking the **Orientation** toggle button. When you click this button, the orientation selected on the **Look** tab changes to match it.



7. Modify the widget's properties by dragging the widget into position on the Visualizer Canvas, and/or setting the properties in the Flex section of the **Look** tab.
8. To lock a particular Flex property so that it cannot be modified (whether by dragging, or changing its value in its property field), click its lock icon.

9. When you have set the Flex properties for one orientation, change the orientation of the screen on the Visualizer Canvas by clicking the **Orientation** toggle button, click the other orientation button on the **Look** tab, and then modify the widget's Flex properties in that orientation by dragging the widget into position on the Visualizer Canvas, and/or setting its Flex properties on the **Look** tab. Lock any properties whose values you don't want to be inadvertently changed.

10. Repeat steps 4 through 9 for each widget on the form that you want to customize for both portrait and landscape orientations.

Android Wear

Android Wear (now called Wear OS) is a version of Google's Android Operating system designed for smart watches and other wearables. Android Wear pairs with devices running on Android 4.3 or newer and iOS version 8.2 or newer phones. You can use Android Wear to perform functions such as calling, text messaging, and receiving notifications. Watch face styles include round and square. Kony Visualizer supports Sony SmartWatch 3 (320 x 320), Android Wear (400 x 400), and Android Wear (480 x 480).

Ensure that you configured Android Wear properties in Project Settings. For more information, see Android Wear Properties.

The following are the main features of an Android Wear:

- Forms
- Form Properties

Forms

A form is the starting point for any application and is the top most container. A form can contain any number of widgets but cannot contain another form.

Your Android Wear app can consist of multiple forms, and you can move from one form to another using a couple of different methods. You can use an action sequence, such as assigning the **Navigate to Form** action to a button so that, when it is clicked, it displays a different form.

Add an Android Wear Form

To add an Android Wear form to the Visualizer canvas, follow these steps:

1. Open Kony Visualizer.
2. From the **File** menu, select **New Project** or **Open** > <*your Project name*>.
   Your project opens.

3. In the Project tab, expand the **Wearables** folder.
   The Wearables folder expands displaying its contents.
4. Expand the **Android Wear** folder.
   The Android wear folder expands displaying its contents.
5. Click on the down-arrow next to **Forms**.
   A list appears.
6. Select **New Form** from the list.
   A new form is created with a default name.
7. If you want to rename the form, click on the down-arrow next to the form and then click **Rename**.
   Rename the form.

Mark as StartUp

Mark as StartUp functionality is used to make a form function as the app's start up form. The form that is a marked as start up is the first form to appear after the application is loaded.

To mark a form as the start up form, follow these steps:

1. In your Visualizer project, in the Project tab, expand the **Wearables** folder.
2. Expand the **Android Wear** folder.
   The Android wear folder expands displaying its contents.
3. Expand the **Forms** folder.
   All forms available in the folder appear.
4. From the list, select the form you want to choose as the startup form.
5. Click on the down-arrow next to the form and select **Mark as StartUp**.

> ***Note:*** By default, the first form is considered as the start up form until another form is specifically marked as start up.

Add a Widget to a Form

To add a widget to a form, do the following:

1. From the **Default Library** pane of Kony Visualizer, drag the required widget and drop it on a form on the Visualizer Canvas.
2. A new widget with a default name/ID is added to the form.
3. You may choose to rename the widget by changing its **ID** in the **Look** tab under the **Properties** pane of Kony Visualizer or right-click the widget name in the **Project** tab and then click **Rename**.
4. Save the form.

The following widgets are supported on an Android Wear form:

FlexContainer	FlexScrollContainer	Tab	TabPane
Button	Calendar	CheckBoxGroup	DataGrid
Image	Label	Listbox	RadioButtonGroup
RichText	Slider	TextArea	TextBox
Map	Phone	PickerView	Segment
Switch	Video		

Duplicate a Form

When you want to have a form that is similar to an existing form, you can duplicate the required form and reuse it.

To duplicate a form, do the following:

1. Click on the down-arrow next to the form you want to duplicate.
   A drop down menu appears.
2. From that drop down menu click on **Duplicate**.
3. A duplicate form with a default name is created.
4. If you want to rename the form, click on the down-arrow next to the form and then click **Rename**.
   Rename the form.

Delete a Form

When you no longer require a form, you can delete it.

To delete a form, do the following:

1. Click on the down-arrow next to the form that you want to delete.
   A drop down menu appears.
2. Click on **Delete** from that drop down.
   A deletion confirmation alert appears.
3. Click **OK** on the alert to delete the form.

Open the Assets Folder

To navigate to the location where the form assets are available on your computer, do the following:

1. Click on the down-arrow next to the form whose assets folder you want to view.
   A drop down menu appears.
2. Select **Resource Location** from that drop down.
   The Windows Explorer or Finder where the form assets are located appears.

Form Properties

You can customize the following properties of a form:

- Look
- Skin
- Form
- Action
- Review

Look

On the Look tab, you set properties that control the way a form appears on an app. The following are the major properties you can condition for a form:

Property	Definition	Action
ID	It is the unique identifier of the form.	When a form is added, the system assigns a unique ID (or name). You can rename the form ID if required.
Render	This property defines if a form should appear on a specific platform.	By default, a form is rendered for all the platforms. If you do not want to render a form for a specific platform, click the **Edit** button against the **Render** to open the **Render Platforms** dialog box. Clear the platform check box for which you do not want to render the form.
Friendly Name	The friendly name of the form is the name that the Navigation object is to be created for.	You can give any friendly name to the form here.

Property	Definition	Action
Controller	The code for the FormController object is created by the code generation tool for you. It communicates with both the models for the data sources and the viewmodels for the forms.	You can choose which FormControllers actions you want to link to this form. Click the **...** button against the **Controller** to open the **Switch Controller** dialog box.

Skin

On this tab, you define properties such as background color, borders, and shadows for a form. For an Android Wear form, you can define a skin its **Normal**, **Menu Normal**, and **Menu Focus** state. You can copy, paste, assign and duplicate any of these skins.

For more information on skin properties, see Understanding Skins and Themes.

Normal

Under the **Normal** tab the properties we have are:

Property	Definition	Action
General	The name of the skin(s) is stored here.	You can fork a skin and apply whichever one you choose
Background	The background of the form can be customized here.	You can choose the type of gradient, color, and preferred opacity of the background.
Border	This feature helps apply a border to the form.	You can choose the size, type of gradient, color, opacity, and style of the border.

Menu Normal

After we tick the **Enable** checkbox, all the properties can be edited.

Under the **Menu Normal** tab the properties we have are:

Property	Definition	Action
General	The name of the skin(s) is stored here.	You can fork a skin and apply whichever one you choose
Background	The background of the form can be customized here.	You can choose the type of gradient, color, and preferred opacity of the background.
Border	This feature helps apply a border to the form.	You can choose the size, type of gradient, color, opacity, and style of the border.

Menu Focus

After we tick the **Enable** check box all the properties can be edited.

Under the **Menu Focus** tab the properties we have are:

Property	Definition	Action
General	The name of the skin(s) is stored here.	You can fork a skin and apply whichever one you choose
Background	The background of the form can be customized here.	You can choose the type of gradient, color, and preferred opacity of the background.

Property	Definition	Action
Border	This feature helps apply a border to the form.	You can choose the size, type of gradient, color, opacity, and style of the border.

Form

On this tab, you set properties that are common for all the platforms and also, set properties that are unique to a platform.

General

Property	Definition	Action
Enable Idle Timeout	This property is used if the user has been idle on a form for a specific period of time.	You can select **On** or **Off** for this property using the radio buttons provided.
Transition: IN	Specifies how the form appears on the screen. You can choose a transition effect out of 11 available transitions.	Click the **Edit** button against the **Transition: IN** to open the **Transition: IN** dialog box. Select the preferred Transition event from the drop-down.
Transition: OUT	Specifies how the form moves away from the screen. You can choose a transition effect out of 11 available transitions.	Click the **Edit** button against the **Transition: OUT** to open the **Transition: OUT** dialog box. Select the preferred Transition event from the drop-down.
Title	Specifies the title of the Form.	Enter a tile for the Form.

Property	Definition	Action
Retain Scroll Position	Specifies if the Form must remember the scroll position when the user revisits the Form.	You can select **On** or **Off** for this property using the radio buttons provided.
Header Overlap	Specifies if the header must overlap the form. If this field is selected, the form scrolls behind the header background and a part of the header background is transparent.	You can select **On** or **Off** for this property using the radio buttons provided. Click the **...** button against the **Header Overlap** to open the **Header Overlap** dialog box and choose the platforms for which you want this property to work.
Footer Overlap	Specifies if the footer must overlap the form. If this field is selected, the form scrolls behind the footer background and a part of the footer background is transparent.	You can select **On** or **Off** for this property using the radio buttons provided. Click the **...** button against the **Footer Overlap** to open the **Footer Overlap** dialog box and choose the platforms for which you want this property to work.
Layout Type	Specifies if the arrangement of the widgets either in free form or vertical direction.	Choose any layout type from **Free Form** or **Flow Vertical**.
Enable Scrolling	Specifies whether the scrolling is enabled on the flex container or not.	You can select **On** or **Off** for this property using the radio buttons provided.
Bounces	Specifies whether the scroll view bounces past the edge of the content and back again.	You can select **On** or **Off** for this property using the radio buttons provided.

Property	Definition	Action
Horizontal Bounce	Specifies whether the scroll bounce is enabled or disabled in the horizontal direction.	You can select **On** or **Off** for this property using the radio buttons provided.
Vertical Bounce	Specifies whether the scroll bounce is enabled or disabled in the vertical direction.	You can select **On** or **Off** for this property using the radio buttons provided.
Vertical Scroll Indicator	Specifies whether the scroll indicator must be shown or not in the vertical direction.	You can select **On** or **Off** for this property using the radio buttons provided.
Paging	Specifies the whether the paging is enabled for the container. If this property is set to true, the scroll view stops on multiples of the scroll view's bounds when the user scrolls.	You can select **On** or **Off** for this property using the radio buttons provided. By default, it is **Off**.
Content Offset X	Specifies the X coordinate of the top-left scrollable region. When the value is set, the container scrolls even if the scrolling is disabled. This will always return the value that you set, but never reflects the actual computed offset.	You can specify the coordinates using **Dp**, **Px**, and **%** units. The default unit is **Dp**.
Content Offset Y	Specifies the Y coordinate of the top-left scrollable region. When the value is set, the container scrolls even if the scrolling is disabled. This will always return the value that you set, but never reflects the actual computed offset.	You can specify the coordinates using **Dp**, **Px**, and **%** units. The default unit is **Dp**.

Property	Definition	Action
Content Size Width	Specifies the width of the flex container to accommodate all the widgets placed in it. This will returns the values that you set, but never reflects the actual computed content size.	You can specify the coordinates using **Dp**, **Px**, and **%** units. The default unit is **Dp**.
Content Size Height	Specifies the height of the container to accommodate all the widgets placed in it. This will returns the values that you set, but never reflects the actual computed content size.	You can specify the coordinates using **Dp**, **Px**, and **%** units. The default unit is **Dp**.
Snap to Grid	When turned On, this feature sets default positions that a widget will move on drag (or any other) action by a user into the Android Wear layout automatically.	You can select **On** or **Off** for this property using the radio buttons provided.
Snap Grid Size	When turned On, this feature sets default positions that a widget will move on drag (or any other) action by a user. you can specify the grid size using this feature.	You can specify the number of the size.
Default Unit	Specifies the unit that will be used as the default unit value throughout the form properties.	You can choose between **Dp**, **Px**, and **%**.

Android Wear OS

Property	Definition	Action
Soft Input Mode	This property defines how the main Form interacts with the window containing the on-screen soft keyboard. It determines the adjustments made to the Form whether it is resized smaller to make room for the soft keyboard or whether its contents pan to make the current focus visible when part of the Form is covered by the soft keyboard.	You can choose between **Adjust-Pan** and **Adjust-Resize**. **FORM_ADJUST_RESIZE**: Specifies the form is resized and when you click or start typing within the widget which requires an input, the form scrolls up and the widget which requires an input is not overlapped by the keypad or footer. **FORM_ADJUST_PAN**: Specifies the widget which requires an input is placed at the bottom of the popup is overlapped by the keypad. The main Form is not resized to make room for the soft keyboard. Rather, the contents of the Form are automatically panned so that the current focus is never obscured by the keyboard and users can always see what they are typing. This is generally less desirable than resizing, because the user may need to close the soft keyboard to get at and interact with obscured parts of the Form.
Dismiss on Edge Swipe	Using this feature, you can enable or disable dismissing forms by swiping at the edge.	You can turn it **On** or **Off**.

Action

General

On this tab, you define the events that are executed when an action is run. For a notification, you can run the following actions:

Property	Definition	Action
init	This action allows you initialize your interface controller.	Click **Edit** to add actions.
preShow	This action allows you to customize an event which is triggered every time a form is to be displayed.	Click **Edit** to add actions.
postShow	This action allows you to customize an event which is triggered every time after a form is displayed.	Click **Edit** to add actions.
onHide	This action allows you to customize an event which is triggered when a form goes completely out of view.	Click **Edit** to add actions.
onDestroy	This action allows you to customize an event which is triggered when the Form is destroyed.	Click **Edit** to add actions.

Property	Definition	Action
onTouchStart	This action allows you to customize an event which is triggered once the user touches the touch surface.	Click **Edit** to add actions.
onTouchMove	This action allows you to customize an event which is triggered once the user's touch moves on the touch surface.	Click **Edit** to add actions.
onTouchEnd	This action allows you to customize an event which is triggered once the user's touch is released from the touch surface.	Click **Edit** to add actions.

Android Wear OS

Property	Definition	Action
onDeviceMenu	This action allows you to customize an event which is triggered once the user accesses the device menu.	Click **Edit** to add actions.
onDeviceBack	This action allows you to customize an event which is triggered once the user uses the back button on the device menu.	Click **Edit** to add actions.

Property	Definition	Action
onOrientationChange	This action allows you to customize an event which is triggered when there is a change in orientation of the form.	Click **Edit** to add actions.
onActionBarBack	This action allows you to customize an event which is triggered once the user uses the back button on the action bar.	It is enabled only when the onActionBarBack callback is registered on form and the showActionBarIcon is set to true. Click **Edit** to add actions.

For more information on using these actions, see the topic, Add Actions.

Review

This tab can be used to add Notes and Comments to the selected form.

You can search for the notes written about a form by typing its name in the **Search** box.

You can expand, contract, add, and delete notes.

Populate Screens with Widgets

Widgets are the building blocks of a screen (form) in a digital application, and each one has a specific purpose, such as user interaction or animation. Kony Visualizer provides you with built-in widgets that help you achieve your required functionality. You can configure every widget based on your needs.

This topic covers widgets for the Mobile, Tablet, and Desktop channels. For information regarding Watch widgets, see Add Watch Widgets.

Kony Visualizer provides three types of widgets: Container, Basic, and Advanced.

Click a topic for more information.

Access widgets from the Widget Tab

The Three Types of Widgets

Populate Forms with Widgets

Occupy the Parent Widget's Entire Area

Convert Flex Container Widgets

Resize and reposition widgets

Group Widgets into a Container Widget

Copy the Widget Path

Parent or unparent a widget

Set the Default Widget Style

Look Properties for Widgets

Access Widgets from the Widget Tab

The widgets are located on the Widget tab of the Kony Library pane. To add widgets to a form, simply drag and drop them into place. As you do so, alignment guides display to guide your positioning of the widget.

The Three Types of Widgets

Kony Visualizer provides you with three types of widgets.

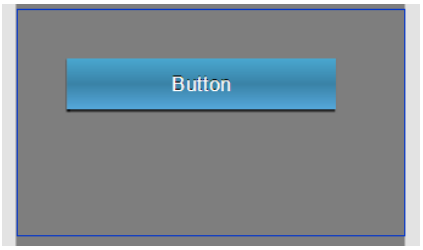
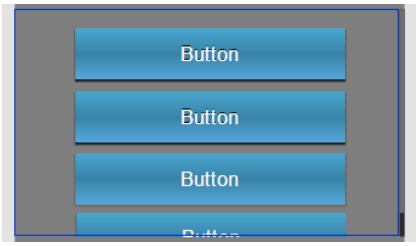
Container Widgets

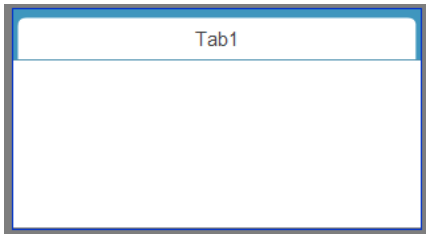
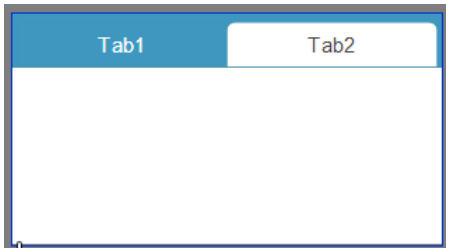
Basic Widgets

Advanced Widgets

Container Widgets

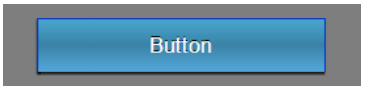
Container Widgets act as containers to group other widgets. Using the container widgets you can group two or more widgets so that you can position them as a unit. Following are the Container Widgets available in Kony Visualizer:

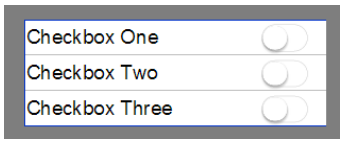
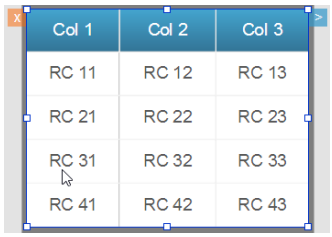
Widget	Description	Image
FlexContainer	A Container Widget with flexible properties. Flexible properties allow you to place widgets anywhere in the form and configure alignment properties such as Left, Right, Top, and Bottom. You can set the properties of the FlexContainer in three units of measurement: dp(device independent pixels, dip = dp); px (actual picture elements [i.e. pixels] on screen); and percentage. If you want, you can convert a FlexContainer to a FlexScrollContainer. For more information, see Convert Flex Container Widgets.	 A FlexContainer Widget with a Button Widget inside.
FlexScrollContainer	A Container Widget that works as a FlexContainer but scrolls horizontally or vertically. If you want, you can convert a FlexScrollContainer to a FlexContainer. For more information, see Convert Flex Container Widgets.	 A FlexScrollContainer Widget with button widgets inside, arranged vertically with the Scroll Direction property.

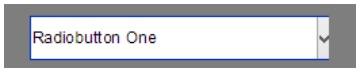
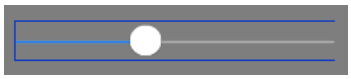
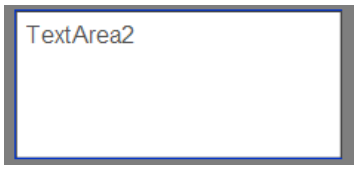
Widget	Description	Image
TabPane	A Container Widget in which you can group widgets in tabs. Each tab can have its own set of widgets, and you can add a new tab by using the Tab Widget.	
Tab	A Container Widget to be used only with the TabPane Widget. Drag-and-drop this widget onto a TabPane Widget to create a new tab.	

Basic Widgets

Basic Widgets help you build the user interface in your application. Along with designing the application, you can also configure events to these widgets or write code snippets to achieve a functionality.

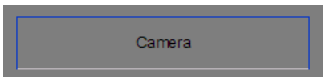
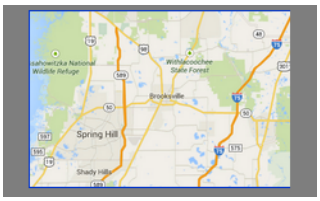
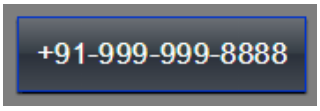
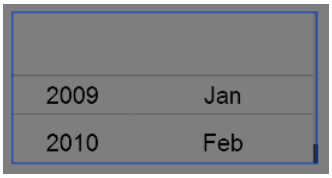
Widget	Description	Image
Button	A Button Widget enables you to provide input to an application or to trigger an event. For example, you can navigate to a form, interact with a dialog box, or confirm an action.	

Widget	Description	Image
Calendar	Calendar Widget allows you to select a date from a graphical calendar. The Calendar Widget appears as a label with a small calendar icon (icon does not appear on Mobile Web platforms) and displays the date or the date format you specified. Click the Calendar Widget to interact.	
CheckBoxGroup	The CheckBoxGroup Widget allows you to make selections from a group of check boxes. When you select a check box, a check mark appears inside the check box, to indicate the selection.	
DataGrid	The DataGrid Widget allows you to present a collection of data in rows and columns (tabular format).	
Image2	The Image widget is a non-interactive widget that you can use to display a graphic (local or remote). You can use an Image Widget in scenarios such as displaying your company's logo, displaying a snapshot, and providing an illustration.	
Label	The Label Widget displays non-editable text on the Form and is non-interactive.	

Widget	Description	Image
ListBox	The ListBox Widget displays a list of items such as a drop-down box and allows you to select a single item at a time.	
RadioButtonGroup	The RadioButtonGroup is a widget that allows you to define a set of radio buttons, and a user can choose one of the buttons as an option.	
RichText	The RichText Widget displays non-editable and formatted text on the Form. HTML formatting tags in the RichText Widget to display text with styles (bold, underlined and so on), links, and images.	
Slider	The Slider Widget allows you to select a value from a defined range of values by moving the thumb (an indicator) in a horizontal direction.	
TextArea2	The TextArea2 Widget provides a means by which the user can enter text.	
TextBox2	The TextBox2 Widget is used to capture input from the user.	

Advanced Widgets

Advanced widgets provide you the capability to achieve the most commonly used functionality in your application. These Widgets are developed by Kony. Also, the option to configure properties of the widgets is provided.

Widget	Description	Image
Browser	Use the Browser Widget to display HTML content of your application without navigating away from the application or opening the native browser.	
Cordova Browser	Use the Cordova Browser Widget to make the content of a Cordova application accessible to the user. For more information, see Create Cordova Applications.	
Camera	Uses the device's native camera and its functionality for image and video capture.	
Map	A Map Widget displays locations defined by latitude and longitude on an on-screen map.	
Phone	Accesses the native phone dialer and initiate a voice call to the number that appears on the widget.	
PickerView	A PickerView Widget uses either a spinning wheel control or flat view picker to display multiple sets of values and allows you to select a combination of values.	
Segment2	A Segment2 Widget consists of multiple segments (rows or records), and each segment (row or record) can have multiple child widgets.	
Switch	The Switch Widget presents two mutually exclusive choices or states.	

Widget	Description	Image
Video	Used for displaying a video as referenced by a URL or the video user upload.	

Populate Forms with Widgets

Using the following options you can populate a form with widgets:

- **Kony Library**: You can populate a form with widgets by dragging a widget from the Widget Library and dropping the widget on a form.

  > ***Note:*** You cannot place a **Tab** widget directly on a form. You can drop a Tab widget only inside a **TabPane** container widget.

  > ***Important:*** You cannot create new VBox(deprecated) forms, but you can import VBox forms from your previous projects into your latest projects and continue to work with them seamlessly.
  > When you create a new template in a project in Kony Visualizer from V8 SP2 onwards, the top-level FlexContainer automatically is created along with your template. You can delete the FlexContainer and add a VBox(deprecated) form if needed.

- **User-defined Collections**: From the user-defined collections, you can drag and drop a widget.
- **Duplicate**: If you have a widget that you want to duplicate within the same form, use the Duplicate feature. To duplicate a widget, right-click a widget and click **Duplicate**. The duplicated widget appears offset from the prior widget by a horizontal and vertical offset of 10 units, making it easy to distinguish from the prior widget.
- **Copy and Paste**: If you want to reuse a widget on another form or channel, you can copy and paste it. To copy and paste a widget, right-click a widget, click **Copy**, and then click **Paste**.

> ***Note:*** The Duplicate feature and Copy and Paste feature differ in that you can duplicate a widget within the same form whereas you can copy a widget and then paste the widget within the same form, a different form, or a different channel.

Occupy the Parent Widget's Entire Area

You can fit a widget to its parent such that it occupies the parent widget's entire area.

To fit a widget to its parent, do the following:

- On the Visualizer Canvas, right-click the widget you want to fit to its parent, and then click **Fit to Parent**. The Top and Left properties of the widget are set to 0, and the Height and Width properties are set to 100%, causing the widget to occupy the parent widget's entire area.
- This option is available only on Flex forms, and when the parent widget's property **Layout Type** is **Free Form**.
- This option not available for the Tab widget.

Convert Flex Container Widgets

After adding a flex container to a form and populating it with widgets, you might decide that your app would be better served by having a flex scroll container—a container that allows the user to scroll vertically or horizontally. Or you might find yourself in the opposite situation. You may have a flex scroll container that you want to convert to a non-scrolling flex container. You can convert any flex container to a flex scroll container—and vice versa—with a couple of clicks.

To convert a flex container widget to its alternate, do the following:

1. On the **Project** tab of the Project Explorer, locate and click the flex container or flex scroll container that you want to convert. The container displays on the Visualizer Canvas and is now the container with focus.
2. On the Visualizer Canvas, right-click the instance, point to **Convert To**, and then select the available alternate type of container available.

Resize and Reposition Widgets

You can re-size and reposition widgets, either individually or as part of a multiple selection, in two ways: by dragging their boundary handles, or by changing the Flex properties located on the Look tab of the Properties pane.

To re size and reposition widgets, do the following:

1. Select the widget or widgets. For more information, see Select Multiple Items.
2. To reposition, hover over the widget (or any one of the widgets if multiple widgets are selected), and then drag the widget(s) to the desired position. Alternately, on the Look tab of the Properties pane, change the Flex properties for Left or Right, and Top or Bottom.
3. To re size, hover over a boundary handle and then drag the widget(s) to the desired size. Depending on whether you drag horizontally or vertically, the widget(s) re size in one-unit increments relative to their original dimensions. Alternately, if you want the widgets to have the same dimensions, on the Look tab of the Properties pane, change the Flex properties for Width and Height.
4. On the Visualizer Canvas, click the context menu arrow of one of the selected widgets, point to **Group Into Flex**, and then select either **Flex Container** (for a non-scrollable container) or **Flex Scroll Container** (for a container that can be scrolled vertically and horizontally).

The Flex properties on the Look tab of the Properties pane are as follows:

- **Left** - Determines the left edge of the widget and measured from the left boundary of the parent container.
- **Top** - Determines the top edge of the widget and measured from the top boundary of the parent container.
- **Width** - Determines the width of the widget and measured along the x-axis.
- **Min Width** - Specifies the minimum width of the widget. This property is considered only when width property is not specified.

- **Min Height** - Specifies the minimum height of the widget. This property is considered only when the height property is not specified.
- **Center X** - Determines the center of the widget measured from the left boundary of the parent container.
- **Z Index** - Specifies the stack order of the widgets. A widget with higher zIndex is in front of the one with lower zIndex.
- **Right** - Determines the right edge of the widget and it is measured from the right boundary of the parent container.
- **Bottom** - Determines the bottom edge of the widget and is measured from the bottom boundary of the parent container.
- **Height** - Determines the height of the widget and is measured along the y-axis (height of the parent)
- **Max Width** - Specifies the maximum width of the widget. This property is considered only when the width property is not specified.
- **Max Height** - Specifies the minimum height of the widget. This property is considered only when the height property is not specified.
- **Center Y** - Determines the center of the widget measured from top boundary of the parent container.

Group Widgets into a Container Widget

With Kony Visualizer, you can group widgets together under a single container widget so that you can manipulate them as a single unit, including moving and duplicating them.

To group widgets into a container widget, do the following:

1. Select the widgets you want to group together. For more information, see Select Multiple Items.
2. On the Visualizer Canvas, click the context menu arrow of one of the selected widgets, point to **Group Into Flex**, and then select either **Flex Container** (for a non-scrollable container) or **Flex Scroll Container** (for a container that can be scrolled vertically and horizontally).

Copy the Widget Path

If you need to make reference to a particular widget in a module, you can copy the widget path and paste it into your code rather than having to type it out. Widget paths follow the hierarchy of the forms and containers that they appear in. A common basic widget path is organized in the following way:

```
FormName.ContainerName.WidgetName
```

To copy the path of a widget, do the following:

1. In the Project Explorer, on the Project tab, navigate to the widget whose path you want to copy.
2. Click the context menu arrow of the widget, and then click **Copy Widget Path**.

   You can now paste the widget path into your code module.

Parent or Unparent a Widget

You can make a widget into a parent and then add widgets under it so that they are grouped together.

To parent or unparent widgets, do the following:

1. Drag and drop a widget onto your form.
2. Select the widget from the Project Explorer.
3. From the Project Explorer, drag and drop the widget you want onto the form. On the Project Explorer the widget you dropped is indented under the previously selected widget.
4. To unparent a widget, right-click the child widget and click **Unparent**. The widget's link is removed from the parent widget and linked one-level up in the project hierarchy.

Set the Default Widget Style

For every type of widget, Kony Visualizer has its own generic default style, but you may want to set your own default widget style to simplify the process of formatting new widgets.

To set the default widget style, do the following:

1. On the Visualizer Canvas, select the widget that you want to use as the default style for any new widgets of that type that you create.
2. Click the widget's context menu arrow, and then select **Set Default Widget Style**. Any new widgets of that type that you create are automatically formatted with the new characteristics.
3. To reset a type of widget to be formatted according to Kony Visualizer's default formatting for that widget type, click the widget's context menu arrow, and then select **Reset to Visualizer Default**.

Look Property for Widgets

For information on common properties of widgets, see Widget Look Properties.

FlexContainer

Use a FlexContainer widget to create a layout area on a form that can contain other widgets. If you want the layout area to include scroll bars, use a FlexScrollContainer widget. For more information, see Convert Flex Container Widgets.

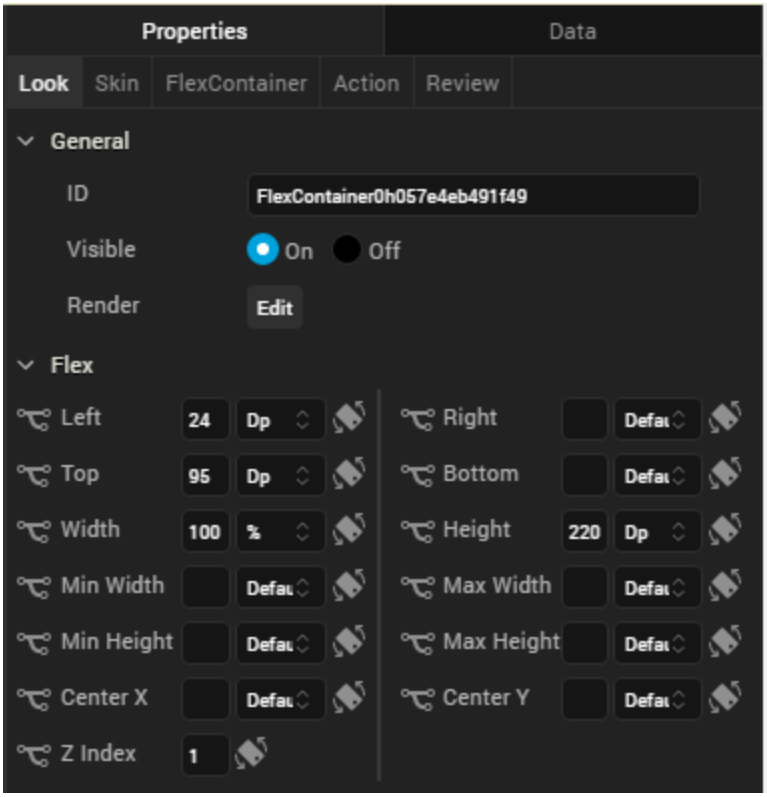
To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.

- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Auto Growth of a FlexContainer Widget

You can specify that a FlexContainer widget dynamically adjusts its height based on the height of its child widgets by setting the Height property on the Look tab to *Preferred*. The height of the FlexContainer then adjusts when the height of any child widgets changes.

Following are important considerations for a FlexContainer that is set to adjust dynamically:

- A FlexContainer will not dynamically adjust it has a static height; for example, if Top and Bottom values are set.
- A FlexContainer that contains other dynamically-adjusting widgets adjusts it's height after the child widgets have adjusted. For example, if the FlexContainer contains another FlexContainer

that is set to grow dynamically and a Segment2 widget with the Auto Grow property enabled, the height of the parent FlexContainer dynamically adjusts after the heights of the child FlexContainer and Segment2 widgets adjust.

The following table summarizes how to calculate the height of child widgets:

Height	Center Y	Top	Bottom	Child Height
Yes	Yes	Yes	Yes	centerY + height/2 + bottom
Yes	Yes	No	Yes	centerY + height/2 + bottom
Yes	Yes	Yes	No	centerY+ height/2
Yes	Yes	No	No	centerY+ height/2
No	Yes	Yes	Yes	(centerY – top) * 2 + top + bottom
No	Yes	Yes	No	(centerY – top) * 2 + top
No	Yes	No	Yes	centerY + bottom + cph/2
No	Yes	No	No	centerY + CPH/2
Yes	No	Yes	Yes	top + height + bottom
No	No	Yes	Yes	top + CPH + bottom
Yes	No	Yes	No	top + height
No	No	Yes	No	top + CPH
Yes	No	No	Yes	height + bottom
No	No	No	Yes	cph + bottom
Yes	No	No	No	height
No	No	No	No	CPH

Note: The Computed Preferred Height (CPH) is determined by the calculated height for the content-driven widget, the default value returned from the configuration file, and the calculated height of the Auto Grow Segment or FlexContainer.

Limitations:

- If top or bottom values of child widgets are not provided, then zero values are used when determining the height of a FlexContainer.
- If the height of a widget changes when the widget's skin state changes (for example, from normal to focus), the height of a FlexContainer will not grow dynamically.
- If a widget's top value is negative, then the widget will be clipped in case of **Free Form** and overlapped on the previous widget in **Flow Vertical**.
- A FlexContainer will not grow dynamically when placed inside an HBox or a VBox.

  > ***Note:*** You can place a FlexContainer inside an HBox or VBox only while creating a Map or Gridcalendars template.

- If clip bounds are disabled for a child FlexContainer, the children of the child FlexContainer that are out of the bounds do not affect the height of the parent FlexContainer. Only the heights of direct children of the parent FlexContainer are be used to determine its height.
- A Segment template created using a FlexContainer will dynamically grow only when the Segment2 view type is **Table View**.
- FlexContainer dynamic growth is not supported on a **Tab Widget**.
- In the Windows Platform, the Segment2 Auto Grow property is not supported if the Segment 2 widget is placed inside a FlexContainer.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For a FlexContainer widget, you can apply a skin and its associated properties for the following states:

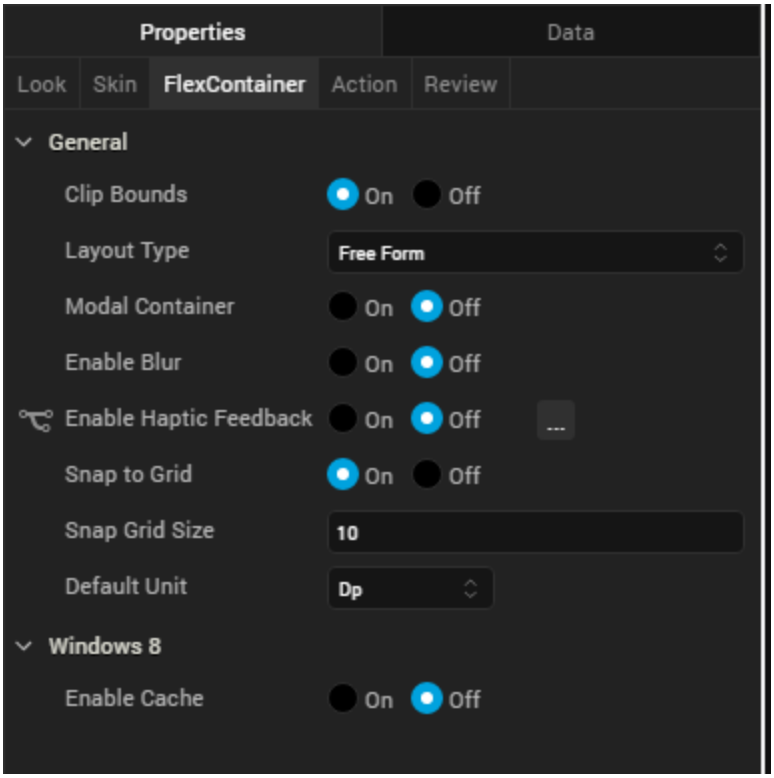
Skin	Definition
Normal	The default skin of the widget.

Skin	Definition
Focus	The skin applied when the widget has the focus.
Blocked UI	The skin applied to block the interface until the action in progress (for example, a service call) completes. ***Note:*** Blocked UI is available only for SPA platforms.

For more information about applying skins, see Understanding Skins and Themes.

FlexContainer Properties

FlexContainer properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Clip Bounds

Specifies whether to clip child widgets when they go out of boundaries.

Layout Type

Specifies whether the arrangement of widgets in the FlexContainer flows horizontally, vertically, or in both directions.

Default: Vertical

Child Widget Align

Specifies whether the arrangement of widgets in the child widget of the FlexContainer flows horizontally, vertically, or in both directions.

Default: Right to Left.

Visualizer generates **reverseLayoutDirection** property with **true** for Right to Left and Bottom to Top. Visualizer generates this property with **false** for Left to Right and Top to Bottom.

Snap to Grid

Specifies whether the widget aligns to the nearest intersection of lines in the grid, or other widgets.

Snap Grid Size

Specifies the grid size. This option is available only when **Snap to Grid** is enabled.

Default Unit

Specifies the default unit used for interpretation of numbers with no qualifiers when passed to layout properties.

Following are the options:

- dp: Specifies the values in terms of device independent pixels.
- px: Specifies the values in terms of device hardware pixels.
- %: Specifies the values in percentage relative to the parent dimensions.

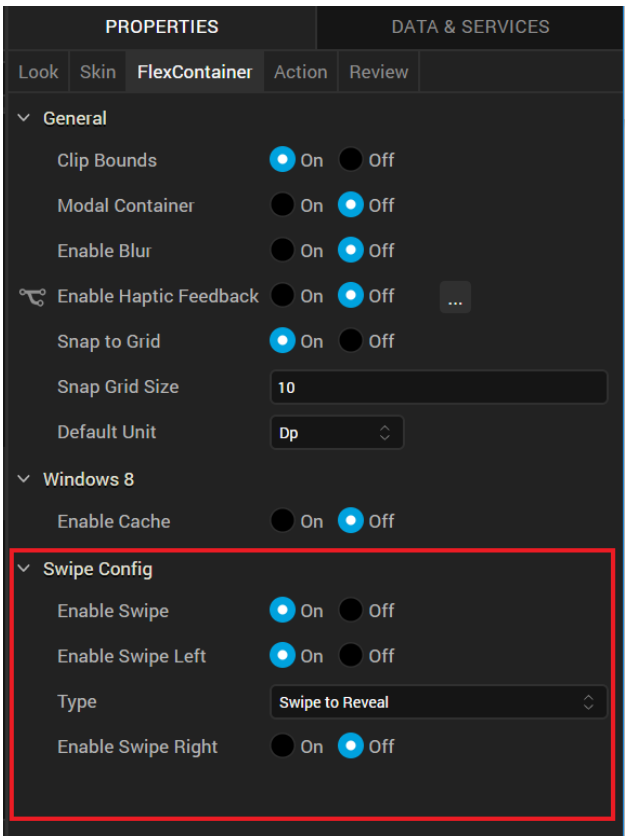
Swipe Config

From Kony Visualizer V9 onwards, you can configure the swipe functionality for a Segment widget. The Swipe config settings enable users to dismiss a row or to reveal certain actions when they swipe a row to the left or right.

> ***Note:*** The Swipe Config setting is applicable only on the Flex Container widget that is present inside a segment widget.

To enable the swipe left or swipe right functionality on the rows of a Segment, follow these steps:

1. In Visualizer, from the **Project** explorer, navigate to a <channel> > **Forms**.
2. Right-click **Forms** and then click **New Form**.
3. Drag and drop a **Segment** widget onto the form, and then rename the Segment.
4. On the Templates tab, go to **Mobile** > **Segment**.
5. Right-click **Segment**, select **New Template**, and then rename the template as **Temp1**.
6. On the **Project** Explorer, click the Segment widget.
7. On the **Properties** panel, click the **Segment** tab. For **Row Template**, select **Temp1**.
8. Go to the **Templates** Explorer, and then select the Flex Container under **Temp1**.
9. On the **Properties** panel, click the **FlexContainer** tab. The **Swipe Config** section appears.
10. Under the **Swipe Config** section, select **On** for **Enable Swipe**. The **Enable Swipe Left** and **Enable Swipe Right** settings appear.



11. To enable a swipe left or swipe right functionality, select **On** for **Enable Swipe Left** or **Enable Swipe Right** respectively. After the selection, right/left FlexContainers are added under the Swipe Container.

12. Then, select the actions that the rows of a Segment must perform on swiping:

    - **Swipe to Reveal:** Enables you to display the left or right swipe container when you swipe a row to the left or right. The right FlexContainer is visible when you swipe the row to the left and vice-versa.

    - **Swipe to Dismiss:** Enables you to perform an action on a row when you swipe the row to the left or right.

13. From the Templates Explorer, select the FlexContainer under the **Temp1** template.
    If you select **Swipe to Dismiss**, the **onSwipeLeft** or **onSwipeRight** actions are available and can be configured under the **Action** tab of the Properties panel.

> ***Note:*** If you have not used a row template but dragged the widgets directly onto the Segment, the **onSwipeLeft**or **onSwipeRight** properties are available on the **Action** tab of the Segment.

14. Click **Edit** beside the **onSwipeLeft** or **onSwipeRight** action. In the Action Editor, add the actions that the row must perform on swiping to left or right.
15. To edit the Swipe Config settings, go to the Properties panel, and then click the **Segment** tab. Under **General** settings, click **Edit** beside the **Swipe Config** option.

The swipe functionality can be configured to a row of a segment through code by using the widgetSwipeMove property.

> ***Note:*** On Android and SPA/Desktop Web platforms, if you reset the data of a Segment widget, make sure that you include entries for the widgets that are inside the swipe container. Otherwise, the widgets are not rendered at the run time.

Actions

Actions define what happens when an event occurs. On a FlexContainer widget, you can run an action when the following events occur:

- **onTouchStart:** The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- **onTouchMove:** The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- **onTouchEnd:** The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.
- **onClick:** The action is triggered when the user clicks on the widget.
- **onSwipeLeft:** This action is triggered when you swipe a row of a Segment widget to the left.

This action is available only when the **enable Swipe** and **enable Swipe Left** options are enabled and the **Type** is set to **Swipe to Dismiss** for the Segment template.

- **onSwipeRight:** The action is triggered when you swipe a row of a Segment widget to the right. This action is available only when the **enable Swipe** and **enable Swipe Right** options are enabled and the **Type** is set to **Swipe to Dismiss** for the Segment template.

For more information, see Add Actions.

FlexScrollContainer

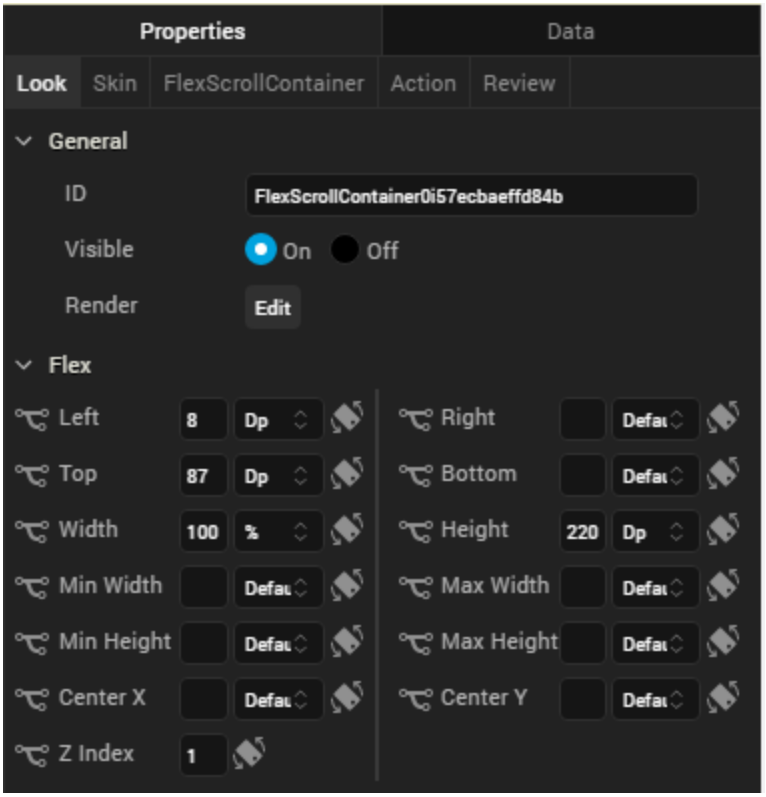
Use a FlexScrollContainer widget to create is a scrollable layout area on a form that can contain other widgets. It can contain any number of widgets within it. If you do not want the layout area to include scroll bars, use a FlexContainer widget. For more information, see Convert Flex Container Widgets.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For a FlexScrollContainer, you can apply a skin and its associated properties for the following states:

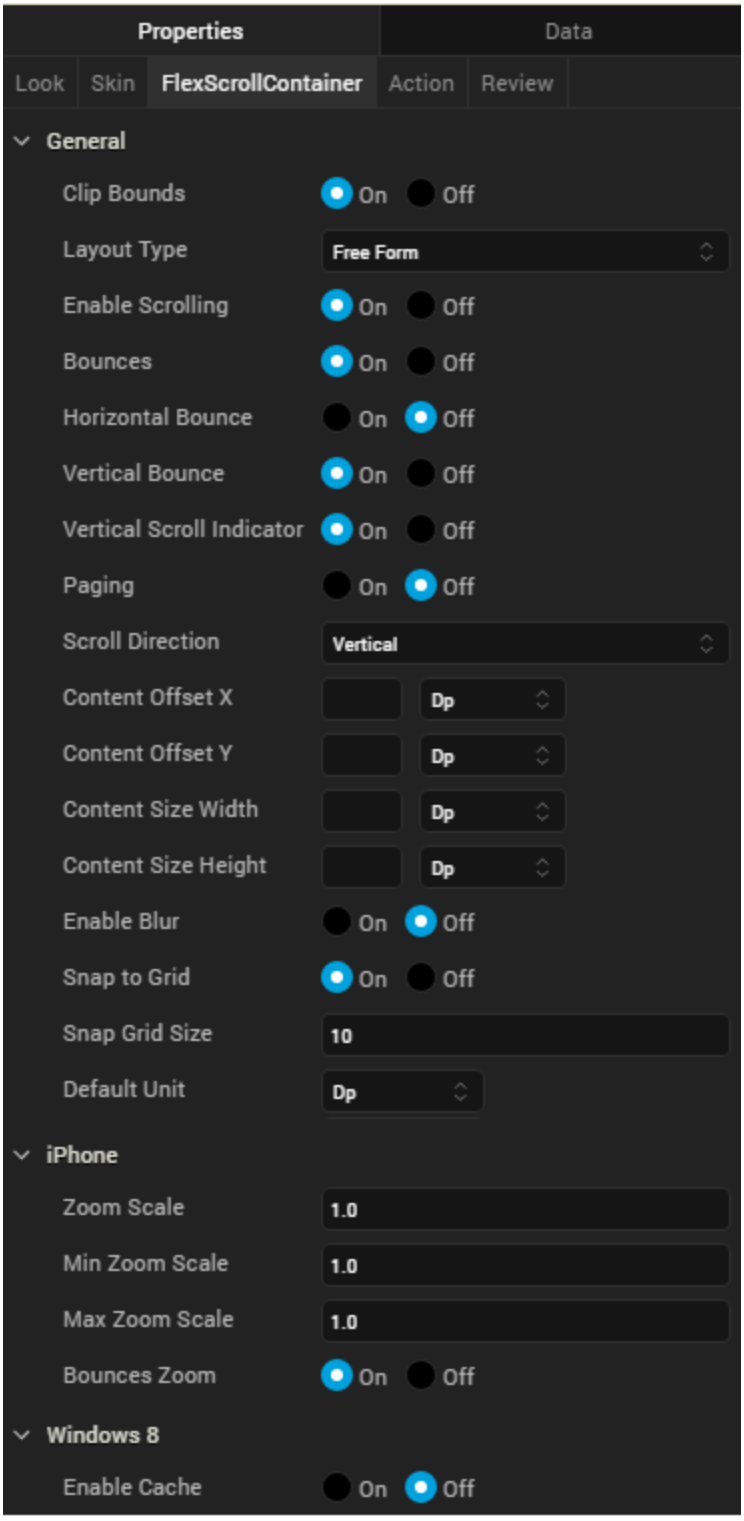
Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the widget has the focus.

Skin	Definition
Blocked UI	The skin applied to block the interface until the action in progress (for example, a service call) completes. ***Note:*** Blocked UI is available only for SPA platforms.

For more information about applying skins, see Understanding Skins and Themes.

FlexScrollContainer Properties

FlexScrollContainer properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Clip Bounds

For Kony Visualizer Version 7.3 and later, specifies whether child widgets are clipped to the bounds of the FlexScrollContainer.

Layout Type

Specifies whether the widget content flows horizontally, vertically, or in both directions.

Default: Vertical

Following are the options:

- Free Form: Navigation occurs in both directions.
- Flow Vertical: Navigation is vertical.
- Flow Horizontal: Navigation is horizontal.

Enable Scrolling

Specifies whether scrolling is enabled.

Bounces

Specifies whether scroll bounce is enabled.

Horizontal Bounce

Specifies whether scroll bounce is enabled in the horizontal direction.

Default: Off

Vertical Bounce

Specifies whether scroll bounce is enabled in the vertical direction.

Default: On

Horizontal Scroll Indicator

Specifies whether the horizontal scroll indicator is enabled.

> ***Note:*** Depending on the platform, scroll indicators may be visible only during scrolling.

Vertical Scroll Indicator

Specifies whether the vertical scroll indicator is enabled.

> ***Note:*** Depending on the platform, scroll indicators may be visible only during scrolling.

Paging

Specifies the whether paging is enabled for scrolling.

Scroll Direction

Specifies the scroll direction, either horizontal, vertical, or both horizontal and vertical.

Default: Vertical

Content Offset X

Specifies the X coordinate of the top-left of the scrollable region. When content offset values are set, the FlexScrollContainer scrolls even if scrolling is disabled. The value does not reflect the actual computed offset.

Content Offset Y

Specifies the Y coordinate of the top-left of the scrollable region. When content offset values are set, the FlexScrollContainer scrolls even if scrolling is disabled. The value does not reflect the actual computed offset.

Content Size Width

Specifies the FlexScrollContainer width to accommodate all the widgets placed in it. The value does not reflect the actual computed content size.

Content Size Height

Specifies the FlexScrollContainer height to accommodate all the widgets placed in it. The value does not reflect the actual computed content size.

Snap to Grid

Specifies whether the widget aligns to the nearest intersection of lines in the grid, or other widgets.

Snap Grid Size

Specifies the grid size. This option is available only when **Snap to Grid** is enabled.

Default Unit

Specifies the default unit to be used for interpretation of numbers with no qualifiers when passed to layout properties.

Following are the options:

- dp: Specifies the values in terms of device independent pixels.
- px: Specifies the values in terms of device hardware pixels.
- %: Specifies the values in percentage relative to the parent dimensions.

Zoom Scale

Specifies the scale factor applied to the FlexScrollContainer content.

Default:1

> ***Note:*** This property is specific to the iOS platform.

Min Zoom Scale

Specifies the minimum zoom scale factor that can be applied to the FlexScrollContainer.

Default: 1

> ***Note:*** This property is specific to the iOS platform.

Max Zoom Scale

Specifies the maximum zoom scale factor that can be applied to the FlexScrollContainer.

Default: 1

> ***Note:*** This property is specific to the iOS platform.

Bounces Zoom

Specifies whether the scroll view animates the content scaling when the scaling exceeds the maximum or minimum limits. If Bounces Zoom is set to On and zooming exceeds either the minimum or maximum limits for scaling, the scroll view temporarily animates the content scaling just past the limits before returning to them. If Bounces Zoom is set to Off, zooming stops immediately when it reaches scaling limits.

> ***Note:*** This property is specific to the iOS platform.

Actions

Actions define what happens when an event occurs. On a FlexScrollContainer widget, you can run an action when the following events occur:

- onScrollStart: The action is triggered when when the user starts scrolling the content. This event occurs asynchronously.
- onScrollTouchReleased: The action is triggered when the user's touch is released from the touch surface. This event occurs asynchronously.
- onScrolling: The action is triggered when scrolling is in progress. This event occurs asynchronously.
- onDecelerationStarted: The action is triggered when the user stops scrolling but the content still moves before the content actually stops. This event occurs asynchronously.
- onScrollEnd: The action is triggered when scrolling is ended. This event occurs asynchronously.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user's touch is released from the touch surface. This event occurs asynchronously.

- widgetToZoom: An event callback is invoked by the platform to return one of the child widgets of source to zoom. The returning source itself makes the complete scroll container zoomable. If a null value is returned then the container does not zoom. This event occurs asynchronously.

  > ***Note:*** This action is specific to the iOS platform.

- onZoomStart: The action is triggered when the FlexScrollContainer is about to zoom. This event occurs asynchronously.

  > ***Note:*** This action is specific to the iOS platform.

- onZooming: The action is triggered when the FlexScrollContainer is zooming. This event occurs asynchronously.

  > ***Note:*** This action is specific to the iOS platform.

- onZoomEnd: The action is triggered when the zooming has ended. This event occurs asynchronously.

  > ***Note:*** This action is specific to the iOS platform.

For more information, see Add Actions.

TabPane

Use a TabPane widget to organize multiple tabs. Each tab hold a collection of widgets on a form. A user views one tab at a time.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a TabPane widget:

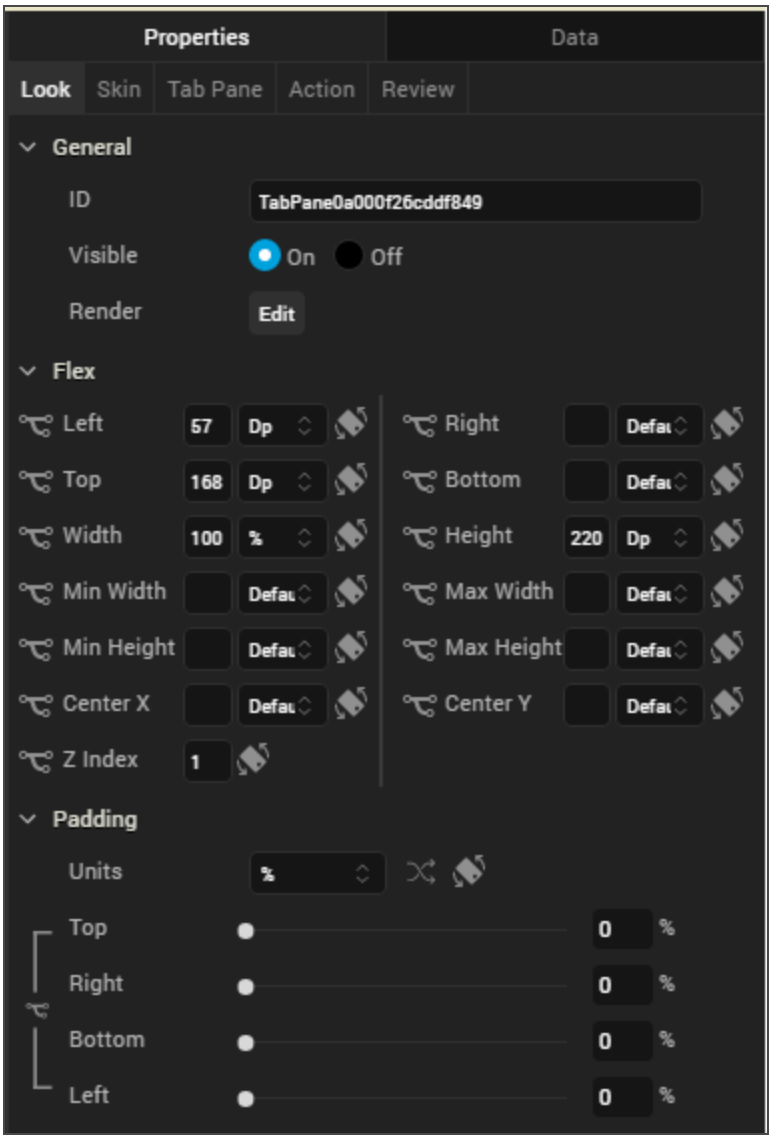
- You can navigate within the TabPane only using the down key.
- If you press the down key, the focus shifts to the next widget in the TabPane.
- If you press the down key while you are on the last widget in the TabPane, you are taken to the top most widget in the TabPane.
- If you press the right or the left arrow keys, you move to next or previous tabs.
- Tab cycling is supported (that is, if you are on the last tab and you press the right arrow, you move to the first tab).
- On devices that have a navigation key, the following are applicable:
    - Each tab has a context menu. The context menu appears in the menu options when the tab has the focus.
    - Widget focus within a tab is saved. For example, if Widget2 of Tab2 has the focus, and you navigate to Tab1 and then back to Tab2, Widget2 still has the focus.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.

- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

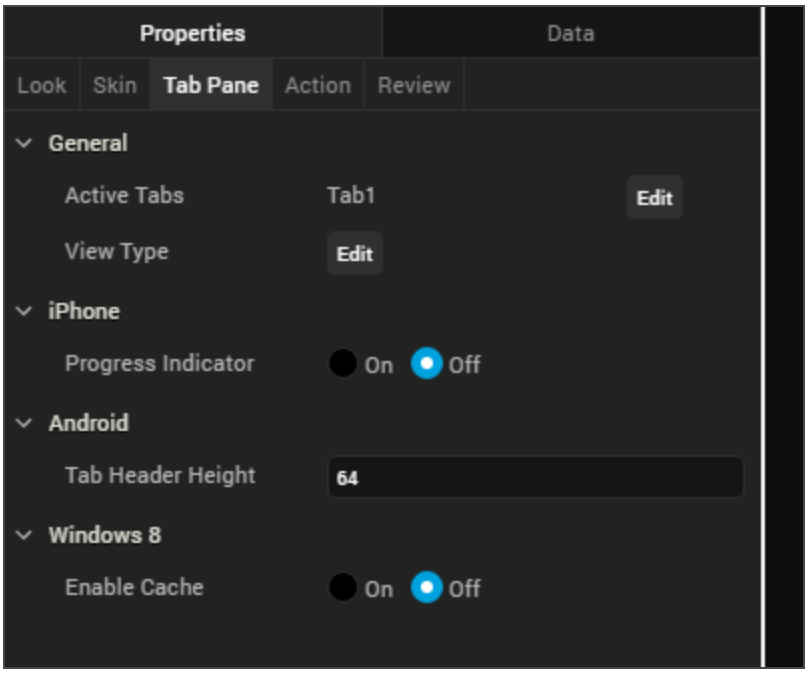
For a TabPane widget, you can apply a skin and its associated properties for the following states

Skin	Definition
Active	The skin applied when a TabPane is active.
Active Focus	The skin applied when a TabPane is active and has the focus.
Inactive	The skin applied to inactive tabs.
Page	The skin for the page indicator. The skin is applicable only when View Type is set to Page and images are specified for Focused Page and Unfocused Page icons.

For more information about applying skins, see Understanding Skins and Themes.

TabPane Properties

TabPane properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Active Tabs

Specifies whether tabs in the TabPane are active or inactive.

Full Screen Widget

For a TabPane on a VBox form, specifies whether the TabPane occupies the entire form.

> ***Note:*** Do not place more than one full-screen widget on a form.

> ***Note:*** This property is available only when a TabPane is placed inside a VBox form.

View Type

Specifies the TabPane view type for each platform. To specify the view type, click the **Edit** button to open the View Type dialog box, select a platform, and then select a view type.

Following are the options:

- Tab
- Collapsible
- Page (iPhone and Android only)
- Panorama (Windows only)

Height

Specifies the height of the TabPane as a percentage of the Height Reference property.

> ***Note:*** This property is available only when TabPane is placed inside a VBox Forms.

Height Reference

Specifies how the Height percentage is calculated.

Following are the options:

- Form Reference: The height percentage is calculated based on the height of the form, excluding headers and footers. This option does not apply if the TabPane is placed inside a popup or a template.

- Parent Width: If the TabPane is placed inside a popup or in templates, the width is calculated based on the width of the parent container.

  > ***Note:*** This property is available only when TabPane is placed inside a VBox Forms.

Progress Indicator

Specifies whether the progress indicator is displayed.

> ***Note:*** This property is specific to the iOS platform.

Progress Indicator Color

Specifies the color of the progress indicator, either Grey or White.

> ***Note:*** This property is specific to the iOS platform.

Tab Header Height

Specifies the height of the tab header.

> ***Note:*** This property is specific to the Android platform.

Actions

Actions define what happens when an event occurs. On a Tab widget, you can run an action when the following events occur:

- onTabClick: The action is triggered when the tab is clicked. This action is available only when the **View Type** is set to **Collapsible View**, and is triggered when you expand or collapse a tab.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.

- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement inside a Widget

The following table summarizes where a TabPane widget can be placed:

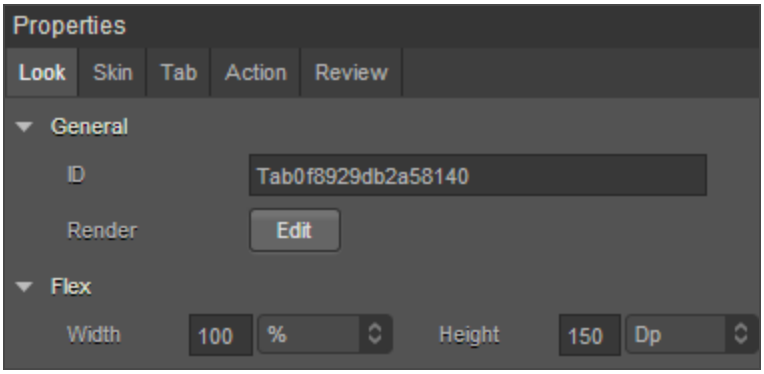
Flex Form	Yes.
VBox Form	Yes, only 1
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes, only 1
VBox	Yes, only 1
ScrollBox	Horizontal Orientation -No Vertical Orientation- No
Tab	No
Segment	Horizontal Orientation -No Vertical Orientation- No
Popup	Yes, only 1
Template	Header- No Footer- No

Tab

Use a Tab widget to hold a collection of widgets on a form. You can use multiple tabs to display groups of related widgets. A user views one tab at a time.

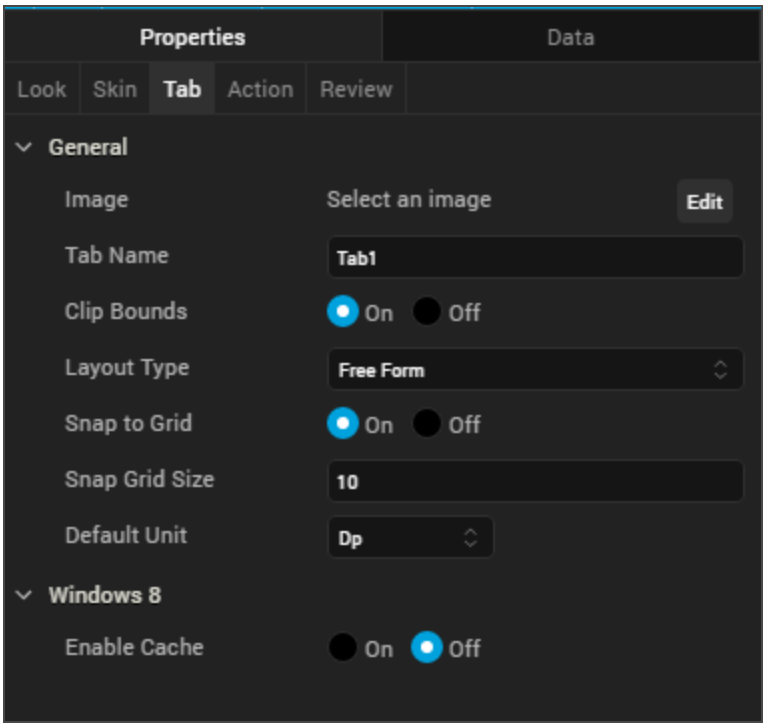
Look Properties

Look properties define the appearance of the widget.



Tab Properties

Tab properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Image

Specifies an image to display next to the tab name. To select an image, click **Edit** to display the **Image** window. From the list of images, choose an image, and then click **OK**.

Tab Name

Specifies the name of the tab.

Clip Bounds

Specifies whether to clip child widgets displayed in the tab pane when they go out of boundaries.

Layout Type

Specifies whether the arrangement of widgets in the tab flows horizontally, vertically, or in both directions.

Default: Vertical

Snap to Grid

Specifies whether the widget aligns to the nearest intersection of lines in the grid, or other widgets.

Snap Grid Size

Specifies the grid size. This option is available only when **Snap to Grid** is enabled.

Default Unit

Specifies the default unit used for interpretation of numbers with no qualifiers when passed to layout properties.

Following are the options:

- dp: Specifies the values in terms of device independent pixels.
- px: Specifies the values in terms of device hardware pixels.
- %: Specifies the values in percentage relative to the parent dimensions.

Actions

Actions define what happens when an event occurs. On a Tab widget, you can run an action when the following events occur:

- onInit: The action is triggered when the user navigates to the tab.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

A Tab widget can only be placed inside a TabPane widget.

Button

Use a Button widget to trigger an action or set of actions; for example, clicking the button might open a form or dialog box, start an action or series of actions, or confirm an action.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

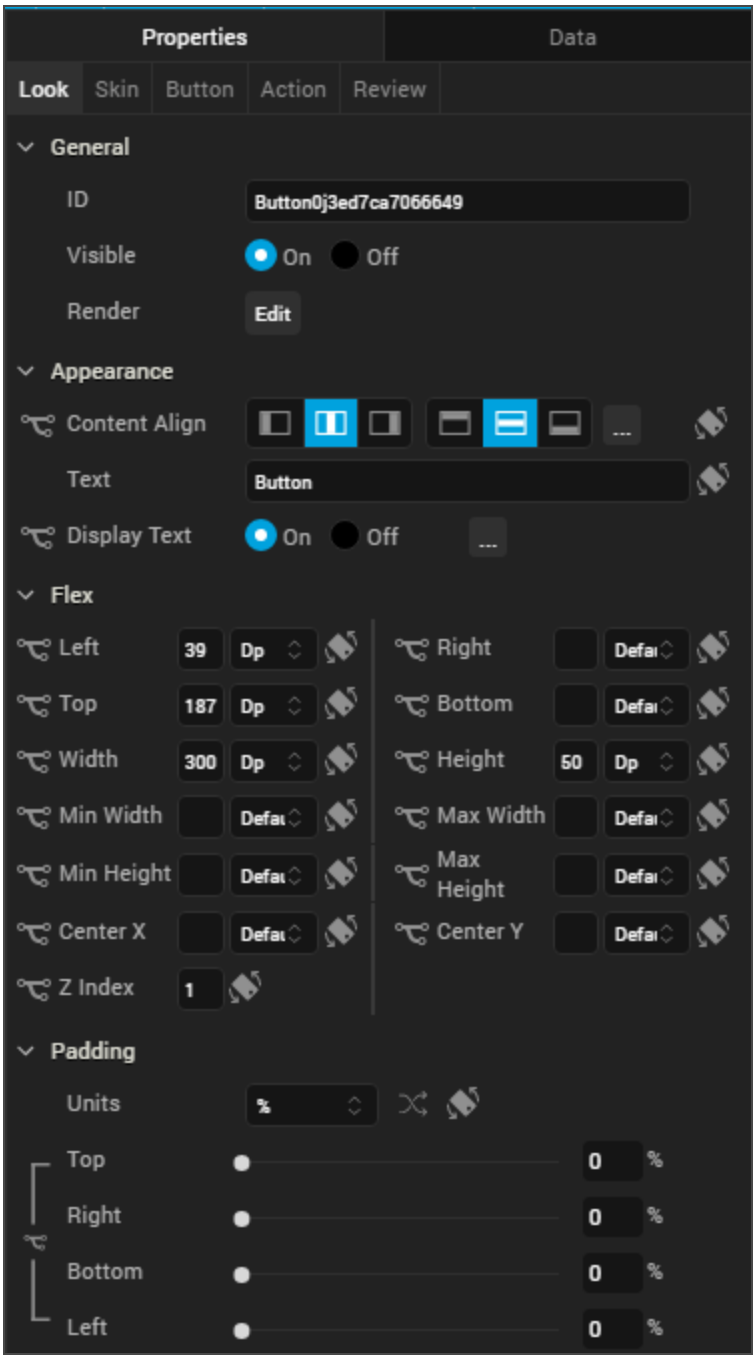
The following are important considerations for a Button widget.

- You can specify different skins for normal and focus states of a button.
- You can specify a background image for normal and focus states of a button.
- To avoid a jumping effect or toverlap of neighboring widgets, make sure the image for normal and focus skins are the same size.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

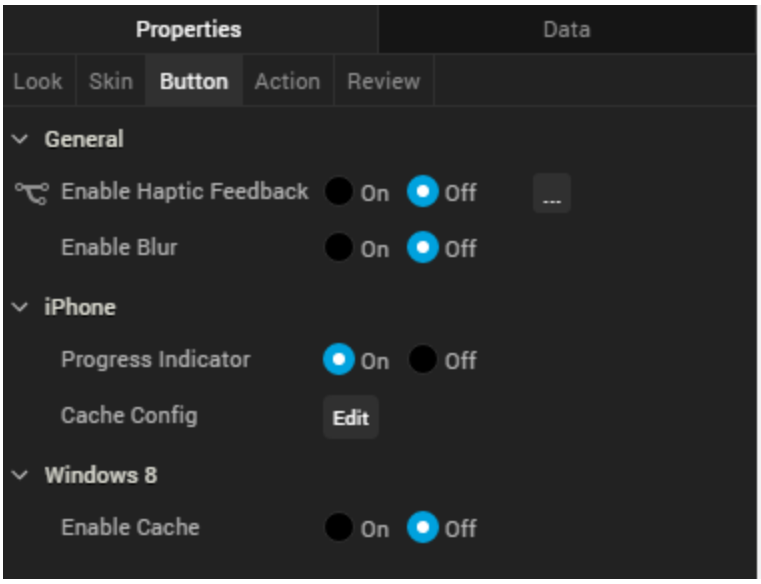
For a Button widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the focus is on the widget.
Blocked UI	The skin applied to block the interface until the action in progress is completed; for example, a service call . ***Note:*** The Blocked UI skin is available only for SPA platforms.
Pressed	The skin to indicate that the widget is pressed or clicked. ***Note:*** The Pressed skin is available only on Android Native platforms.

For more information about applying skins, see Understanding Skins and Themes.

Button Properties

Button properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Progress Indicator

For the iOS platform, specifies whether to display the progress indicator when a user clicks the button. Enable this property when the loading time for an application is long.

Default: The progress indicator is enabled.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget.

Actions

Actions define what happens when an event occurs. On a Button widget, you can run an action when the following event occurs:

- onClick: The action is triggered when the user clicks on the button.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.

- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement inside a Widget

The following table summarizes where a Button widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation -Yes Vertical Orientation -Yes
Tab	Yes
Segment	Horizontal Orientation -Yes Vertical Orientation -Yes
Popup	Yes
Template	Header- No Footer- No

import Calendar

Use a Calendar widget to display or select a date from a calendar. Click the calendar icon to select a date. The date and the calendar display in the format you specify. The icon does not appear on Mobile Web platforms .

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a Calendar widget:

- If you do not specify an image, clicking the calendar icon displays the default calendar image.
- By default, the Calendar widget occupies the complete width of its parent widget.
- Clicking the calendar icon opens the current month calendar. On the Android platform, it is not possible to restrict the date selection between validStartDate and validEndDate with the native calendar view.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.

For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For a Calendar widget, you can apply a skin and its associated properties for the following states:

> ***Important:*** Custom border capability is now extended to iOS for the Calender widget.

Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the focus is on the widget.
Grid	The skin properties applied for the grid calendar.
Cell	The skin properties applied for the grid calendar cell.
Cell - Selected	The skin applied when a grid calendar cell is selected.
Cell - Focus	The skin applied when there is a focus on the grid calendar cell.
Cell - Today	The skin applied for the grid calendar today cell.
Cell -Weekend	The skin applied for the weekend days of a grid calendar.
Cell - Inactive	The skin applied for non-working inactive days of a grid calendar.
Done Button	The skin to be applied for Done button that appear on a calendar pop up.
Cancel Button	The skin to be applied for a Cancel button that appear on a calendar pop up.
Day	The day portion of the currently selected date.

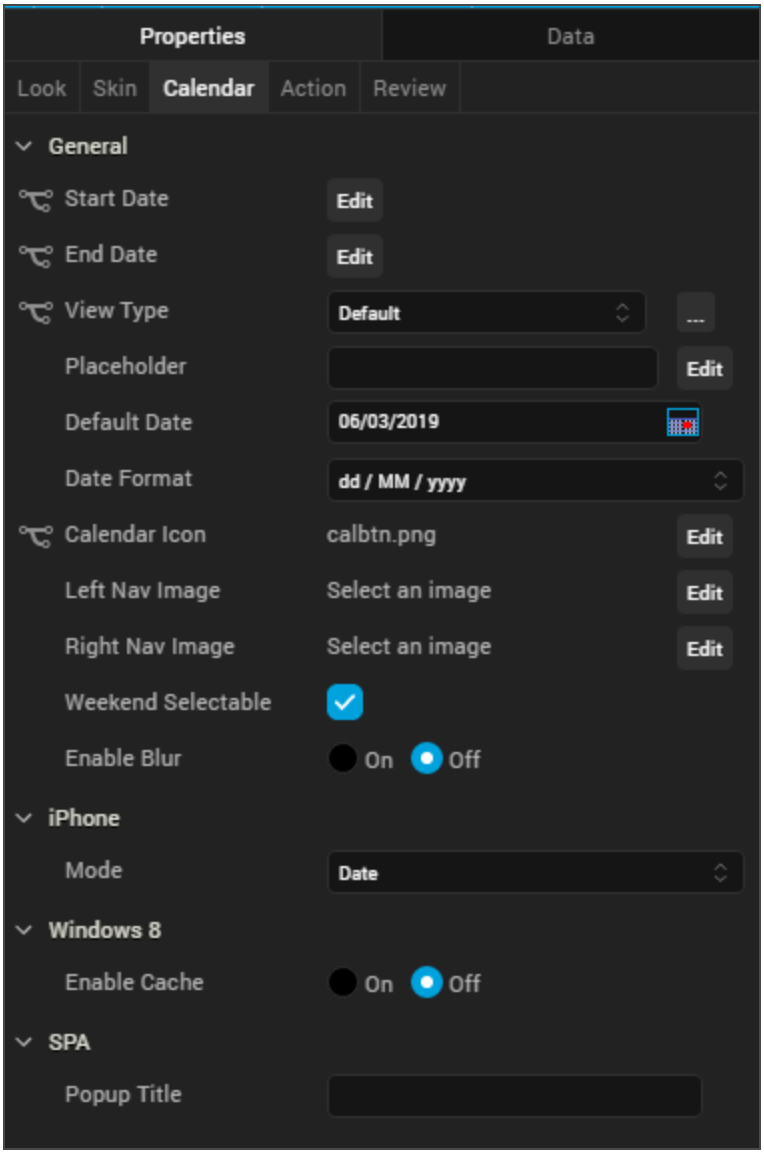
Skin	Definition
Month	The month portion of the currently selected date.
Hover Skin	The look and feel of a widget when the cursor hovers over the widget. ***Note:*** Hover Skin is available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

Calendar Properties

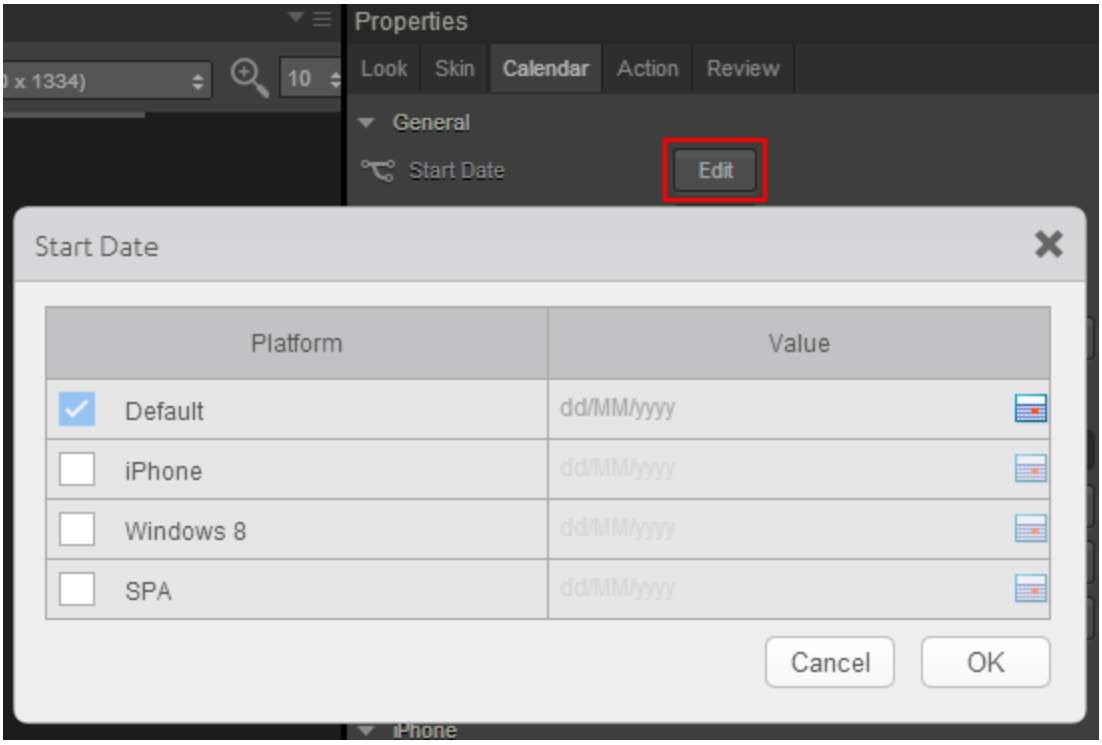
Calendar properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

Note: In this section, properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Start Date

Specifies the start date in the specified Data Format. The default start date is the current date.



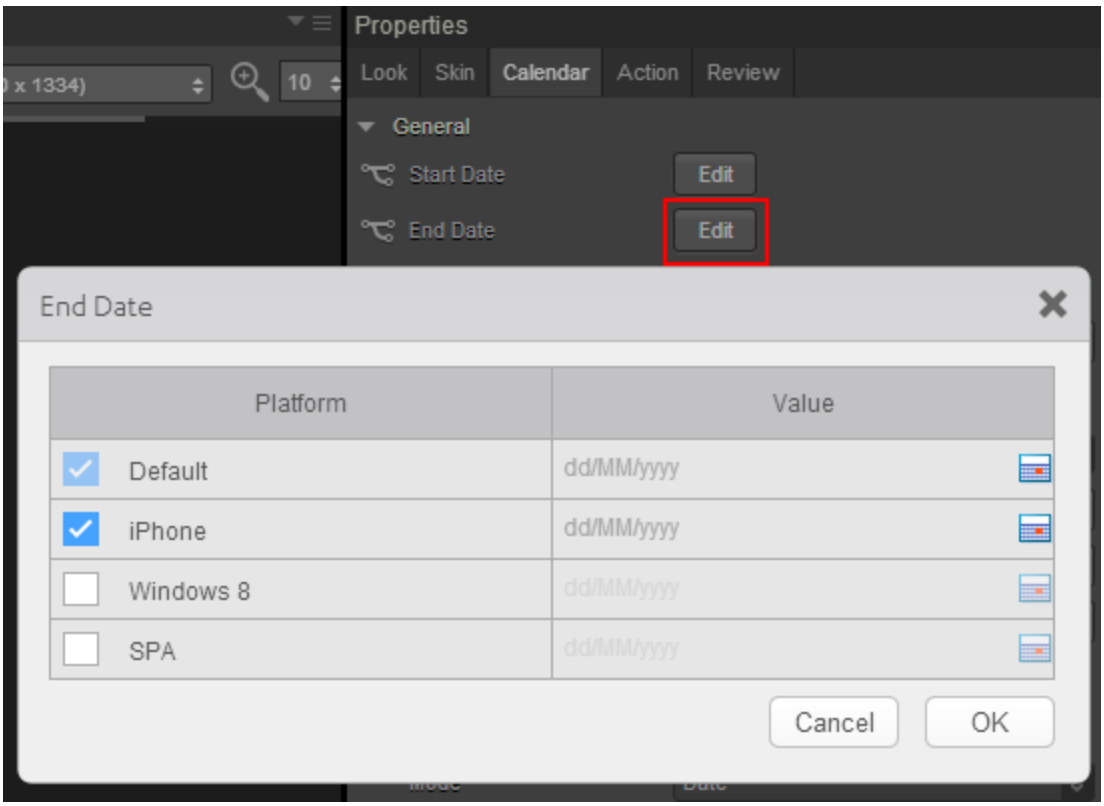
To input a different default start date or to provide a platform-specific start date, click **Edit** to open the **Start Date** dialog box.

The **Default** value is the current date. To change this value, click the calendar icon and select a date from the calendar.

To provide a platform-specific start date, click the desired platform check box, and then click the calendar icon in the corresponding **Value** field. Select a date from the calendar.

End Date

Specifies the end date in the specified Data Format. The default end date is the current date.



To input a different default end date or to provide a platform-specific end date, click **Edit** to open the **End Date** dialog box.

The **Default** value is the current date. To change this value, click the calendar icon and select a date from the calendar.

To provide a platform-specific start date, click the desired platform check box, and then click the calendar icon in the corresponding **Value** field. Select a date from the calendar.

View Type

Specifies the calendar view type:

- Default: Specifies the default native calendar view in respective platforms.
- Onscreen Grid
- Popup Grid (only SPA platforms supports this option.)

The **View Type** of the widget selected from this list becomes the default for all the platforms. You can provide a platform-specific value for a platform by forking the **View Type** property. See Fork a Widget Property for more details.

Default Date

Specifies a default date to be displayed on a Calendar widget. You can choose a date by clicking the calendar icon, and then selecting a date.

This date replaces any Placeholder value.

Data Format

Specifies the date format.

Default: DD/MM/YYY

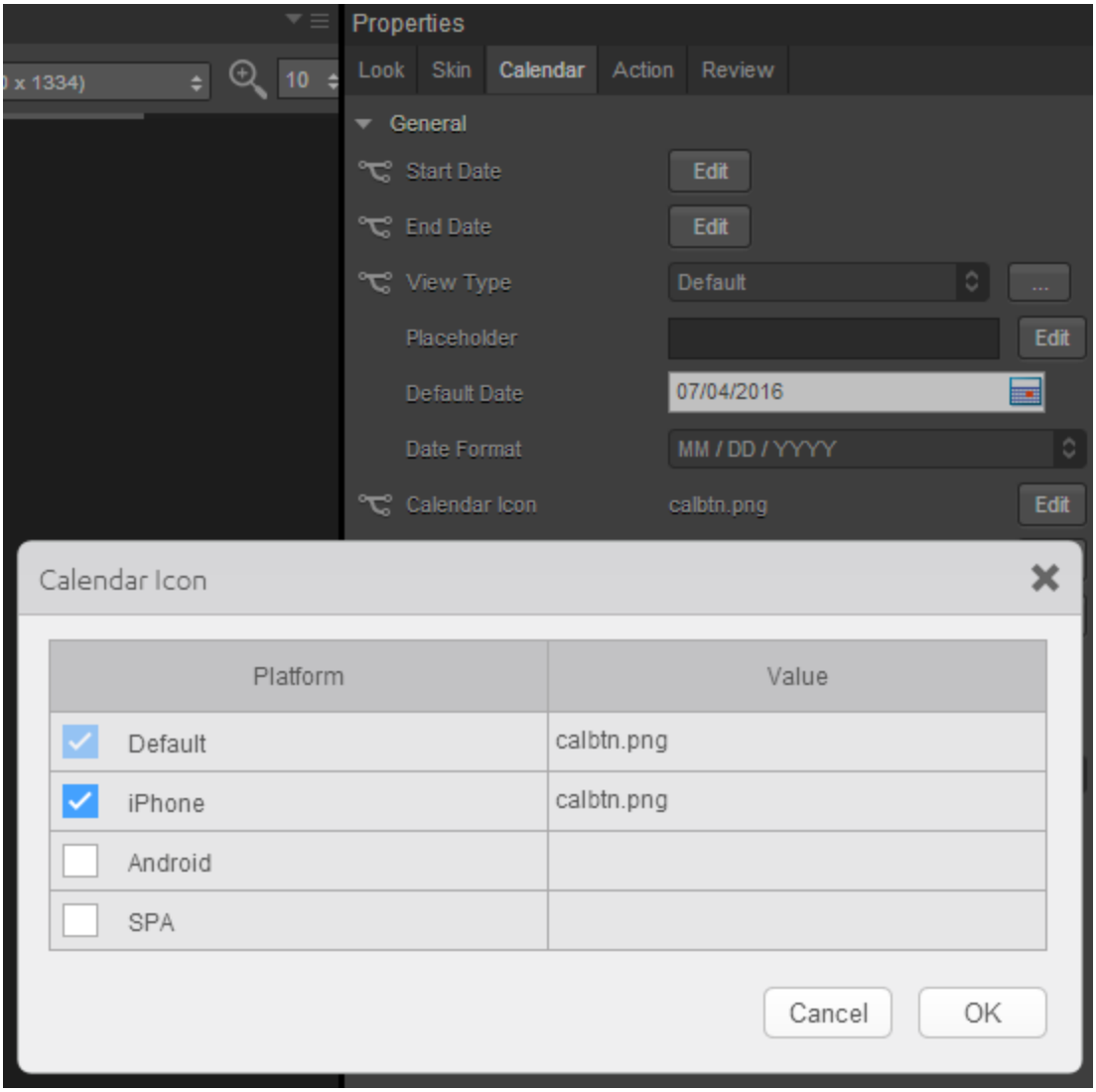
Following date formats are supported:

- DD/MM/YYYY
- MM/DD/YYYY
- MM/DD/YY

Calendar Icon

Specifies the calendar icon.

Kony Visualizer provides a default calendar icon. You can replace the default icon or provide a platform-specific icon by clicking the **Edit** button to open the **Calendar Icon** dialog box.



Select the platform and click the **value** field to open the **Select Image** dialog box. You can either select an image in the dialog box or provide an image URL.

Left Nav Image

Specifies the left navigation image that appears on the calendar popup. Click this image to navigate and select a past date from the calendar.

To specify an image, click the **Edit** button to open the **Left Nav Image** dialog box. You can either select an image in the dialog box or provide an image URL.

Right Nav Image

Specifies the right navigation image that appears on the calendar popup. Click this image to navigate and select a future date from the calendar.

To specifiy an image, click the **Edit** button to open the **Right Nav Image** dialog box. You can either select an image in the dialog box or provide an image URL.

Weekend Selectable

Select the check box to enable selection of weekend dates.

Mode

Specifies the calendar mode. (iOS only)

Following are the options:

- Date (Default): Allows selection of a date only.
- Time: Allows selection of a time only
- Date & Time: Allows selection of both date and time.

Hide Days Header

This property is available only when View Type is set to Onscreen Grid or Popup Grid. It specifies whether the weekdays are hidden on the header of a grid calendar.

- **On** hides weekdays.
- **Off** displays weekdays.

> ***Note:*** This property is specific to iOS and Android platforms.

Hide Months Header

This property is available only when View Type is set as *Onscreen Grid* or *Popup Grid*. It indicates whether the months header is hidden for the grid calendar, including the navigation buttons.

- Click **On** to hide the months header.
- Click **Off** to display the months header.

> ***Note:*** This property is specific to iOS and Android platforms.

Day Text Alignment In Cell

This property is available only when the View Type is set to *Onscreen Grid* or *Popup Grid*. It specifies the alignment of the text for a Calendar Day cell, regarding its boundaries. The alignment values are:

- Top-Left: Specifies the text should align at the top left corner of a Calendar Day cell.
- Top-Center: Specifies the text should align at the top center of a Calendar Day cell.
- Top-Right: Specifies the text should align at the top right of a Calendar Day cell.
- Middle-Left: Specifies the text should align at the middle left of a Calendar Day cell.
- Center: Specifies the text should align at the center of a Calendar Day cell.
- Middle-Right: Specifies the text should align at the middle right of a Calendar Day cell.
- Bottom-Left: Specifies the text should align at the bottom left of a Calendar Day cell.
- Bottom-Center: Specifies the text should align at the bottom center of a Calendar Day cell.
- Bottom-Right: Specifies the text should align at the bottom right of a Calendar Day cell.

> ***Note:*** This property is specific to iOS and Android platforms.

Cell Template

This property is available only when the View Type is set to *Onscreen Grid* or *Popup Grid*. It specifies the common template to be used for a Calendar Day cell. A template is used only when the data is present for a Calendar Day cell. If the data is not set to a cell, the cell appears with the default look, which has no template.

To assign a template for cell template:

1. Click **Edit** to display the **cellTemplate** window.
2. From the list of templates, select a template.
3. Click **OK**.

> ***Note:*** This property is specific to iOS and Android platforms.

Data

To modify the **Data** property, you need to assign a grid template to the Cell Template property.

Data represents the actual data to be rendered in each cell.

To specify the data:

1. Click Edit to display the **Master Data for GridCalendar** window.
2. From the **Data** column, click the calendar icon.
3. Select a date.
4. From the **Template Data** column, click the ellipsis button to display the **Template Data** window and update the data or skin of the widget.
5. Click **OK**.

> ***Note:*** This property is specific to iOS and Android platforms.

Height

This property is available only when View Type is set as *Onscreen Grid* or *Popup Grid*. It specifies the available height of the container in terms of percentage. The percentage is with reference to the value of Height Reference property.

> ***Note:*** This property is specific to the iOS and Android platforms.

Height Reference

This property is available only when View Type is set as *Onscreen Grid* or *Popup Grid* and when you set the Height property.

The container height percentage is calculated based on the following:

- Form Reference: The Calendar height is a percentage based on the height of the Form excluding header and footer.
- Parent Width: Use this option if the Calendar is placed inside a Box. The width is calculated based on the width of the Box.

> ***Note:*** This property is specific to iOS and Android platforms.

Popup Title

For the SPA platform, specifies the title text to be displayed on the calendar popup.

Tool Tip

For the Windows platform, specifies a message that displays when you hover the mouse pointer over the widget.

Actions

Actions define what happens when an event occurs. On a Calendar widget, you can run an action when the following event occurs:

- onSelection: The action is triggered when an item is selected.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.

- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

The following table summarizes where a Calendar widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

CheckBoxGroup

Use a CheckBoxGroup widget to select from a group of check boxes when a user can make one or more selections. A check mark inside the check box indicates the selection.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a CheckBoxGroup Widget.

All Platforms

- A CheckBoxGroup widget is always a group widget.
- Use a CheckBoxGroup widget if there are a limited number of possible selections and you can make more than one selection. If you can make only one selection from the group, use a RadioButtonGroup widget. To display a list of selections, use a ListBox widget.

Android

- If you set the Orientation property to horizontal, do not place more than two items in the group. If you place more than two items and the associated text is large, additional items may not fit in the screen width and will not be visible.

iPhone

- You cannot apply skins in the on-off switch view.

SPA

- The Focus skin is not supported.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.

For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

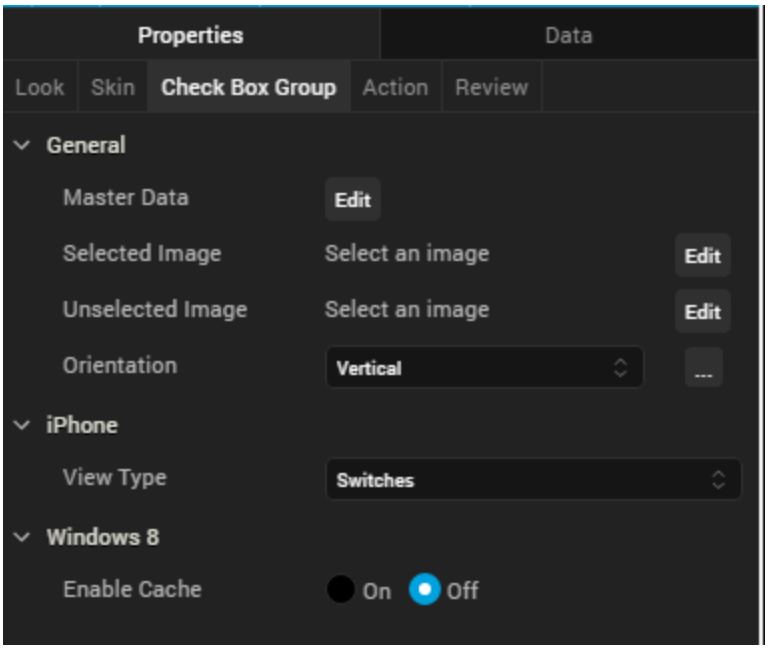
For a CheckBoxGroup widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the focus is on the widget.

For more information about applying skins, see Understanding Skins and Themes.

CheckBoxGroup Properties

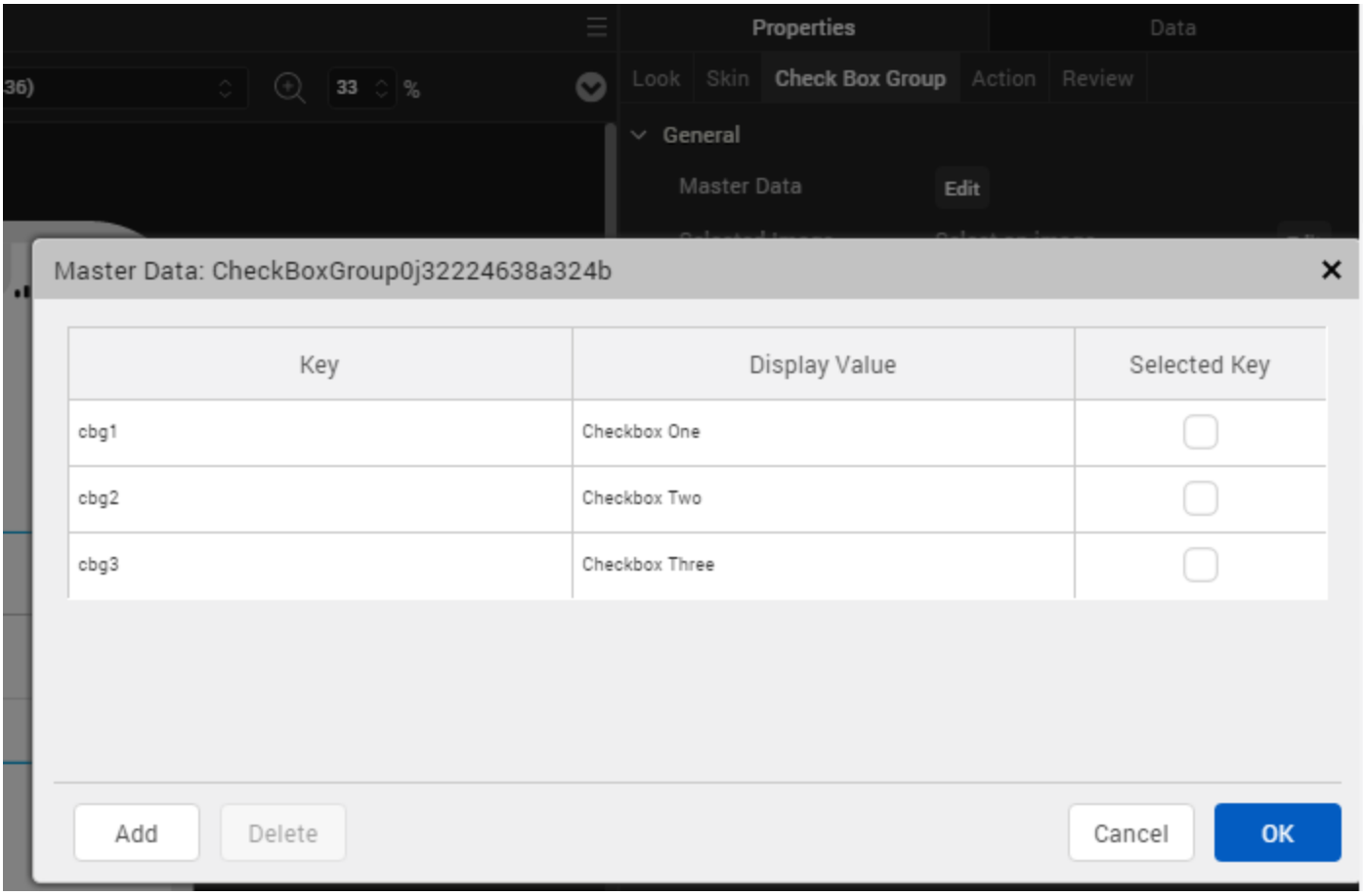
CheckBoxGroup properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Master Data

Specifies the set of values that must be displayed for the user to make a selection from the available choices.

To specify this set of values, click the **Edit** button of the **Master Data** field to open the **Master Data** dialog box.



The Master Data dialog box contains the following columns:

- Key: The unique identifier of each check box.
- Display Value: The label or descriptive text displayed for each check box
- Selected Key: Whether each check box is selected by default. You can select more than one check box.

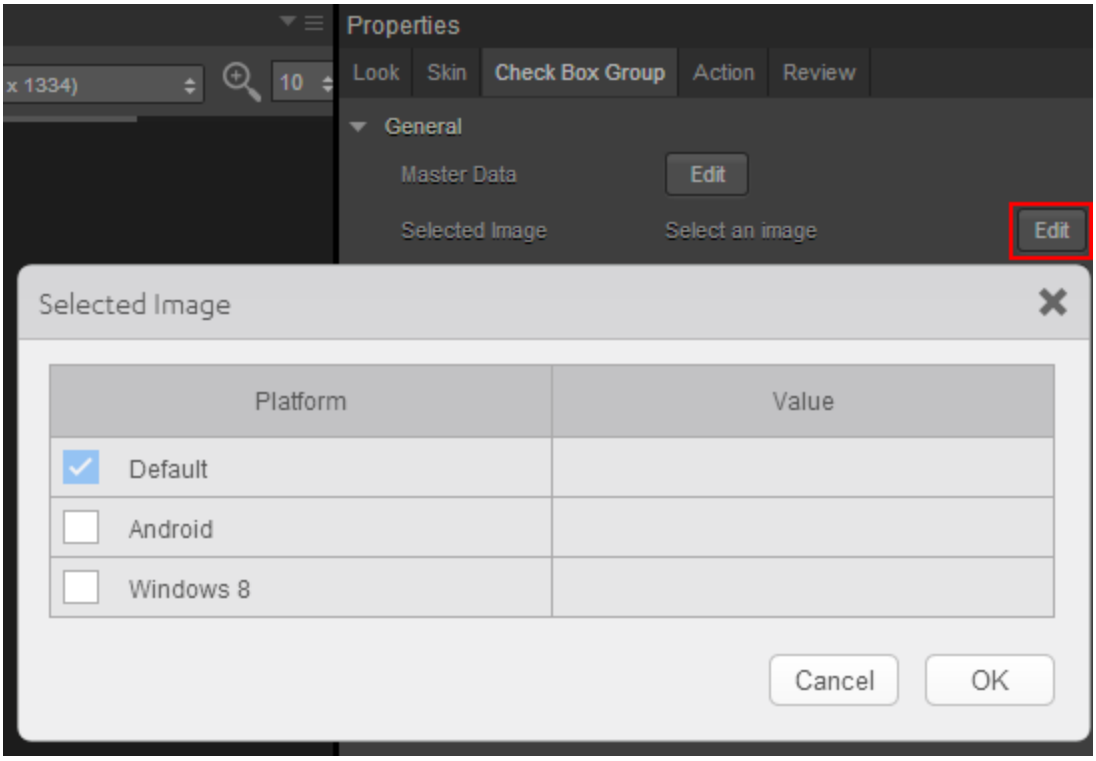
To add more check boxes to the widget, click **Add**. To delete a check box, click inside a check box, and then click **Delete**.

Click **Apply** to create the master data.

Selected Image

Specifies the image to be displayed when you make a selection.

To provide a default or platform-specific image, click the **Edit** button to open the **Selected Image** dialog box.



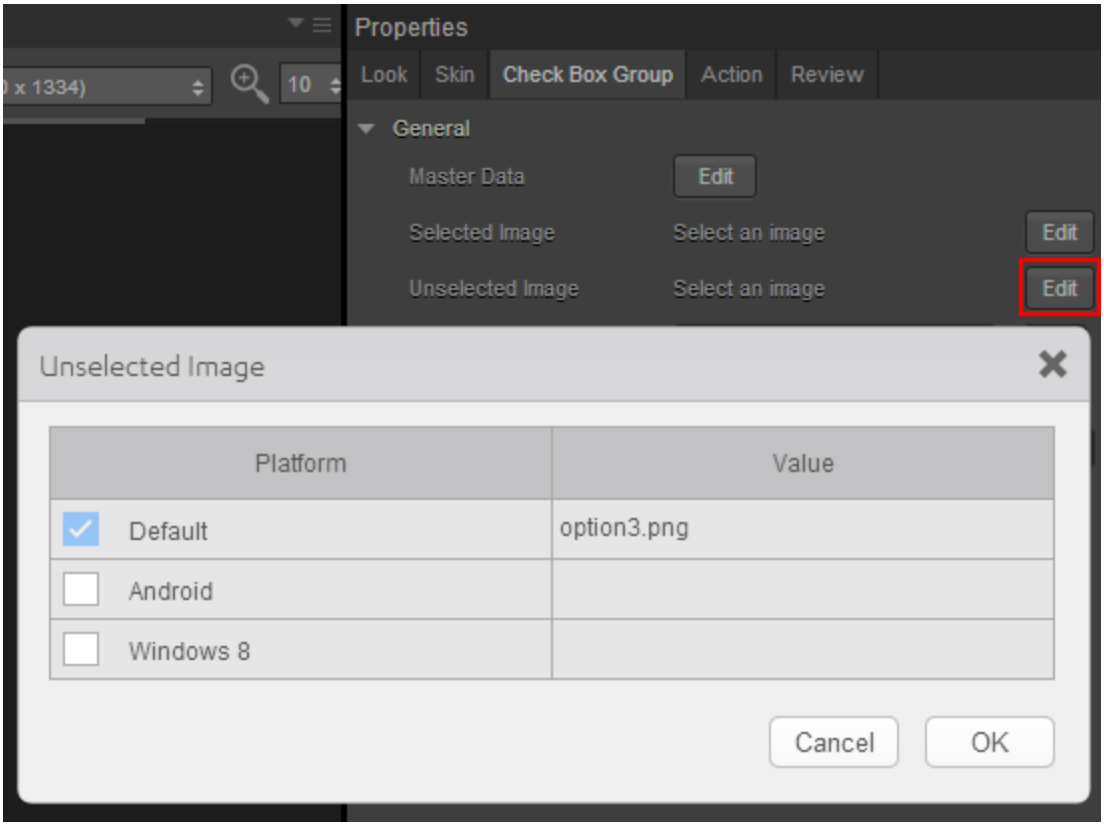
To provide a platform-specific image or replace the default image, select the platform and click inside the corresponding **Value** field to open the **Selected Image** dialog box. You can either:

- Select an available image.
- Provide an image URL.

Unselected Image

Specifies the image to be displayed when a selection is cleared.

To provide a default or platform-specific image, click the **Edit** button to open the **Unselected Image** dialog box.



To provide a platform-specific image or replace the default image, select the platform and click inside the corresponding **Value** field to open the **Selected Image** dialog box. You can either:

- Select an available image.
- Provide an image URL.

Orientation

Specifies whether the alignment of the check boxes is horizontal or vertical.

Default: Vertical

View Type

For the iOS platform, specifies the view type of the CheckBoxGroup, either Switches, Table, or On-screen Wheel.

Default: Switches

Group Cells

When the view type is **Table**, specifies whether the Group Cells style is applied. The Group Cells style groups items in the check box group.

Default: The Group Cells style is applied.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a CheckBoxGroup widget, you can run an action when the following event occurs:

- onSelection: The action is triggered when an item is selected.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

The following table summarizes where a CheckBoxGroup widget can be placed:

Flex Form	Yes

VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

DataGrid

Use a DataGrid widget to display a collection of data in rows and columns (tabular format). The DataGrid widget supports common table formatting options, such as alternating the row background color, customizing the grid line, and the ability to hide or show headers.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

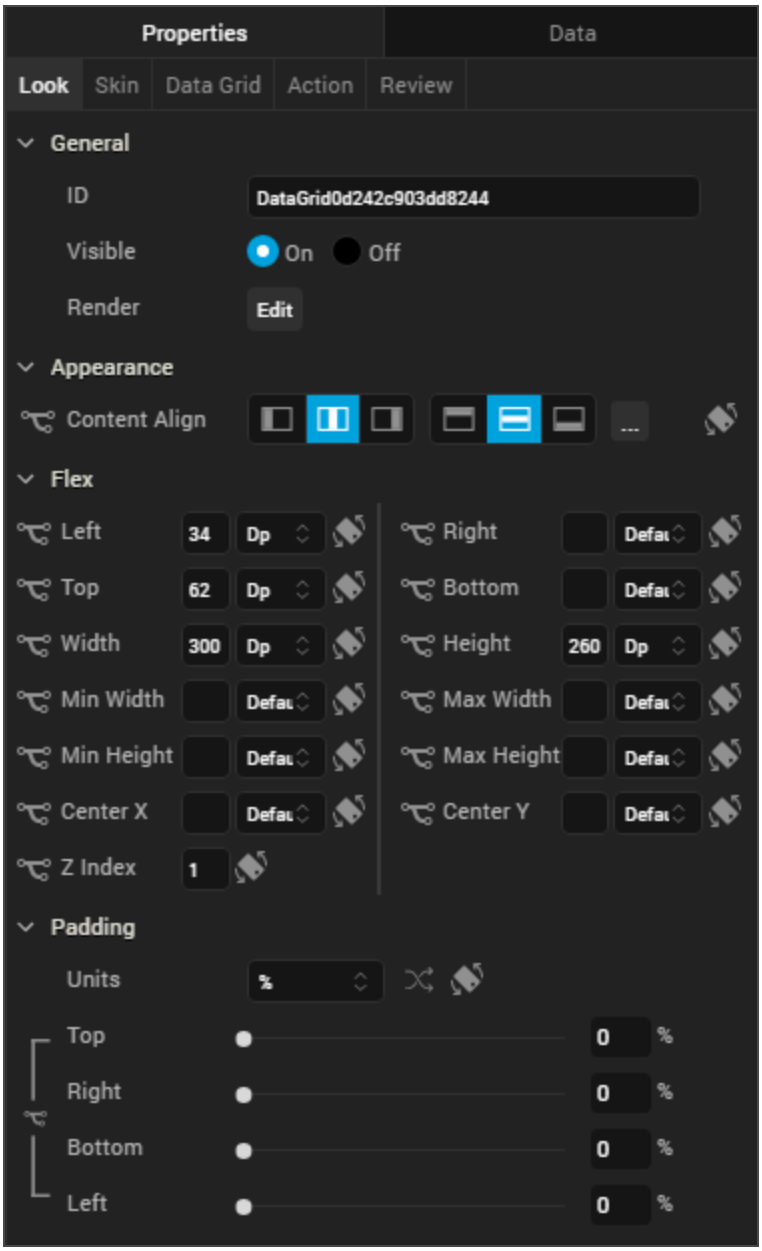
The following are important considerations for a DataGrid widget.

- To set the data, first specify the rows and columns using the Data property.
- If the DataGrid supports the Multi-Select property, make sure to specify a Row - Focus property. Otherwise, you will not be able to distinguish multiple selections.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

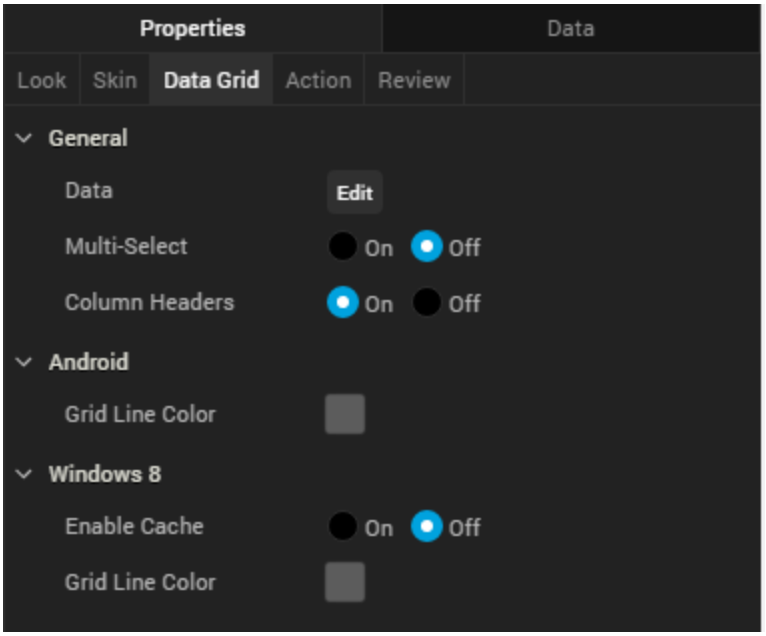
For a DataGrid widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Header	The skin applied to the Header row.
Row	The skin applied when the row does not have the focus.
Row - Alternate	The skin applied to alternate rows.
Row - Focus	The skin applied when the row has the focus.
Hover Skin	The look and feel of the widget when the cursor hovers over the widget. ***Note:*** Hover skins are available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

DataGrid Properties

DataGrid properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Data

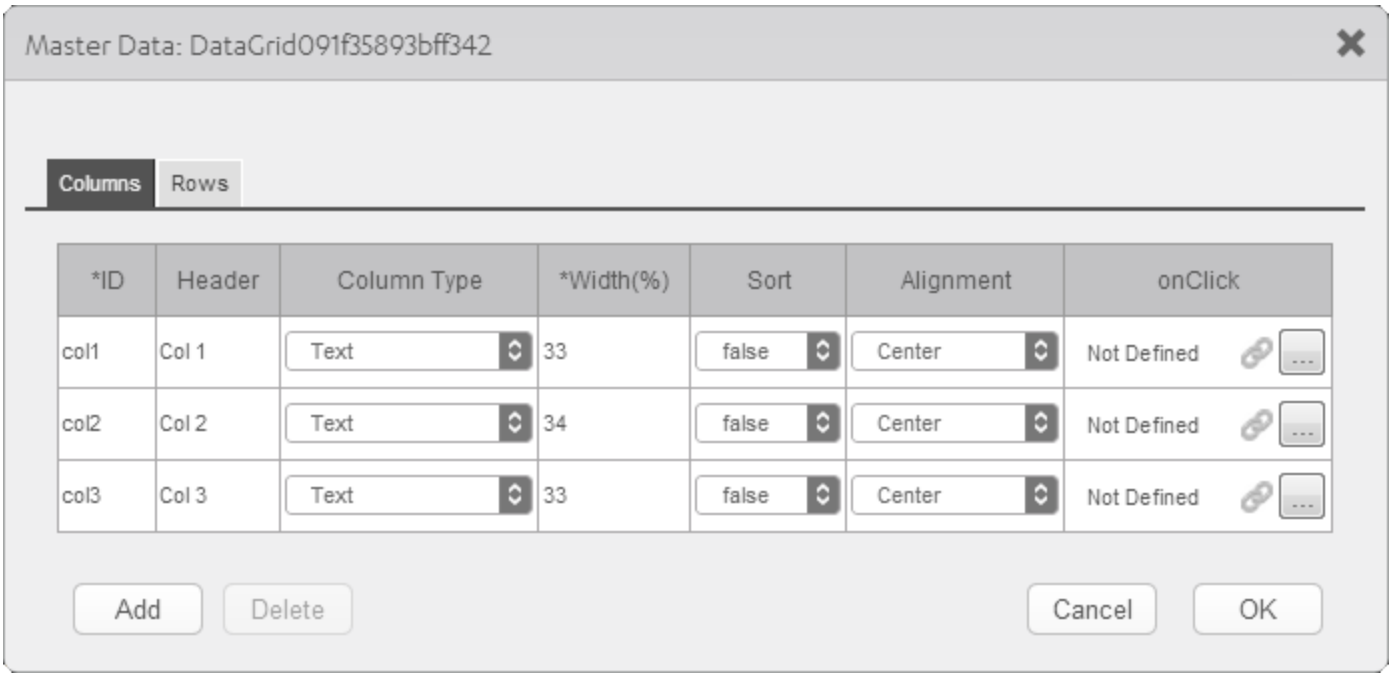
Represents the data displayed in each cell of the data grid.

To input the data, click the **Edit** button to open the **Master Data : DataGrid** dialog box.

The **Master Data : DataGrid** dialog box includes two tabs:

- Columns
- Rows

Columns



The Columns table lets you provide details such as:

- ID: Specifies the unique identifier of a column.
- Header: Specifies the Column header.
- Column Type: Specifies whether the column type is text or image.
- Width (%): Specifies the width of each column as percentage of the data grid.

- Sort: Specifies whether sorting is enabled for the column.
- Alignment: Specifies the alignment of the content in each column. The following alignment options are available:
    - Top-Left
    - Top-Center
    - Top-Right
    - Middle-Left
    - Center
    - Middle-Right
    - Bottom-Left
    - Bottom-Center
    - Bottom-Right
- OnClick: Specifies an action that takes place when you click on the header of a column. To specify an OnClick action for a column, click the ellipses button (...).

Rows

The fields in the **Rows** tab are based on the inputs provided in the **Columns** tab and include:

- Column Headers are the **Header** values specified in the **Columns** tab.
- Table cells whose **Column Type** is **Text** are text fields. You can type the text within these cells.
- Table cells whose **Column Type** is **Image** are image fields. You can choose an image to be displayed in these cells. Click the ellipses button (...) in the image field to open the**Select Image** dialog box, and then select an image or provide the URL of the image and click **OK**. The selected image displays in the cell.
- The text or image is aligned according to the **Alignment** setting.

Multi-Select

Specifies whether you can choose multiple rows of the DataGird. The Row - Focus skin is applied to selected rows.

Column Headers

Specifies the visibility of the DataGrid column headers.

Grid Line Color

Specifies the color of the grid line.

> ***Note:*** This property is specific to the Android platform.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a DataGrid widget, you can run an action when the following event occurs:

- onRowSelected: The action is triggered when a row is selected.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information on using this action, see Add Actions.

Placement Inside a Widget

The following table summarizes where a DataGrid widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation - Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Image

Use an Image widget to display a graphic image such as a company logo, photo, or illustration in PNG, JPEG, GIF, or WebP format.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for an Image widget:

- Before using an image in a project, copy the image to a project subfolder. For more information, see Add and Manage Images and Other Media.
- On an Android device, imported images from a drawable folder are ignored during functional preview.
- An Image file name must contain only lowercase characters and numbers, must begin with an alphabetic character, and can't be a JavaScript reserved word or keyword.

The following table shows examples of valid and invalid file names:

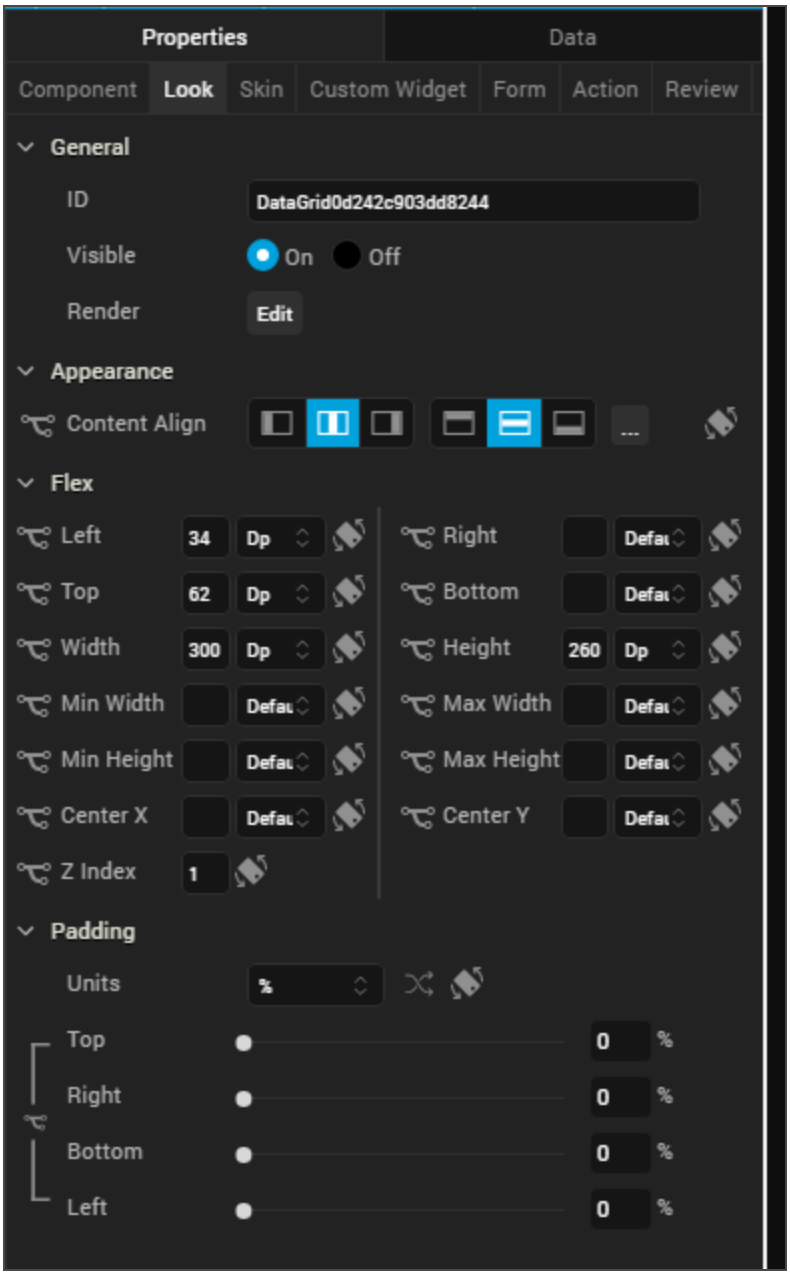
Valid File Names	Invalid File Names	Remarks
myicon.png	Myicon.png	The file name is invalid because it contains an uppercase character.
icon2.png	Icon_2.png	The file name is invalid because it contains uppercase character and an underscore.
accntsummary.png	aCCNT&summary.png	The file name is invalid because it contains uppercase characters and special character.
accountdetails.png	Details.png	The file name is invalid because it contains an uppercase character.

Valid File Names	Invalid File Names	Remarks
flightstatus123.png	continue.png	The file name is invalid because it contains a Java keyword.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

> ***Note:*** On iOS and the Android platforms, rounded border skins are not supported.

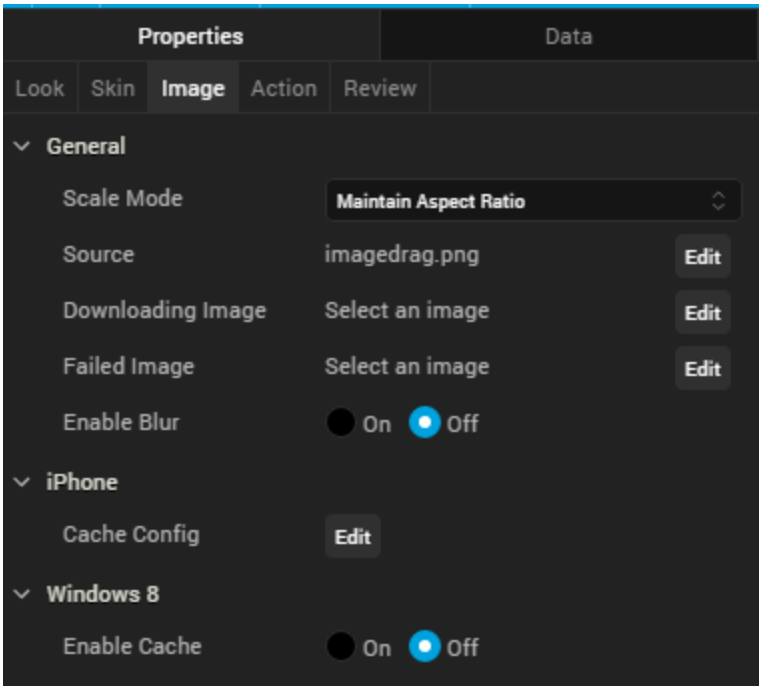
For an Image widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.
Hover Skin	The look and feel of the widget when the cursor hovers over the widget. ***Note:*** Hover skins are available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

Image Properties

Image properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Scale Mode

Specifies how the image is scaled within the Image widget.

Default: Maintain Aspect Ratio

The following scale options are available:

- Fit To Dimensions
- Maintain Aspect Ration
- Crop

Source

Specifies the source of the image to be displayed. You can specify an image from the Resources folder or an image URL.

To specify an image:

1. Click the **Edit** button to open the Source dialog box.
2. Do one of the following:
    - Locate and select the image you want from the list of available images.

      Enter a partial file name in the **Search** box to narrow down the number of images listed. You can also change how the images are displayed by clicking the Layout icons in the upper-left portion of the Source dialog box. You can display the image files as a grid of small previews; a list that includes file names, dimensions, and sizes; or as a file list with a preview pane that displays a preview of the currently highlighted image.

    - In the http Image URL text box, enter an image URL.
3. Click **Open**. The Source property displays the file name or URL of the image.

Downloading Image

Specifies the image displayed when the remote source is being downloaded.

To specify an image, click the **Edit** button open the **Downloading Image** dialog box. You can either select an available image or specify an image URL.

Failed Image

Specifies the image to be displayed when the remote resource is not available.

To specify an image, click **Edit** button to open the **Failed Image** dialog box. You can either select an available image or specify an image URL.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On an Image widget, you can run an action when the following event occurs:

- onDownloadComplete: The action is triggered when the image download is complete.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

The following table summarizes where an Image widget can be placed:

Flex Form	Yes

VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
Scroll Box	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	Yes
Popup	Yes
Template	Header- No Footer- No

Label

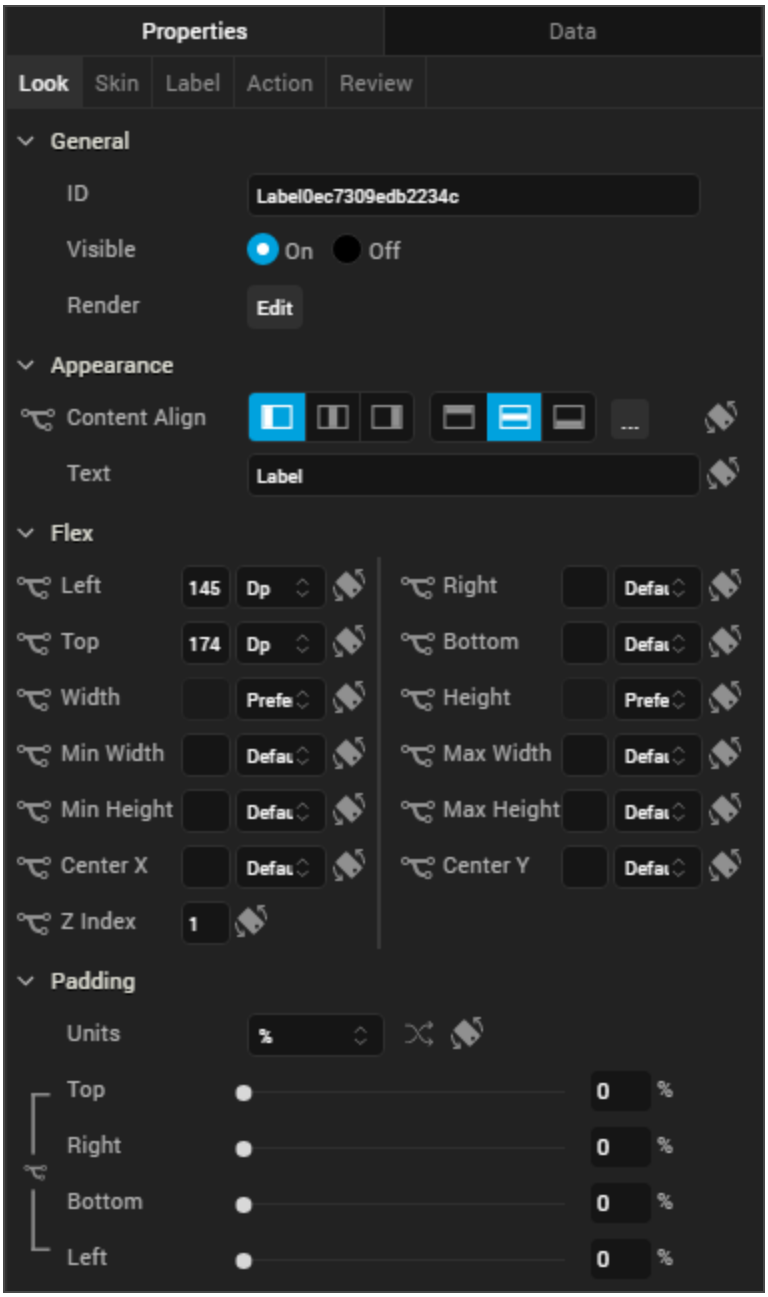
Use a Label widget to display static text. A label typically identifies a nearby text box or other widget.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

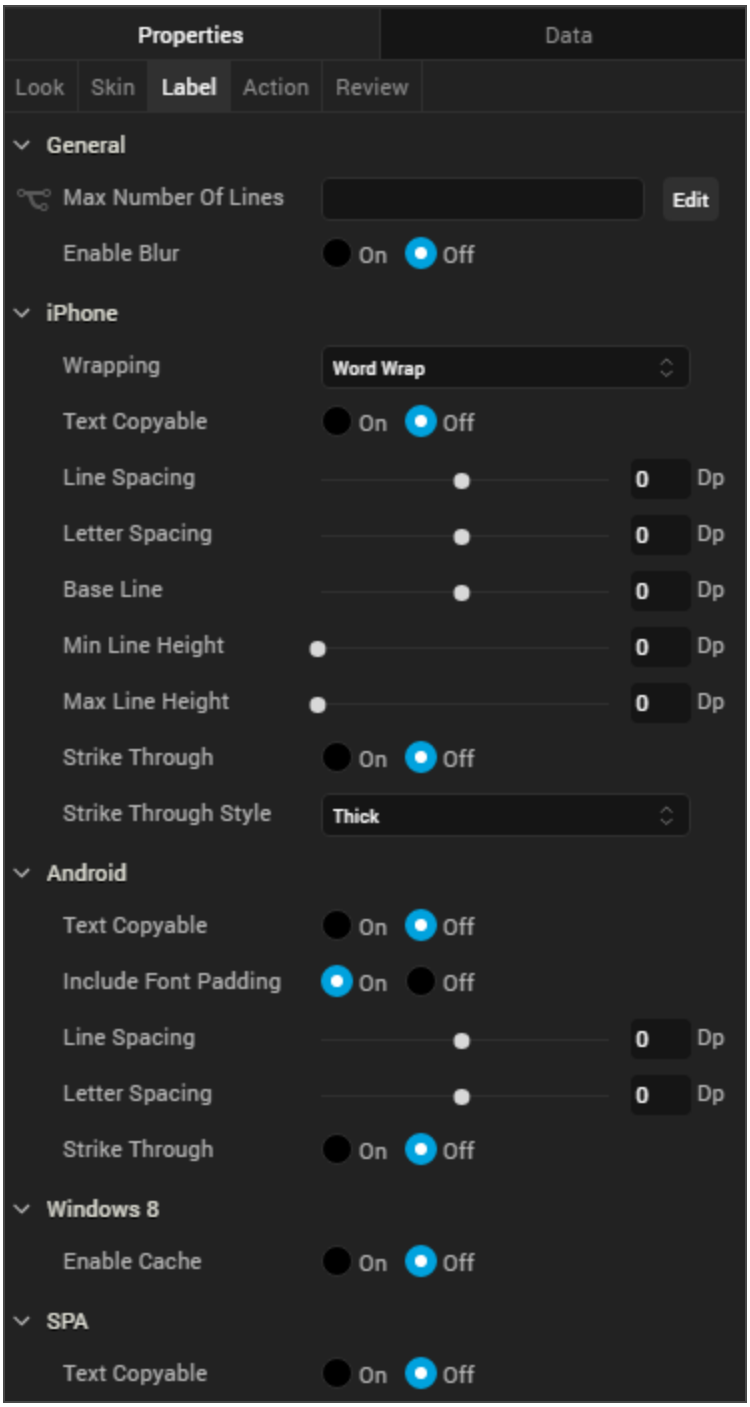
For a Label widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.
Hover Skin	The look and feel of the widget when the cursor hovers over the widget. ***Note:*** Hover skins are available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

Label Properties

Label properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Wrapping

Specifies how the label wraps text, either Word Wrap or Character Wrap:

- Word Wrap (Default): Text wraps between words at the end of a line.
- Character Wrap: Text wraps between characters at the end of a line.

> ***Note:*** This property is specific to the iOS platform.

Text Copyable

Specifies whether label text can be copied.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget.

Actions

Actions define what happens when an event occurs. On an Image widget, you can run an action when the following event occurs:

- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

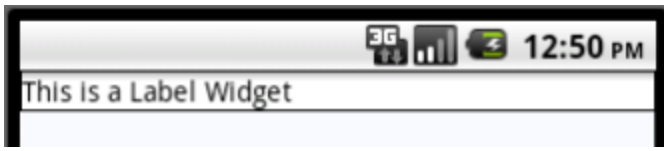
The following table summarizes where an Image widget can be placed:

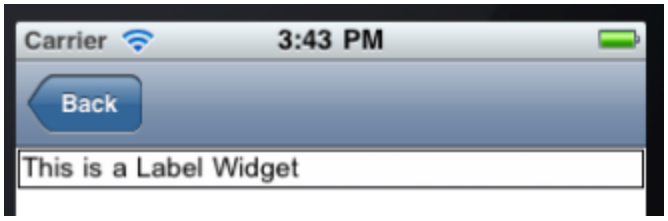
Flex Form	Yes
VBox Form	Yes

FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	Yes
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Label widget on various platforms is as follows:

Platform	Appearance
Android	

Platform	Appearance
iOS	
SPA	

ListBox

Use a List Box widget to display a drop-down list of items. You can then click on an item to select it. You can now add a list box to a segment. When you add a list box to a segment, you can:

- Assign data to the list box easily as when you add the list box to a segment, the widget data map is automatically assigned to the list box.
- Dynamically assign data to the list box using the mandated format for list box. i.e. `[["key1", "value1"],["key2", "value2"],["key3", "value3"]]`

When the list box is selected, the wheel pops-up in iOS. The segment row will not cover/hide the row. If the segment height isn't enough for the list box to expand, the segment expands to display the list box. This may distort the UI temporarily.

When the onRowClick event is triggered in a segment, the selected list box data is passed on to the action. If the default option is mentioned, then it is passed on in ["key1","value1"] format. If a default option is not mentioned, then null is assigned to the list box value. The list box is activated only when the segment row is in focus. You cannot open multiple list-boxes simultaneously.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Example

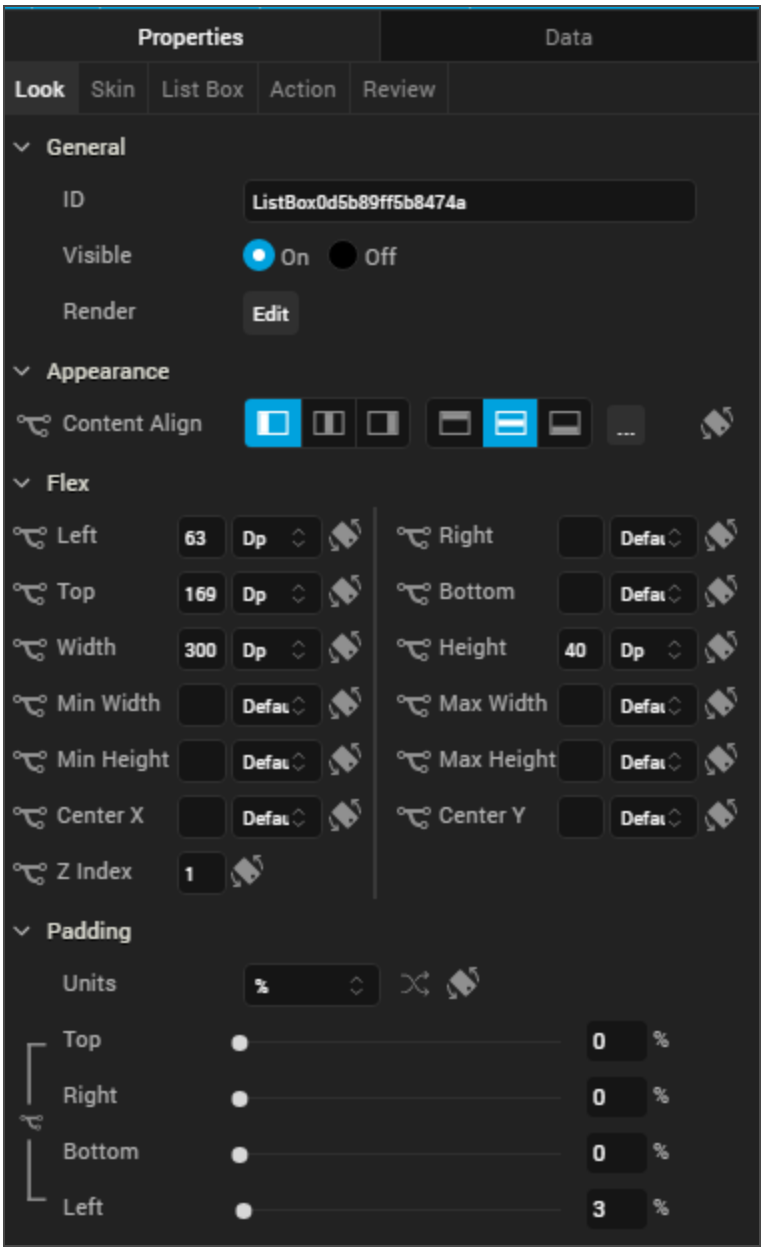
```
var masterData = [{
        "ListBoxadd": {
            "masterData": [
                ["lb1", "ListBox One"],
                ["lb2", "ListBox Two"],
                ["lb3", "ListBox Three"]
            ],
            "selectedKey": ["lb1"]
        },
    }, {},{}];
    this.view.DataSegment.rowTemplate = "rowFlex";
    this.view.DataSegment.sectionHeaderTemplate = "";
    this.view.DataSegment.widgetDataMap =  {
                    "rowFlex": "rowFlex",
                    "ListBoxadd": "ListBoxadd"
                };
    this.view.DataSegment.setData(masterData);
```

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.

- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For a ListBox widget, you can apply a skin for the following states:

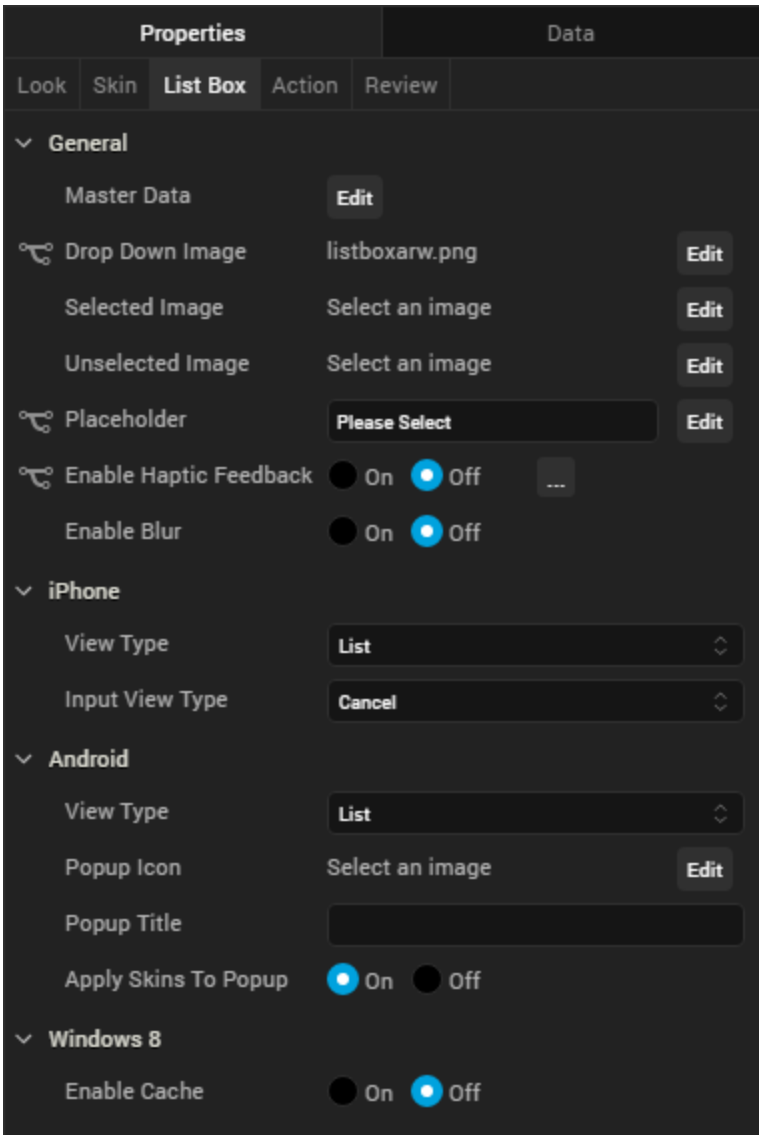
Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the focus is on the widget.
Native Field	The skin applied to each item in the ListBox popup window.
Native Field Focus	The skin applied when an item in the ListBox popup window has the focus.
Blocked UI	The skin applied to block the interface until the action in progress (for example, a service call) completes. ***Note:*** The Blocked UI skin property is available only for SPA platforms.
Placeholder	Reads the font color set in the skin and ignores other attributes. Android does not support setting a background color for a placeholder.
Hover Skin	The look and feel of a widget when the cursor hovers over the widget. ***Note:*** Hover skins is available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

ListBox Properties

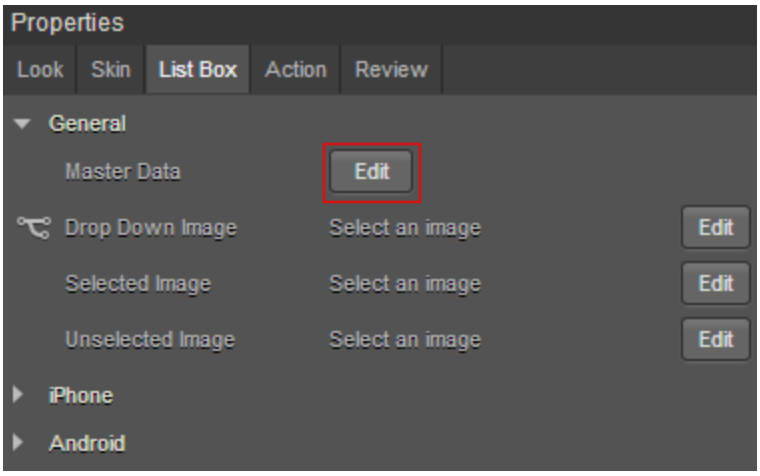
ListBox properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

> ***Note:*** In this section, properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Master Data

Specifies the set of values displayed in the list box. To enter or add values, click the **Edit** button to open the **Master Data** dialog box.



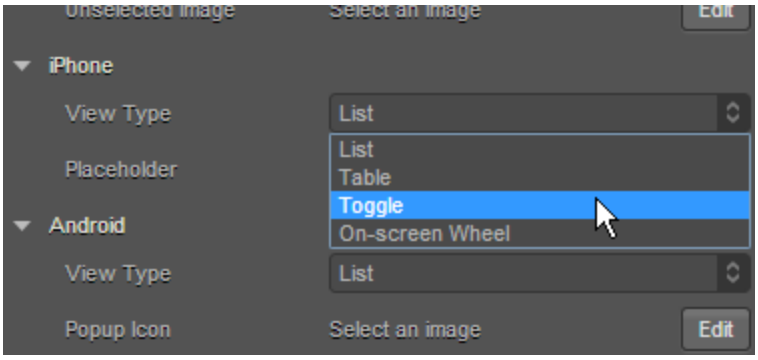
The **Master Data** dialog box contains the following columns:

- Key: The unique identifier of each list box entry.
- Display Value: The values displayed in the list box.
- Select Key : Specifies the default list box value.

To add List box values, click **Add**. To delete a list box entry, click inside a cell, and then click **Delete**. Click **Apply** to create the master data.

View Type

Specifies the list box view mode. Select a view type from the View Type drop-down list under either the iPhone or Android properties. The options available for iOS and Android vary.



View Types for iOS

Following are the options available for iOS platform:

- **List.** Displays the list items in a conventional, drop-down format.
- **Table.** Displays all available list items. The selected item has a check mark next to it.
- **Toggle.** Displays all available items as a series of side by side segments. The background of the selected item is a different color from the other options. This color is referred to as the Tint Color, and you can change it by setting Enable Tint Color to **On**, and then selecting the color you want from the color palette. In addition, you can select one of three styles for the toggle: Plain, Bordered, or Bar. To distribute the segments in equal proportions, set the Equal Segments option to **On**.
- **On-screen Wheel.** Similar to the iOS Picker control, presents the list options as a cylinder that you rotate to make your selection.

View Types for Android

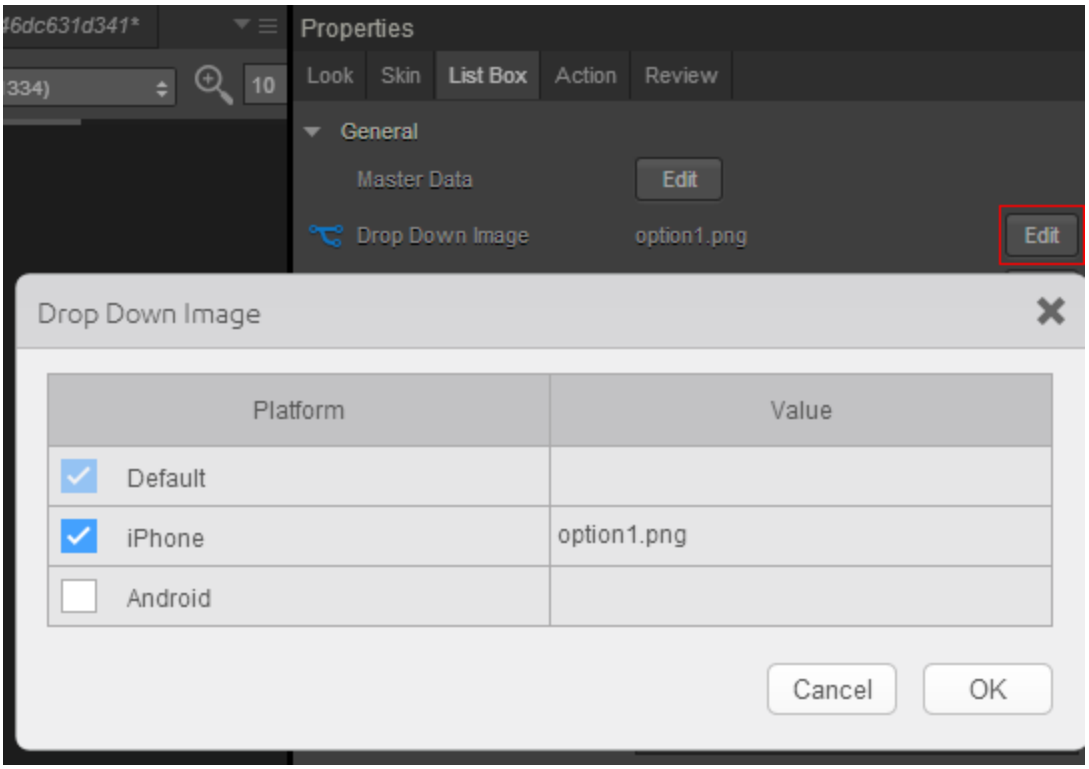
Following are the options available for the Android platform:

- **List.** Displays the list items in a conventional, drop-down format.
- **Spinner.** Displays the list options as though they're on a cylinder that you rotate to make your selection.

Drop Down Image

Specifies the image used for the drop-down box indicator. The default is an inverted triangle.

To specify a different image or a platform-specific image, click the **Edit** button to open the **Drop Down Image** dialog box.



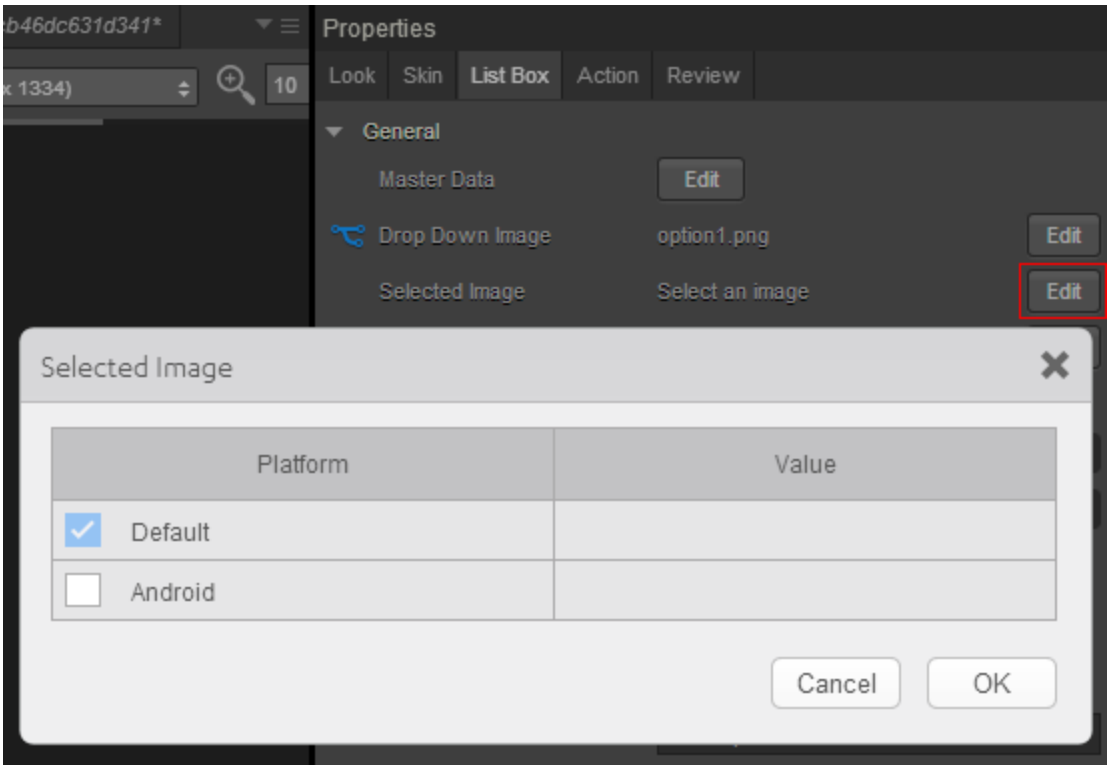
Select the desired platform and click inside corresponding **Value** field. From the **Select Image** dialog box, you can either select an available image or provide an image URL.

Selected Image

Specifies the image to be displayed when you make a selection.

> ***Note:*** If you specify a **Selected Image**, make sure to also specify an **Unselected Image**.

To specify a default image or a platform-specific image, click the **Edit** button to open **Selected Image** dialog box.



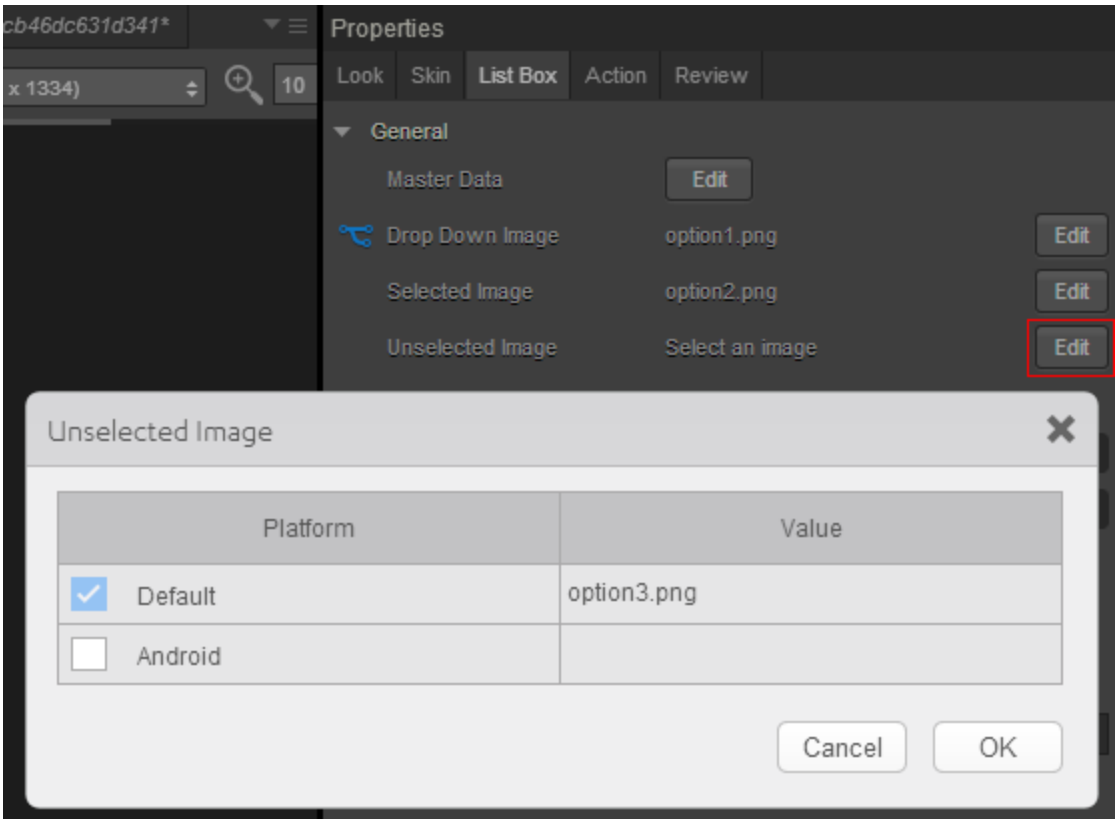
Select the desired platform and click inside corresponding **value** field. From the **Select Image** dialog box, you can either select an available image or provide an image URL.

Unselected Image

Specifies the image to be displayed when a selection is cleared.

> ***Note:*** If you specify a **Selected Image**, make sure to also specify an **Unselected Image**.

To specify a default image or a platform-specific image, click the **Edit** button to open the **Unselected Image** dialog box.



Select the desired platform and click inside corresponding **value** field. From the **Select Image** dialog box, you can either select an available image or provide an image URL.

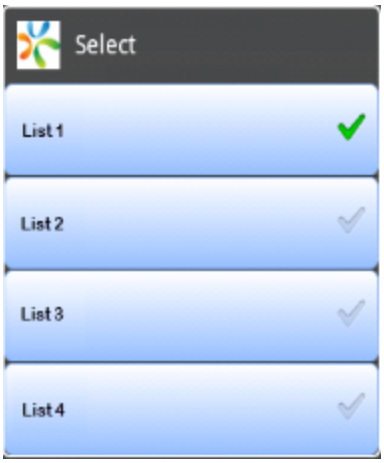
Placeholder

Specifies temporary or substitute text displayed until a selection is made; for example, a hint provided as a word or phrase.

> ***Note:*** This property is specific to the iOS platform.

Popup Icon

Specifies the icon that appears on the top left of the popup window's title area.



To select a popup icon, click the **Edit** button to open the **Popup Icon** dialog box. Select an image and click **OK**.

> ***Note:*** This property is specific to the Android platform.

Popup Title

Specifies the list box title text.

> ***Note:*** This property is specific to the Android platform.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a ListBox widget, you can run an action when the following event occurs:

- onSelection: The action is triggered when an item is selected.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.

- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

The following table summarizes where a ListBox widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

RadioButtonGroup

Use a RadioButtonGroup widget to select from a group of radio buttons when a user can choose only one option.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a RadioButtonGroup Widget.

All Platforms

- A RadioButtonGroup widget is always a group widget.
- Use a RadioButtonGroup widget if there are a limited number of possible selections and you can make only one selection. If you can make more than one selection from the group, use a CheckBoxGroup widget. To display a list of selections, use a ListBox widget.

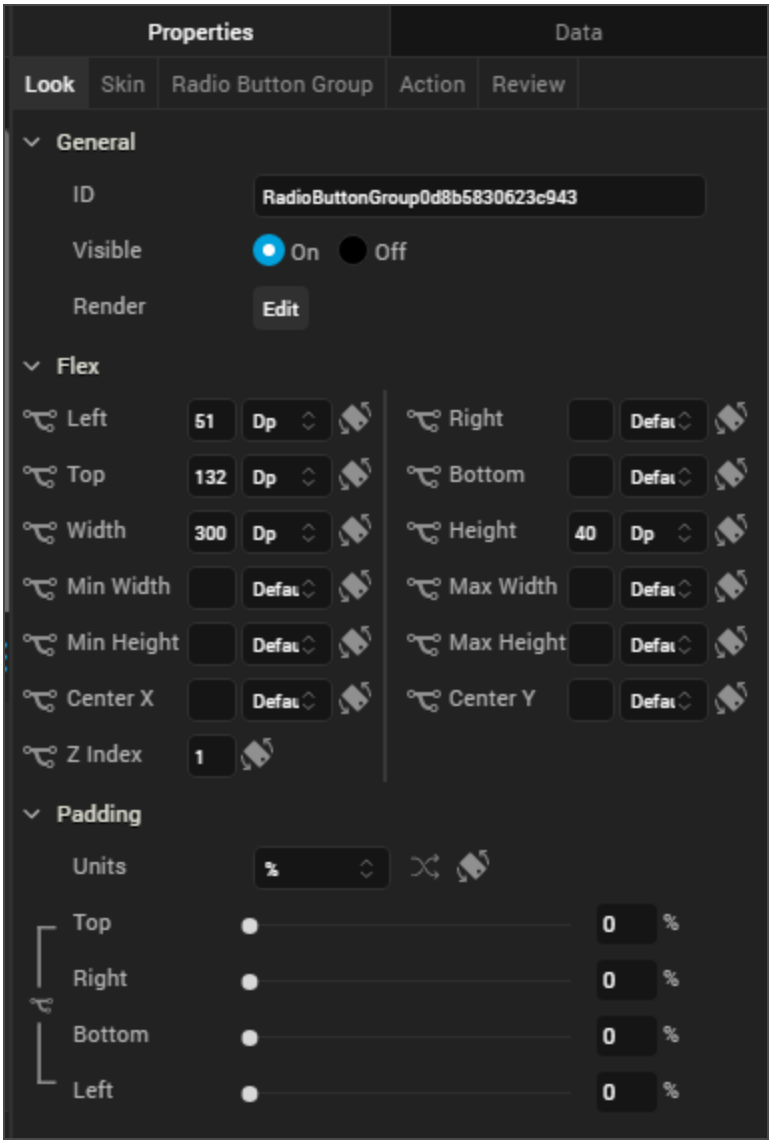
Android

- If you set the Orientation property to horizontal, do not place more than two items in the group. If you place more than two items and the associated text is large, additional items may not fit in the screen width and will not be visible.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

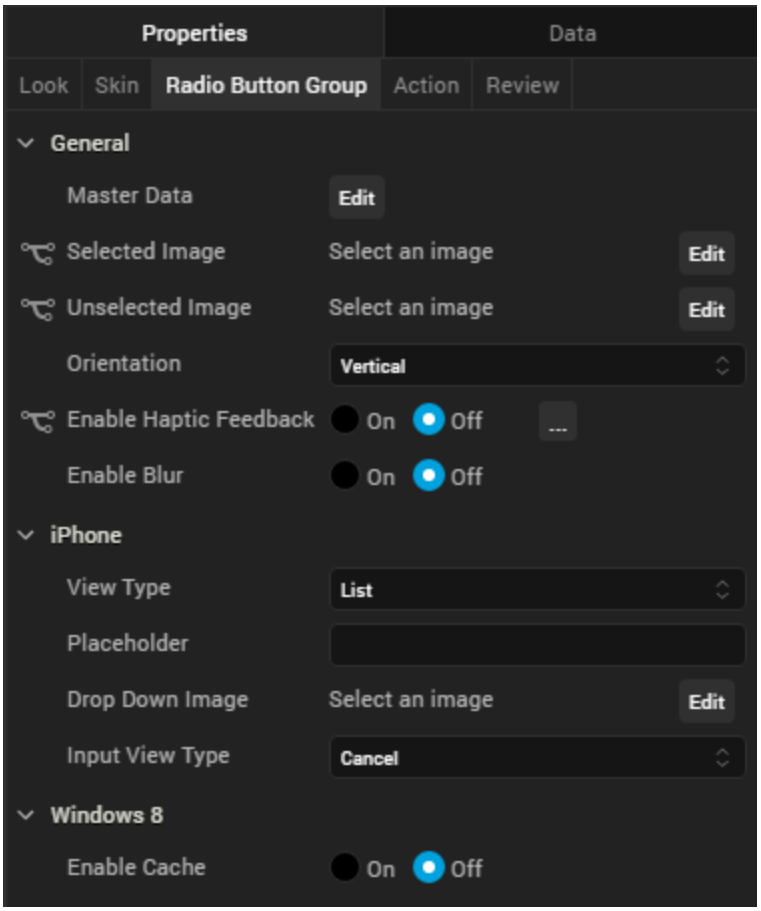
For the RadioButtonGroup widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the focus is on the widget.
Hover Skin	The look and feel of the widget when the cursor hovers over the widget. ***Note:*** Hover Skins is available only on Windows (native) Tablet devices.

For more information about applying skins, see Understanding Skins and Themes.

RadioButtonGroup Properties

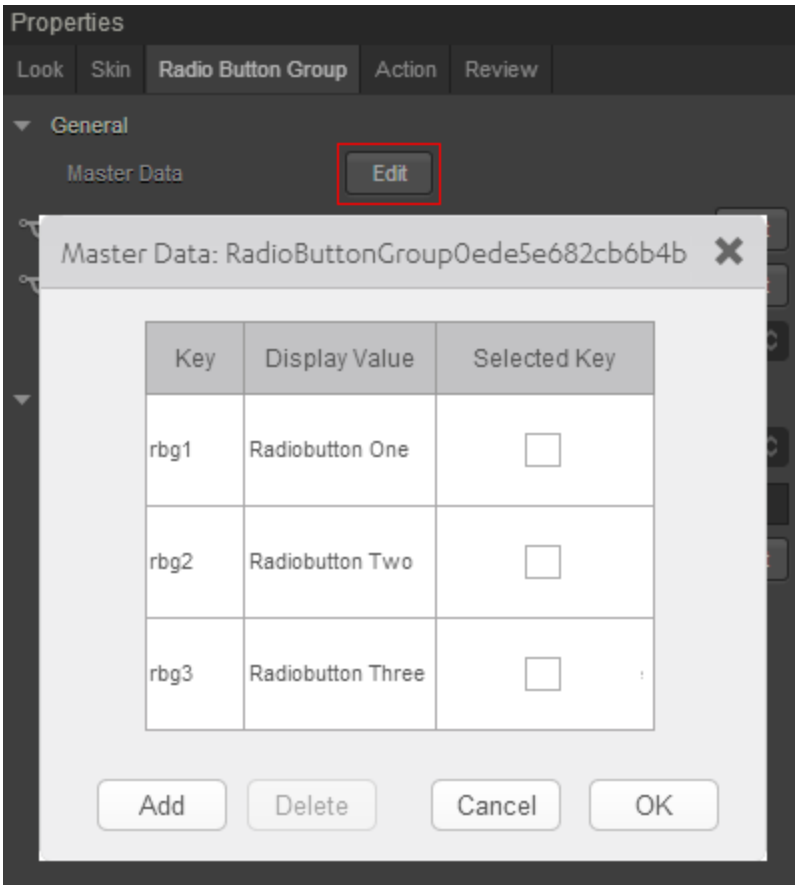
RadioButtonGroup properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Master Data

Specifies the set of values that must be displayed for the user to make a selection from the available choices.

To specify this set of values, click the **Edit** button of the **Master Data** field to open the **Master Data** dialog box.



The Master Data dialog box contains the following columns:

- Key: The unique identifier of each radio button.
- Display Value: The label or descriptive text displayed for each radio button.
- Select Key : Whether a radio button is selected by default.

To add more radio buttons to the widget, click **Add**. To delete a radio button, click inside a cell, and then click **Delete**.

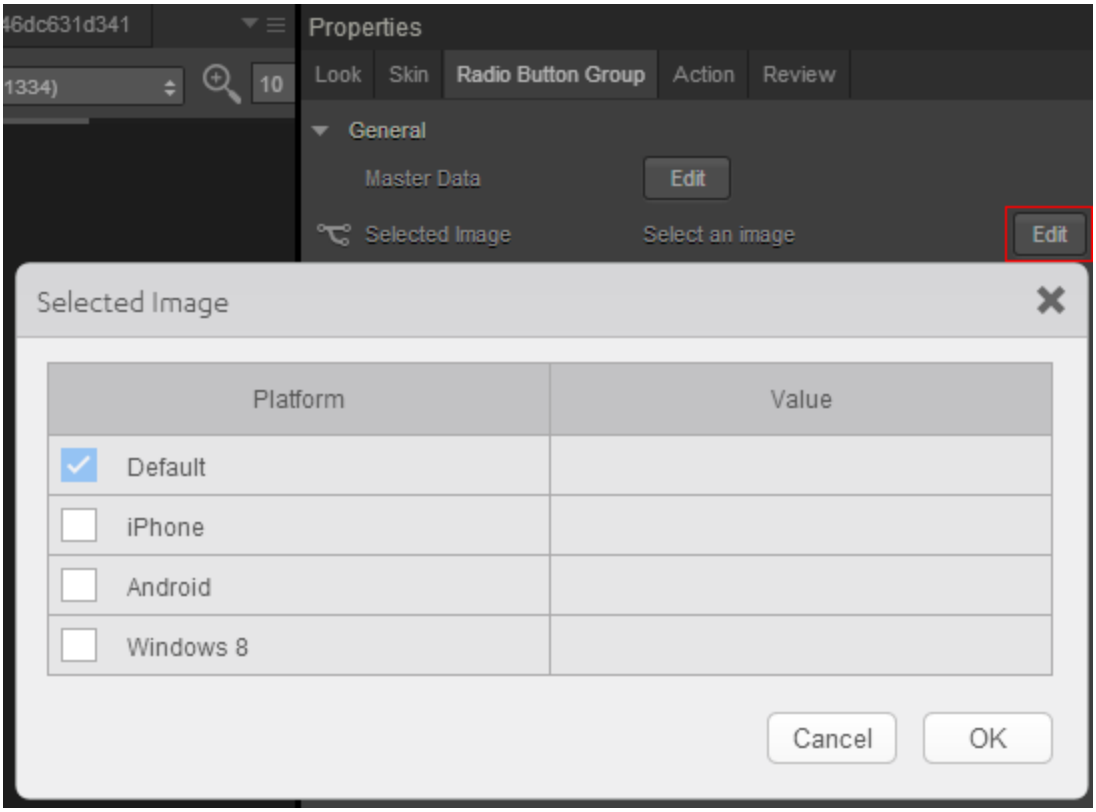
Click **OK** to create the master data.

Selected Image

Specifies the image to be displayed when you make a selection.

> ***Note:*** If you specify a **Selected Image**, make sure to also specify an **Unselected Image**.

To provide a default or platform-specific image, click the **Edit** button to open the **Selected Image** dialog box.



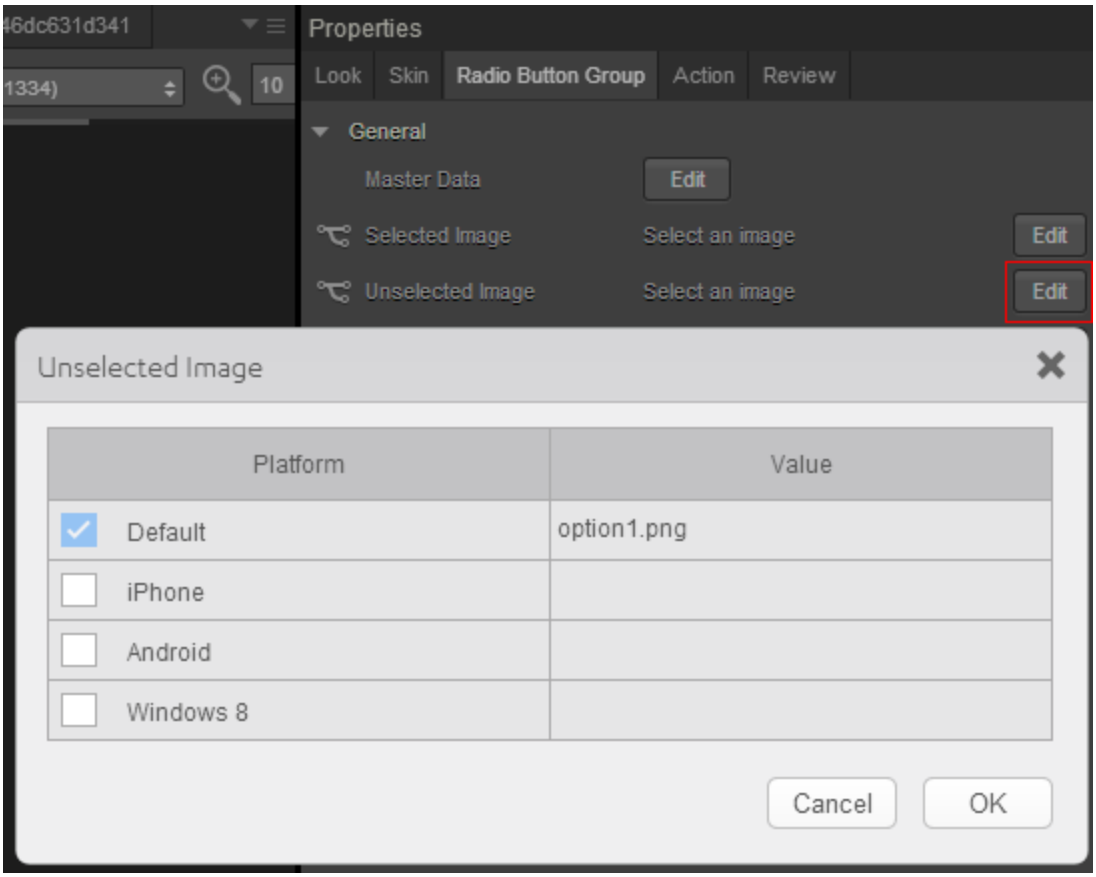
To provide a platform-specific image or replace the default image, select the platform and click inside the corresponding **Value** field to open the **Selected Image** dialog box. You can either:

- Select an available image.
- Provide an image URL.

Unselected Image

Specifies the image to be displayed when a selection is cleared.

To provide a default or platform-specific image, click the **Edit** button to open the **Unselected Image** dialog box.



To provide a platform-specific image or replace the default image, select the platform and click inside the corresponding **Value** field to open the **Selected Image** dialog box. You can either:

- Select an available image.
- Provide an image URL.

Orientation

Specifies whether the alignment of the radio buttons is horizontal or vertical.

Default: Vertical

View Type

For the iOS platform, specifies the view type of the RadioButtonGroup , either List, Table, Toggle, or On-screen Wheel.

Default: List

> ***Note:*** If you select the On-screen Wheel view type, you cannot view the on-screen wheel on the Visualizer canvas.

Drop Down Image

If the view type is List, specifies the image used for the drop-down box indicator. The default is an inverted triangle.

To specify a different image or a platform-specific image, click the **Edit** button to open the **Drop Down Image** dialog box.

From the **Drop Down Image** dialog box, select an image and click **OK**.

Group Cells

If the view type is Table, specifies whether the Group Cells style is applied. The Group Cells style groups items in the radio button group.

Default: Cells are not grouped.

View Style

If the view type is Toggle, specifies the view style of the toggle button, either Plain, Bordered, or Bar.

Default: Plain

Equal Segments

If the view type is Toggle, specifies whether to distribute the segments in equal proportions.

Default: Segments are distributed in equal proportions.

Enable Tint Color

If the view type is Toggle, specifies whether to enable a tint color.

Default: Tint color is not enabled.

Tint Color

If tint color is enabled, specifies the tint color. To select a tint color, click the color picker to open the color selection dialog box, and then select a color.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a RadioButtonGroup widget, you can run an action when the following event occurs:

- onSelection: The action is triggered when an item is selected.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see the topic, Add Actions.

Placement Inside a Widget

The following table summarizes where a RadioButtonGroup widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes

VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

RichText

Use a RichText widget to display formatted text. The RichText widget uses HTML formatting tags to display text with links, images, and character styles such as bold and italic.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

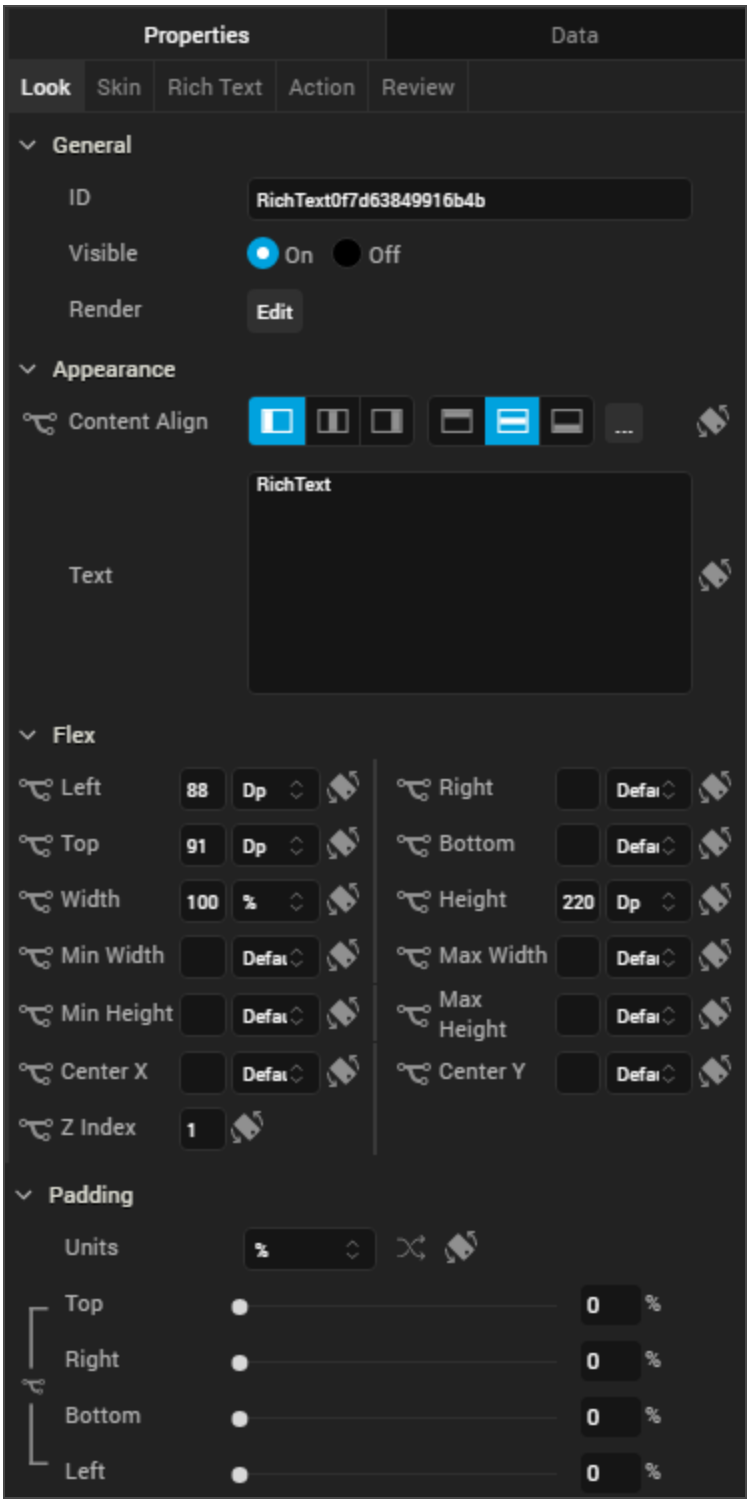
The following is an important considerations for a RichText widget.

- If you specify a skin for a RichText widget, font level settings such as color style and size are applied to the entire content of the widget. Use the label style HTML formatting tag to override the text color specified by the skin.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For a RichText widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of a widget.
Link	The skin applied to a link in the RichText widget.
Link Focus	The skin applied to link when it has the focus.
Telephone Link	The skin applied to a telephone link in the RichText widget. ***Note:*** Telephone Link skins are available only on the Windows platform.
Hover	The look and feel of the widget when the cursor hovers over the widget. ***Note:*** Hover skins are available only on the Windows (native) Tablet platform.
Super Script	The skin applied to superscripts in the RichText widget. ***Note:*** Super Script skins are available only on the Windows platform.

For more information about applying skins, see Understanding Skins and Themes.

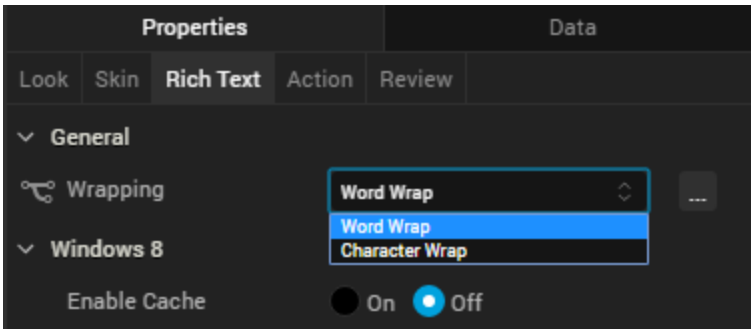
Rich Text Properties

Rich Text properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

Rich Text Wrapping

Specifies how the label wraps text, either Word Wrap or Character Wrap:

- Word Wrap (Default): Text wraps between words at the end of a line.
- Character Wrap: Text wraps between characters at the end of a line.



> ***Note:*** This property is specific to the iOS platform.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a RichText widget, you can run an action when the following event occurs:

- onClick: The action is triggered when the user clicks on the widget.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see the topic, Add Actions.

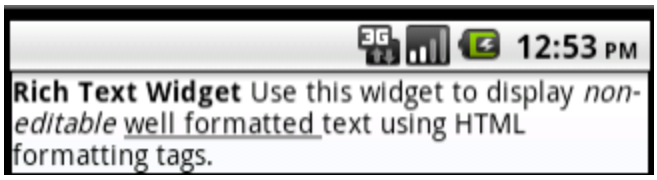
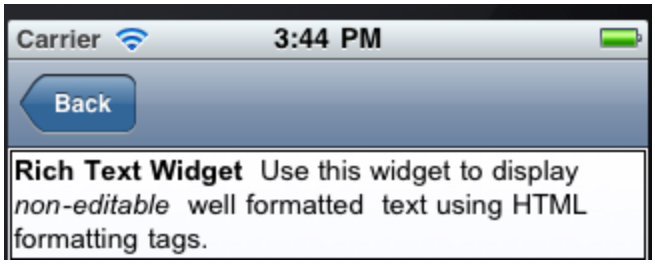
Placement Inside a Widget

The following table summarizes where a RichText widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	Yes
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the RichText widget varies as follows:

Platform	Appearance
Android	
iOS	
SPA	

Slider

Use a Slider widget to select from a range of values by moving a slider indicator horizontally. You can display minimum and maximum values, and update the displayed value when you drag the indicator along the slider.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

> ***Note:*** The Slider widget is not supported in SPA platforms.

Important Considerations

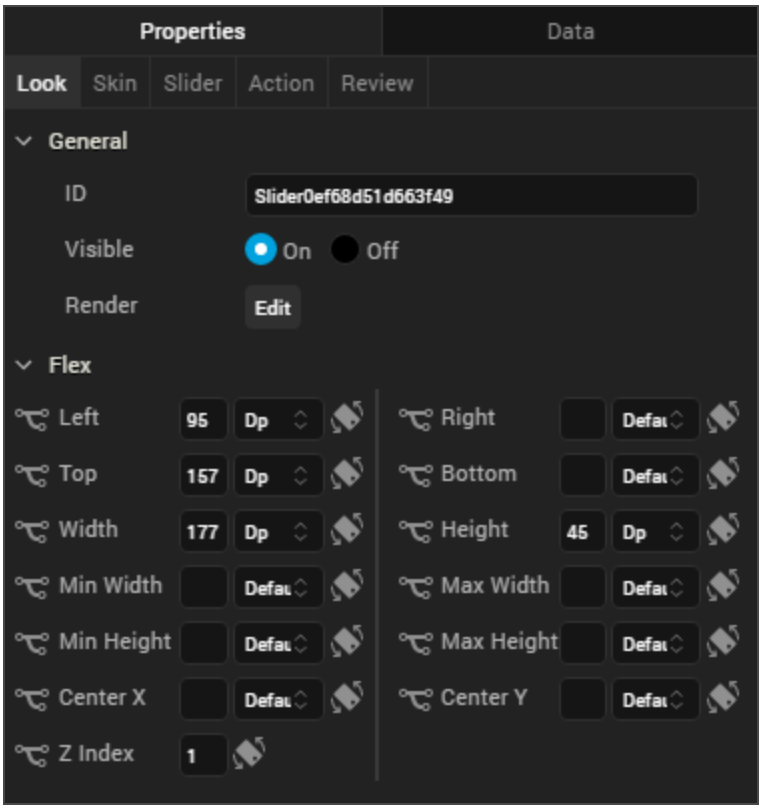
The following are important considerations for a Slider widget.

- All platforms (except iPhone): Use the Min Label and Max Label properties to specify minimum and maximum values.
- iPhone: Use the Min Value Image and the Max Value Image properties to display images of minimum and maximum values.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For a Slider widget, you can apply a skin and its associated properties for the following states:

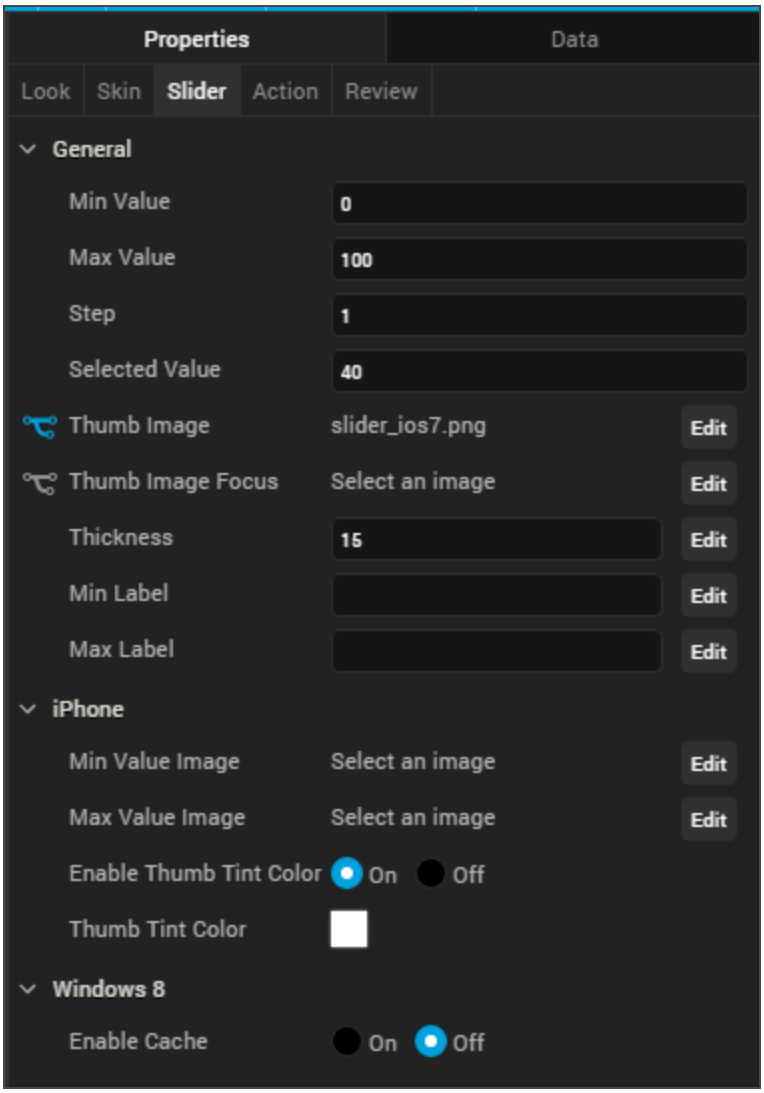
Skin	Definition
Left	The skin applied to the background of the slider on left side of the thumb image.
Right	The skin applied to the background of the slider on right side of the thumb image.
Min Label	The skin applied to the min property of the slider.
Max Label	The skin applied to the max property of the slider.

For more information about applying skins, see Understanding Skins and Themes.

Slider Properties

Slider properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

> ***Note:*** In this section, the properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Min Value

Specifies the minimum value that you can select.

Default: 0

Max Value

Specifies the maximum value that you can select.

Default: 100

Step

Specifies the increment by which the slider value is increased or decreased between the minimum and maximum values.

Default: 1

You can specify a value between the difference of the maximum and minimum values of the slider; for example, if the minimum value is 40 and the maximum value is 45, the step value can be between 1 and 5.

Selected Value

Specifies the value that is displayed as the selected value.

Default: 40

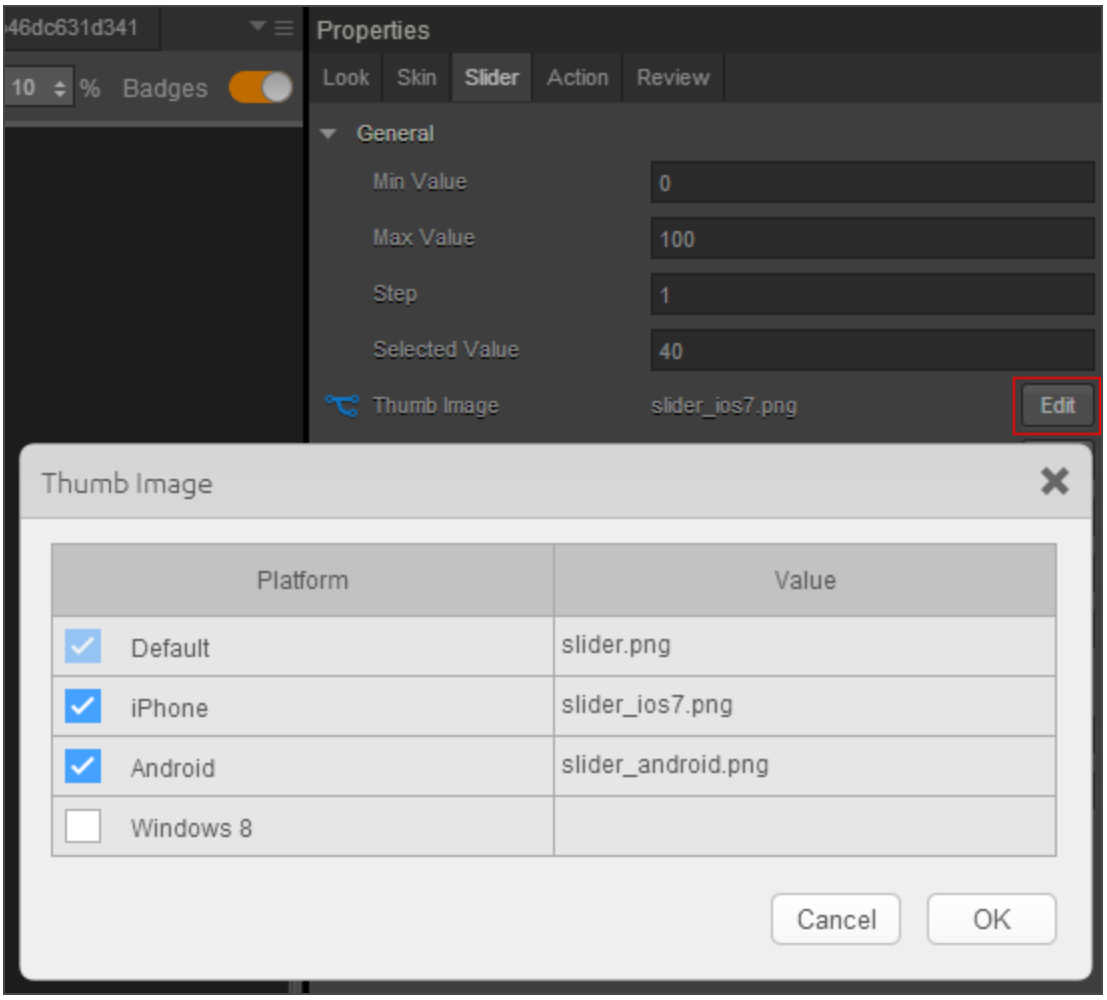
If you do not specify a selected value, the value displayed is the minimum value plus half the difference between the minimum and the maximum value. For example, if the minimum value is 0 and the maximum value is 100, the value displayed is 50.

If you specify a selected value that is less than the minimum value, the selected value uses the minimum value. If you specify a selected value that is greater than the maximum value, the selected value uses the maximum value.

Thumb Image

Specifies the image used for the indicator, or thumb.

To provide a default or platform-specific image, click the **Edit** button to open the Thumb Image dialog box.

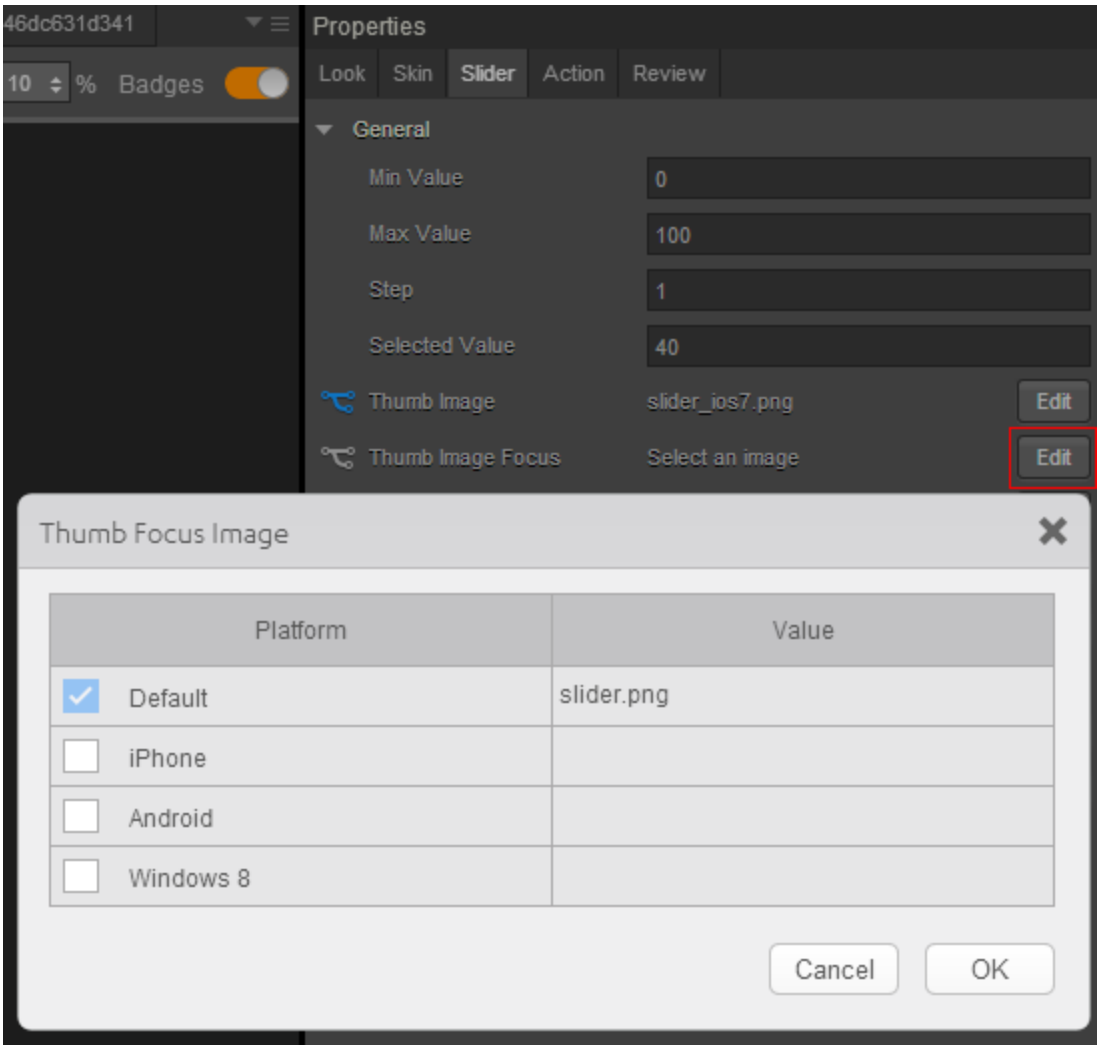


To provide a platform-specific image or to replace the default image, select the desired platform and click inside the **Value** field to open the **Select Image** dialog box. You can either select an available image or provide an image URL.

Thumb Image Focus

Specifies an image to indicate that the thumb has the focus.

To provide a default or a platform-specific image, click the **Edit** button to open the **Thumb Focus Image** dialog box.



To provide a platform-specific image or to replace the default image, select the desired platform and click inside the **Value** field to open the **Select Image** dialog box. You can either select an available image or provide an image URL.

Thickness

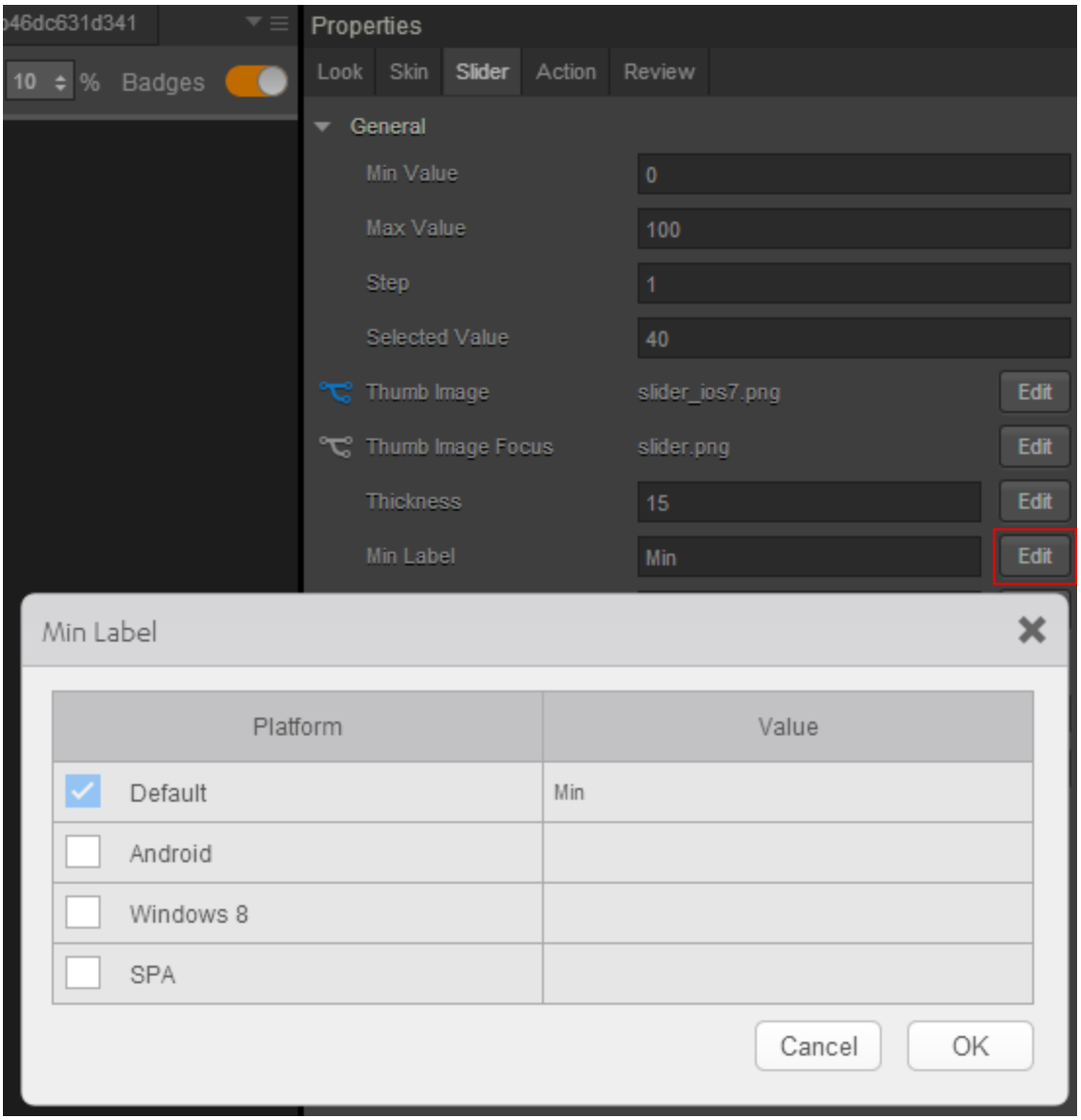
Specifies the thickness of the seek bar.

Default: 15

Min Label

Specifies the text displayed below the minimum value of the slider. This text become the default for all platforms.

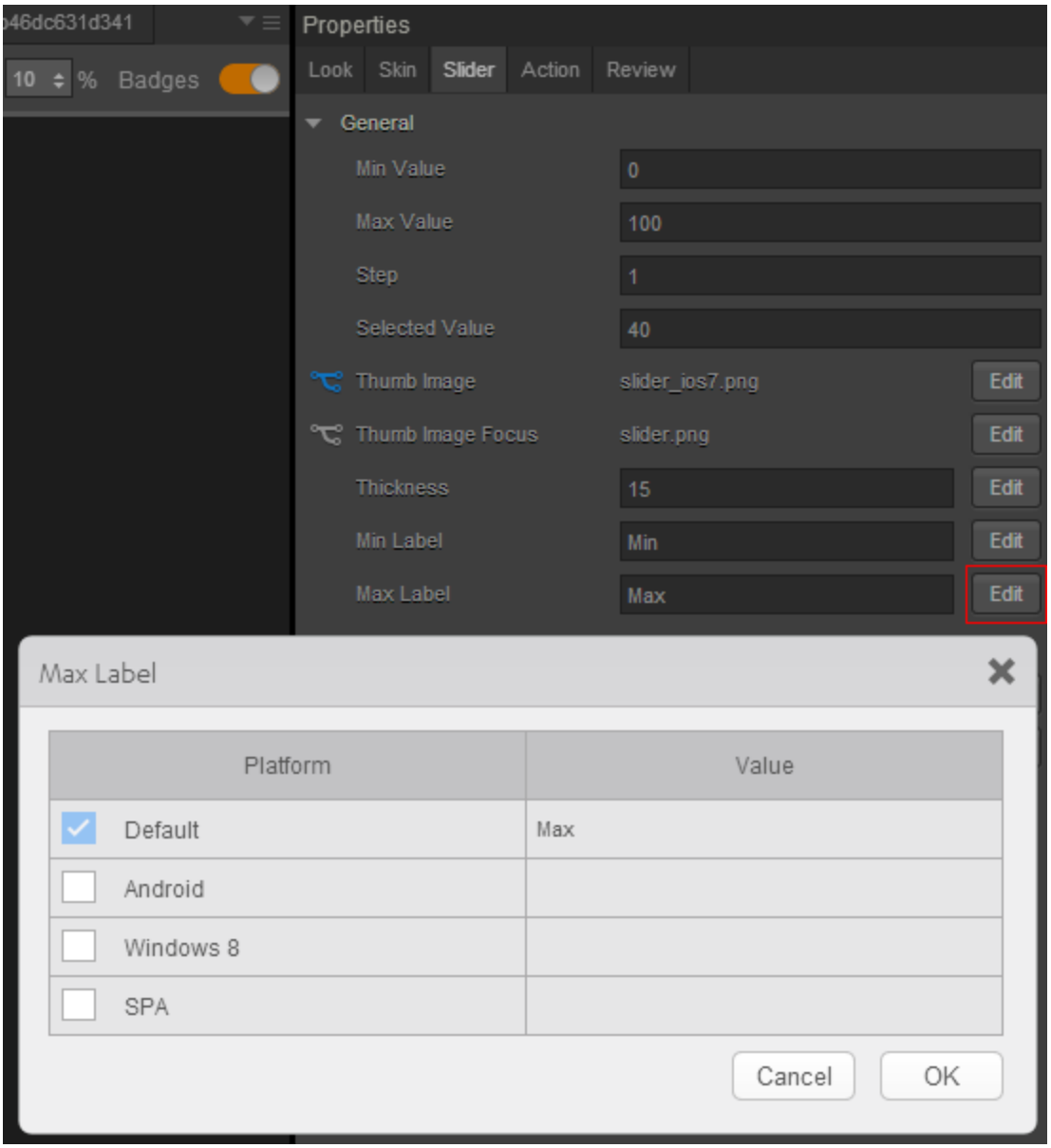
To provide a platform-specific label, click the **Edit** button to open the **Min Label** dialog box.



Max Label

Specifies the text displayed below the maximum value of the slider. This text become the default for all platforms.

To provide a platform-specific label, click the **Edit** button to open the **Max Label** dialog box.



Min Value Image

For the iOS platform, specifies an image for the minimum value of the slider.

Click the **Edit** button to open the **Min Value Image** dialog box, and either select an available image or provide and image URL.

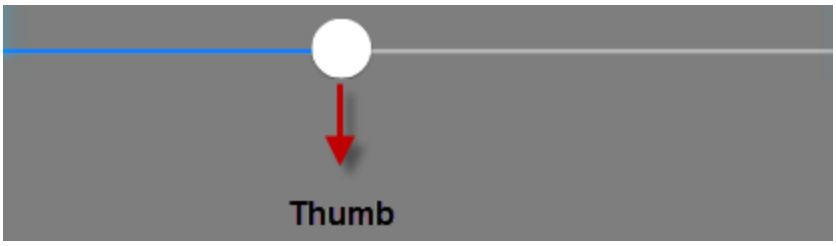
Max Value Image

For the iOS platform, specifies the image for the maximum value of the slider.

Click the **Edit** button to open the **Max Value Image** dialog box, and either select an available image or provide and image URL.

Enable Thumb Tint Color

For the iOS platform, specifies whether to enable a color for the slider thumb.



Thumb Tint Color

For the iOS platform, specifies a tint color for the thumb if tint color is enabled. To select a tint color, click the color picker to open the color selection dialog box, and then select a color.

Actions

Actions define what happens when an event occurs. On a Slider widget, you can run an action when the following events occur:

- onSlide: The action is triggered when the slider moves.
    - For touch-based devices, the action is triggered when you stop sliding the thumb indicator.
    - For non touch-based devices, the action is triggered when the left or right key is released.
- onSelection: The action is triggered when you makes a selection.
    - For touch-based devices, the action is triggered when you stop sliding the thumb indicator.
    - For non touch-based devices, the action is triggered when the left or right key is released.

- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see the topic, Add Actions.

Placement Inside a Widget

The following table summarizes where an Image widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes

Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Slider widget on varies as follows:

Platform	Appearance
Android	
iPhone	

TextArea2

Use a TextArea widget to enable a user to enter multiple lines of text. For example, you can add a TextArea widget to the **Feedback** section of an application to enable users to enter comments.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a TextArea Widget.

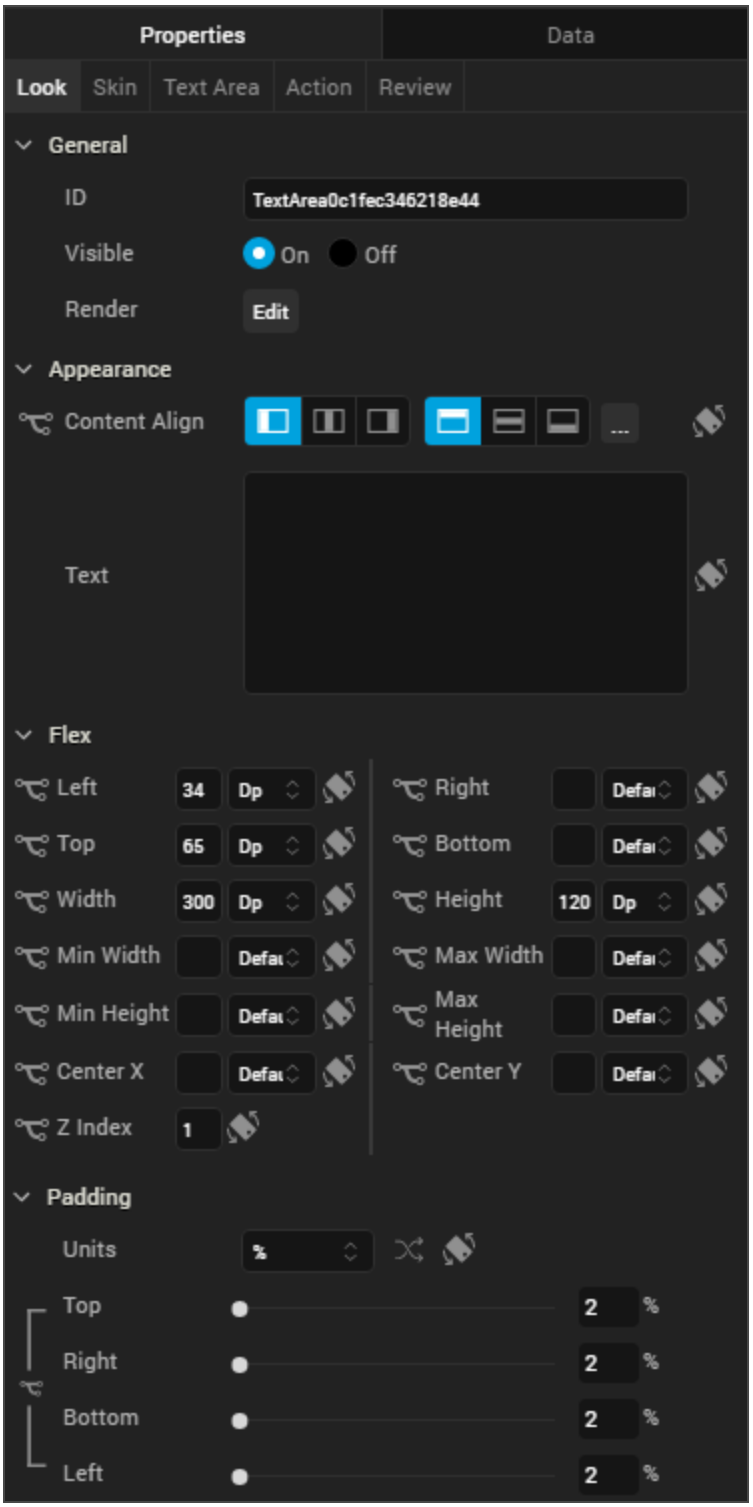
- Editing on devices with a small form factor takes place on a new screen.
- Editing on devices with a medium or large form factor takes place on the same screen.

- In Mobile web, some browsers by default enable a vertical or horizontal scroll bar for the text area, even if the number of lines are less than number of visible lines. The Mobile Web platform does not control whether the scroll bars appear.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the TextArea widget, you can apply a skin and its associated properties for the following states:

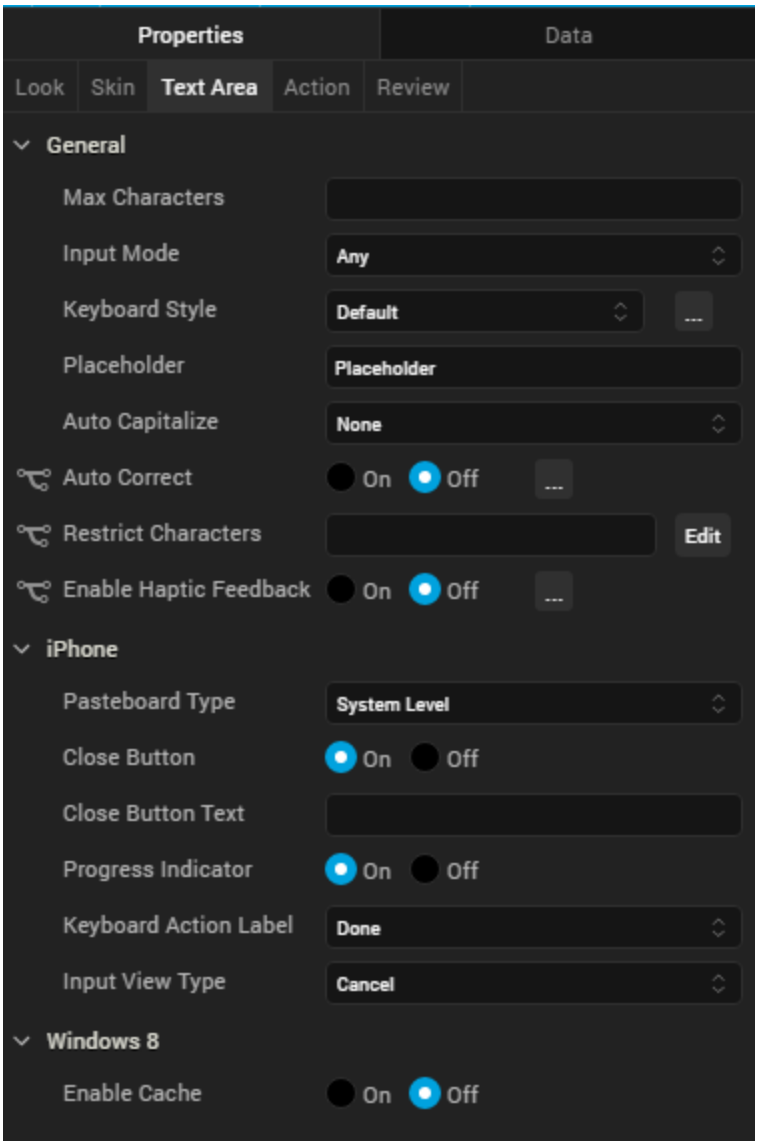
Skin	Definition
Normal	The default skin of a widget.
Focus	The skin applied when the widget has the focus.
Blocked UI	The skin applied to block the interface until the action in progress (for example, a service call) is completed. ***Note:*** The Blocked UI skin is available only for SPA platforms.
Placeholder	The skin applied to placeholder text in the widget. Only the font color skin attribute is applicable.
Hover Skin	The look and feel of a widget when the cursor hovers over the widget. ***Note:*** The Hover skin is available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

Text Area Properties

TextArea properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

> ***Note:*** In this section, the properties that can be forked are identified by an icon ( ) located to the left of the property. For more information, see Fork a Widget Property.



Max Characters

Specifies the maximum number of characters that the text field can accept.

Input Mode

Specifies whether the text area accepts any characters or only numeric values. The Numeric Only option is not supported on server-side Mobile Web platforms.

Keyboard Style

Specifies the style of keyboard displayed when a user enters text or numeric values in the text area.

> ***Note:*** This property is specific to the Android platform.

Placeholder

Specifies placeholder text for the text area; for example, a hint that describes what should be entered.

Auto Correct

Specifies whether auto-correction is enabled. Click the ellipsis (...) button to make the Auto Correct property setting platform-specific.

> ***Note:*** You cannot execute this property on the Visualizer Canvas.

> ***Note:*** This property is specific to the iOS and the SPA platform.

Auto Capitalize

Specifies the character capitalization behavior.

Following are the options available:

- None: No action takes place on the input string.

  Example : This is sample text.

- Words: Changes the first character of all the words to uppercase. (Not supported on Mobile Web)

  Example : This Is Sample Text.

- Sentences: Changes the first character of all the sentences to uppercase.

  Example : This is sample text.

- All: Changes all the characters to uppercase. (Not supported on Mobile Web)

  Example : THIS IS SAMPLE TEXT.

Pasteboard Type

Enables an application to share data within the application or with another application using system-wide or application-specific paste boards.

Typically, an object in the application writes data to a pasteboard when the user requests a copy or cut operation on a selection in the user interface. Another object in the same or different application then reads that data from the pasteboard and presents it to the user at a new location; this usually happens when the user requests a paste operation.

> ***Note:*** You can only paste the text to a text area with the same pasteboard type as that of the source textbox. For example, if you set the Pasteboard type as *App Level Persistent* , you can paste the text only to another text area whose pasteboard type is also set to *App Level Persistent* .

> ***Note:*** This property is specific to the iOS platform.

Close Button

Specifies whether the keypad window displays a **Done** button.

> ***Note:*** This property is specific to the iOS platform.

Close Button Text

Specifies alternate text for the "Done" button. This property is available only when the *Close Button* is enabled.

> ***Note:*** This property is specific to the iOS platform.

Progress Indicator

Specifies whether to display a progress indicator showing that widget content is being loaded.

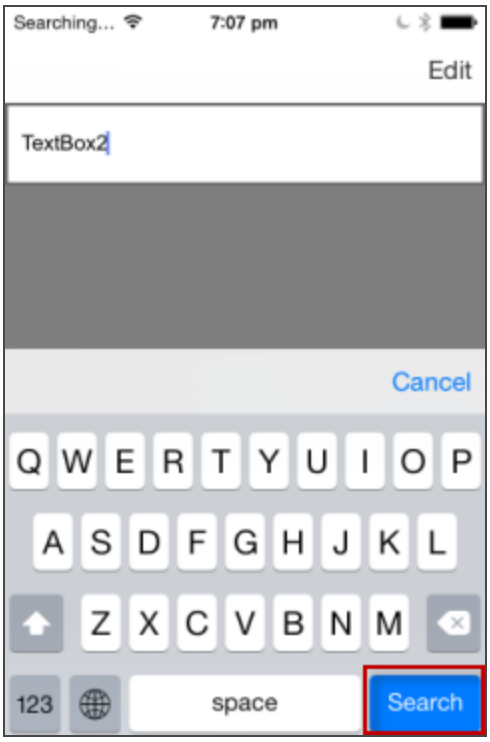
> ***Note:*** This property is specific to the iOS platform.

Keyboard Action Label

Specifies the text displayed on the action key of the keyboard.

Default: Done

For example, the following shows the keyboard action label set to "Search":



> ***Note:*** This property is specific to the iOS platform.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a TextArea widget, you can run an action when the following events occur:

- onTextChange: The action is triggered when text in the text area changes. The action is not triggered if the text changes programmatically.
- onDone: The action is triggered when the user has entered text and clicks or touches the action key.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.
- onBeginEditing: The action is triggered when the user clicks within the text area to start editing (iOS and Android).
- onEndEditing: The action is triggered when the user ends the editing process by clicking on another widget or the **Done** button (iOS and Android).

For more information, see Add Actions.

Placement Inside a Widget

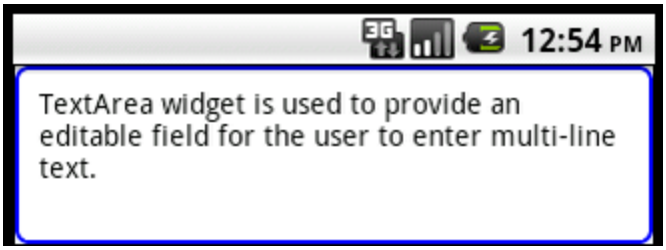
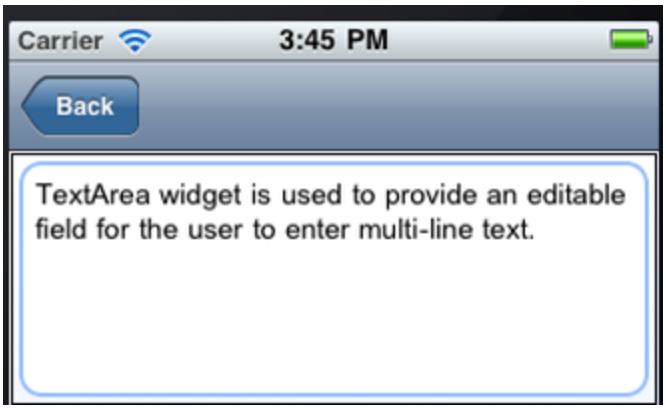
The following table summarizes where a TextArea2 widget can be placed:

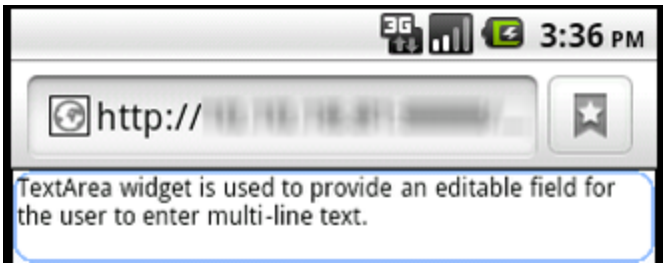
Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes

VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the TextArea2 widget varies as follows:

Platform	Appearance
Android	
iOS	

Platform	Appearance
SPA	

TextBox2

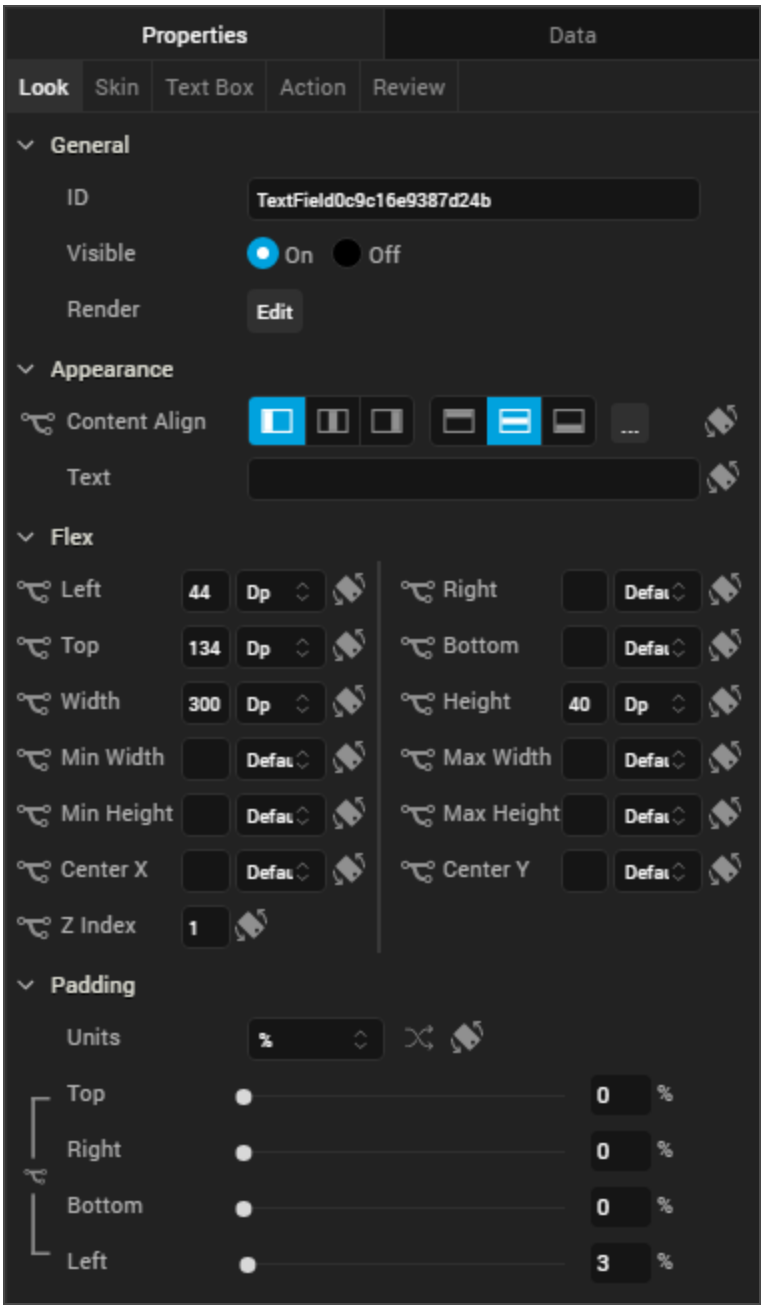
Use a TextBox widget to enable a user to enter single line of text. For example, you can add TextBox widgets to the **Login** page of an application to enable users to enter their login credentials.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the TextBox widget, you can apply a skin and its associated properties for the following states:

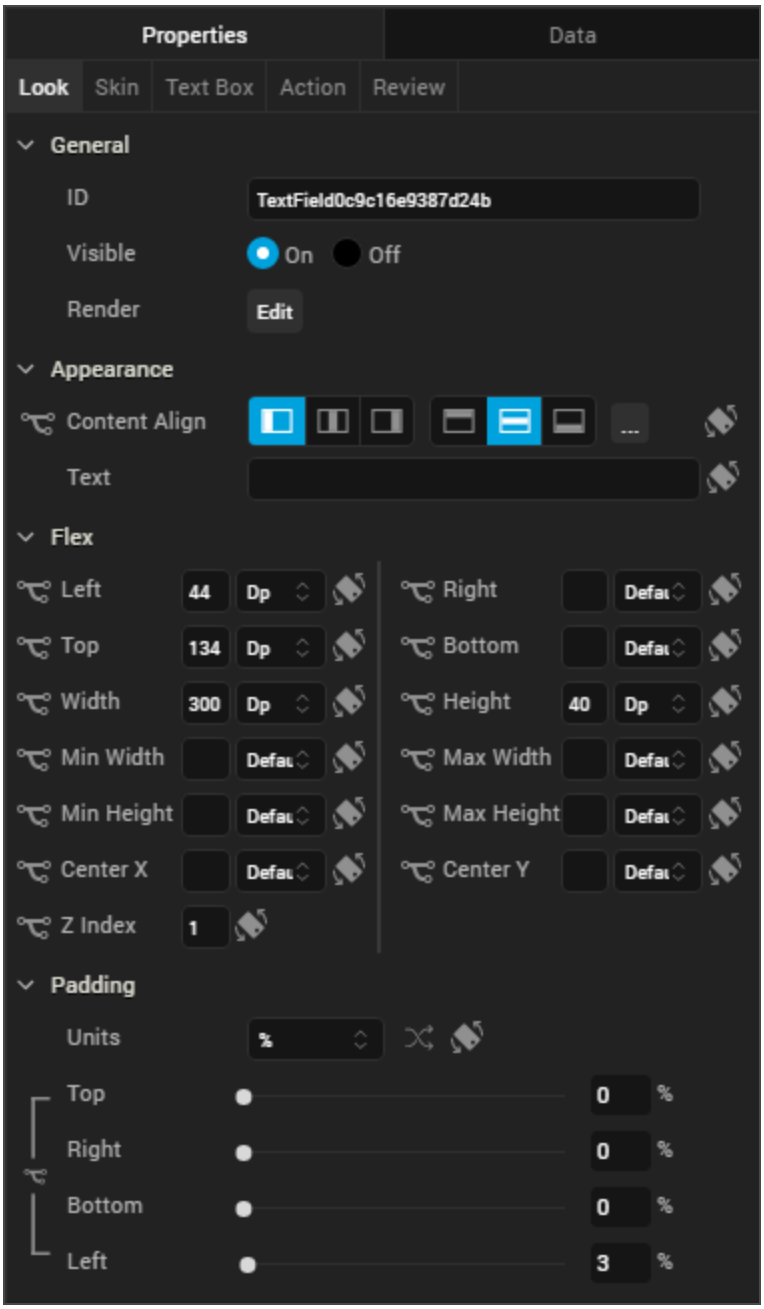
Skin	Definition
Normal	The default skin of a widget.
Focus	The skin applied when the widget has the focus.
Blocked UI	The skin applied to block the interface until the action in progress (for example, a service call) is completed. ***Note:*** The Blocked UI skin is available only for SPA platforms.
Placeholder	The skin applied to placeholder text in the widget. Only the font color skin attribute is applicable.
Hover Skin	Tthe look and feel of a widget when the cursor hovers over the widget. ***Note:*** The Hover skin is available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

TextBox Properties

TextBox properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

Note: In this section, the properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Mask Text

Specifies whether the text entered by the user is hidden by a mask character, such as asterisk or dot. This is typically enabled for a text box used to enter secure information, such as a password.

Default: Off (Mask Text is disabled)

Max Characters

Specifies the maximum number of characters that a user can enter in the text box.

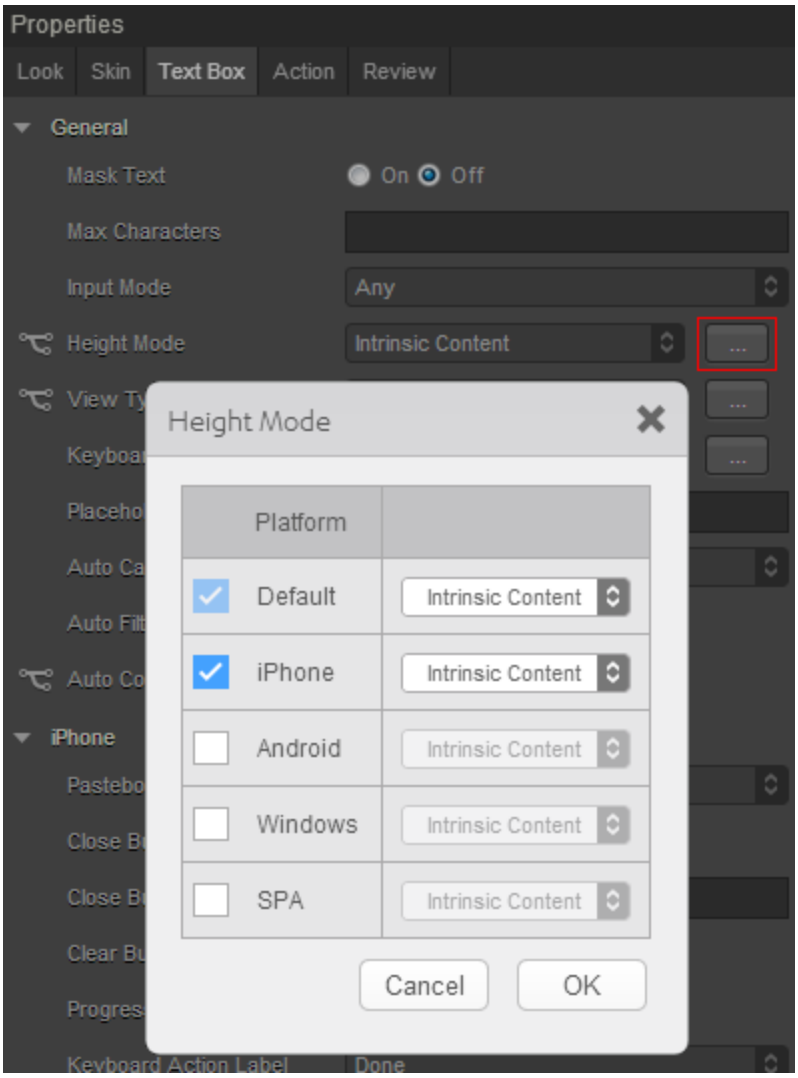
Input Mode

Specifies whether the text area accepts any characters or only numeric values.

Height Mode

Specifies how the text box height is determined, either by the intrinsic content or the system default. The default option is available only for iPhone and SPA platforms.

To provide a platform-specific or default value, click the Ellipsis button (...) to open the Height Mode dialog box, and then select a value.



Container Height

Specifies the dimensions of the text box, based on the Height Reference option. This property is available for non-flex forms when the Height Mode is Custom.

Height Reference

Specifies how the text box dimensions are determined:

- Form Reference: If the text box is not placed inside a popup or in templates, the height percentage is based on the height of the form, excluding headers and footers. T

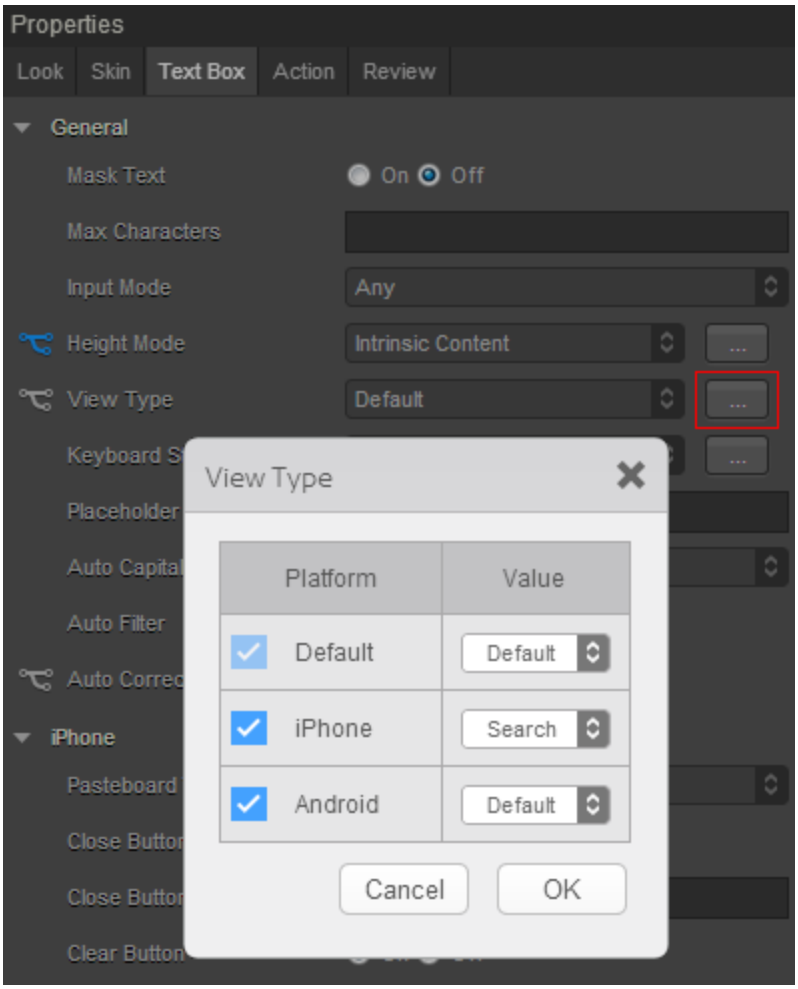
- Parent Width: If the text box is placed inside a popup or in templates, the width is based on the width of the parent container.

> ***Note:*** This property is unavailable on Flex Forms.

View Type

Specifies whether the text box can be used as a search field. The default setting is a text field.

The selected view type becomes the default for all the platforms. To provide a platform-specific value or a different default value, click the Ellipsis button (...) to open the **View Type** dialog box, and then select a value.



Keyboard Style

Specifies the style of keyboard displayed when a user enters text or numeric values in the text area.

> ***Note:*** This property is specific to the Android platform.

Placeholder

Specifies placeholder text for the text area; for example, a hint that describes what should be entered.

Auto Capitalize

Specifies the character capitalization behavior.

Following are the options available:

- None: No action takes place on the input string.

  Example : This is sample text.

- Words: Changes the first character of all the words to uppercase. (Not supported on Mobile Web)

  Example : This Is Sample Text.

- Sentences: Changes the first character of all the sentences to uppercase.

  Example : This is sample text.

- All: Changes all the characters to uppercase. (Not supported on Mobile Web)

  Example : THIS IS SAMPLE TEXT.

Auto Filter

Specifies whether characters entered in the text box are matched against the filter list, and possible matches are displayed.

Default: Off (input characters are not matched against the filter list)

Auto Correct

Specifies whether auto-correction is enabled. Click the ellipsis (...) button to make the Auto Correct property setting platform-specific.

> ***Note:*** You cannot execute this property on the Visualizer Canvas.

> ***Note:*** This property is specific to the SPA platform.

Pasteboard Type

Enables an application to share data within the application or with another application using system-wide or application-specific paste boards.

Typically, an object in the application writes data to a pasteboard when the user requests a copy or cut operation on a selection in the user interface. Another object in the same or different application then reads that data from the pasteboard and presents it to the user at a new location; this usually happens when the user requests a paste operation.

> ***Note:*** You can only paste the text to a text box with the same pasteboard type as that of the source textbox. For example, if you set the Pasteboard type as *App Level Persistent* , you can paste the text only to another text box whose pasteboard type is also set to *App Level Persistent* .

> ***Note:*** This property is specific to the iOS platform.

Close Button

Specifies whether the keypad window displays a **Done** button.

> ***Note:*** This property is specific to the iOS platform.

Close Button Text

Specifies alternate text for the "Done" button. This property is available only when the *Close Button* is enabled.

> ***Note:*** This property is specific to the iOS platform.

Clear Button

Specifies whether the keypad window displays a **Clear** button that clears text in the text box.

> ***Note:*** This property is specific to the iOS platform.

Progress Indicator

Specifies whether to display a progress indicator showing that widget content is being loaded.

> ***Note:*** This property is specific to the iOS platform.

Keyboard Action Label

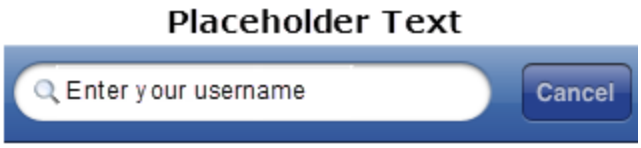
Specifies the text displayed on the action key of the keyboard.

Default: Done

> ***Note:*** This property is specific to iOS and Android platform.

Left Image

Specifies an image displayed on the left side of the text box. For example, use a magnifying glass image to indicate the text box is used as a search field:



To select an image, click the **Edit** button to open the **LeftViewImage** dialog box, and then select an available image or provide an image URL.

> ***Note:*** This property is specific to the iOS platform.

Done	Go	Search	Next
Send	Google	Join	Route
Yahoo	Call		

Auto Complete

Specifies whether characters entered in the text box are matched against a dictionary of words, and word suggestions are displayed.

Default: Off (word suggestions are not displayed)

> ***Note:*** This property is specific to the SPA platform.

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget .

Actions

Actions define what happens when an event occurs. On a TextBox widget, you can run an action when the following events occur:

- onTextChange: The action is triggered when text in the text area changes. The action is not triggered if the text changes programmatically.
- onDone: The action is triggered when the user has entered text and clicks or touches the action key.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.

- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.
- onBeginEditing: The action is triggered when the user clicks within the text box to start editing (iOS, Android, and Single Page Application (SPA)).
- onEndEditing: The action is triggered when the user the user ends the editing process by clicking on another widget or the **Done** button (iOS, Android, and Single Page Application (SPA)).

For more information, see Add Actions.

Placement Inside a Widget

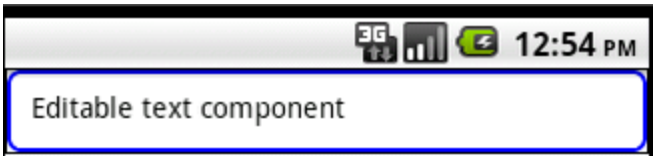
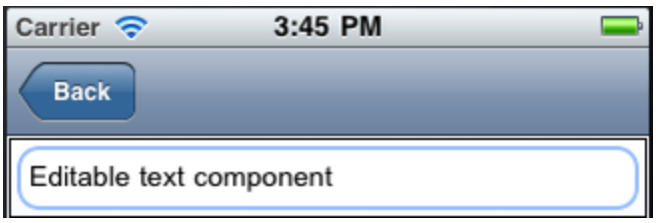
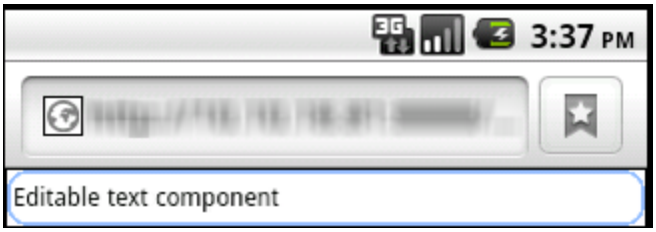
The following table summarizes where a TextBox2 widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes

Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the TextBox2 widget varies as follows:

Platform	Appearance
Android	
iOS	
SPA	

Browser

Use a Browser widget to display HTML content in your application. The HTML content can be static or obtained from a URL.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a Browser widget:

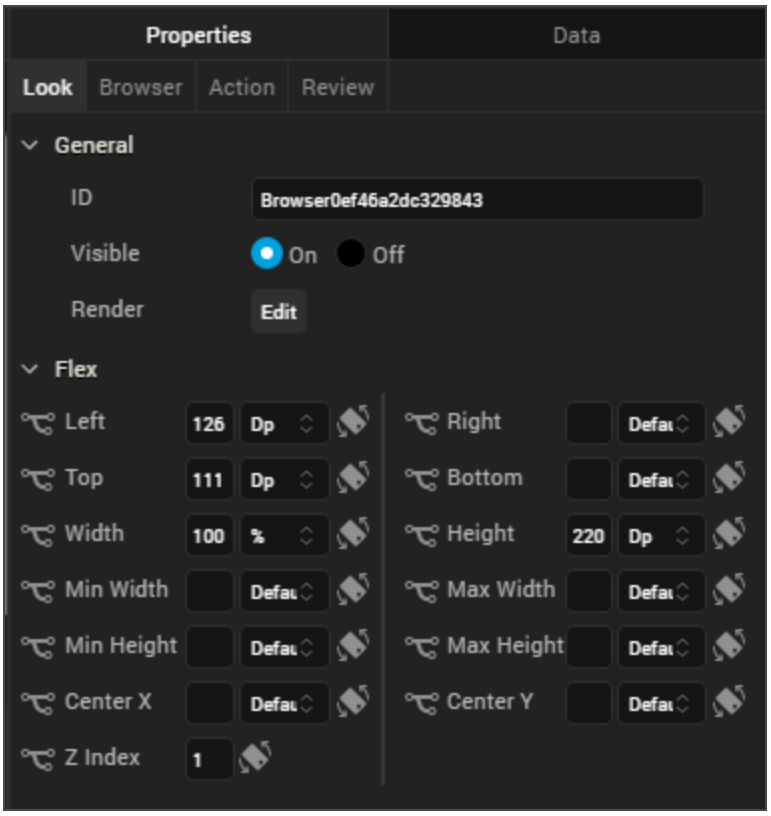
- For the iOS platform, use a Browser widget as a screen-level widget and move other widgets to the header or footer. Otherwise, dual scrolling bars may be displayed.
- The Browser widget is memory- and performance-intensive. Initial RAM usage is high, and the RAM usage grows in proportion to the size and number of rendered images and static text.
- If there are multiple instances of a Browser widget in the same application, information sharing (for example, cookies) may not behave as expected. Do not place multiple Browser widgets on a form, or more than two Browser widgets in an application.
- Do not use a Browser widget to display rich text. It should be used only to display large HTML content. Use a RichText widget to display rich text.
- Avoid using a Browser widget to create an application that looks and behaves like a web browser. Typically, users expect to use the native browser to browse web content.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.

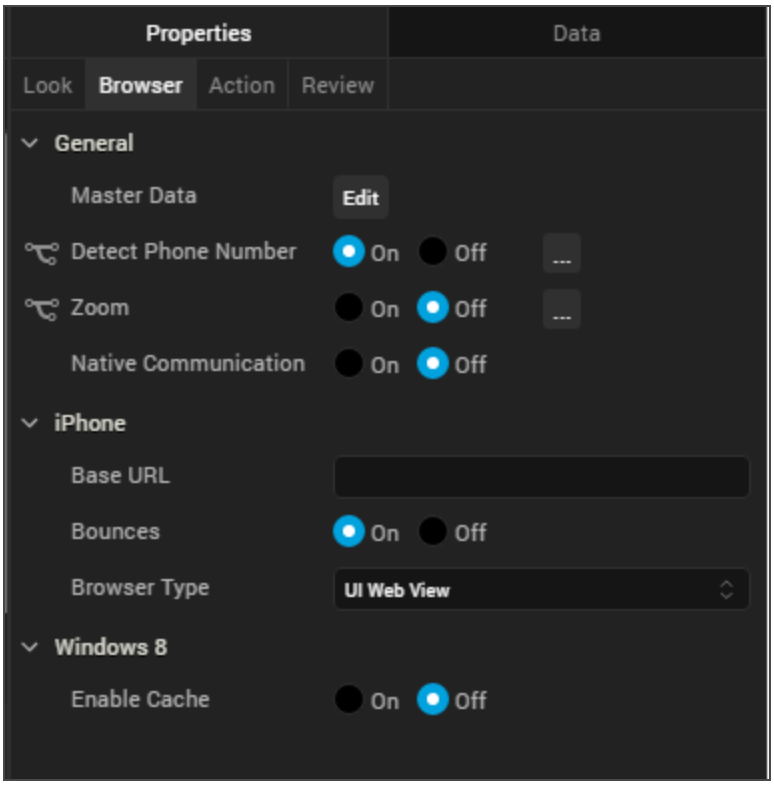
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Browser Properties

Browser properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Master Data

Specifies the content source for the Browser widget. Click **Edit** button to open the **Master Data** dialog box, and select one of the following options:

- **Content.** Displays a text field where you enter or paste the HTML content to be displayed.
- **URL.** Displays a text box where you specify the URL of an HTML page to display. The URL must begin with http:// .
- **Local File.** Displays a text box where you specify a local web-based file for launching local HTML content, such as a web app. For more information, see Add Local HTML Content.

Detect Phone Number

Specifies whether the Browser widget supports the detection of phone numbers on the web page and displays them as clickable links. When a user clicks a phone link, the Phone application launches and dials the number.

The option you choose becomes the default for all the platforms. You can change the default option or provide a platform-specific option by forking the **Detect Phone Number** property. For more information, see Fork a Widget Property.

Native Communication

Specifies whether to enable a web app JavaScript module running in the Browser widget to execute JavaScript code in the Kony native context.

Full Screen Widget

Specifies whether the Browser widget occupies the full screen. Typically, enabling the **Full Screen Widget** property is recommended. If it is disabled, a scroll bar displays either on the Browser widget or the Form.

The option you choose here becomes the default for all platforms. You can change the default option or provide a platform-specific option by forking the **Full Screen Widget** property. For more information, see Fork a Widget Property.

> ***Notes:***
>
> - This property is available only on VBox forms.
> - Do not place more than one Cordova Browser widget as a full-screen widget on a form. Also, if you make a Cordova Browser widget a full-screen widget, place only the Cordova Browser widget on the form and do not place any other widgets on the form.
> - Do not enable the **Full Screen Widget** property for more than one widget on a form. Otherwise, users may experience unexpected display and scrolling behavior.

Zoom

Specifies whether enable the Zoom feature for the Browser widget. The Zoom feature provides the ability to change the scale of the view area.

The option you choose here becomes the default for all the platforms. You can change the default option or provide a platform-specific option by forking the **Zoom** property. For more information, see Fork a Widget Property.

Base URL

For the iOS platform, specifies a URL to provide additional web-based functionality.

Enable Cache

For the Windows 8 platform, specifies whether data is cached relative to the Browser widget. For more information, refer to *kony.evaluateJavaScriptInNativeContext* in the Kony Visualizer API Developer's Guide.

Actions

Actions define what happens when an event occurs. On a Browser widget, you can run an action when the following events occur:

- onFailure: The action is triggered when the specified Master Data URL fails to load content.

  > ***Note:*** This action is executed only for the given request URL, but not for the subsequent web navigation request failures.

- onSuccess: The action is triggered when the specified Master Data URL successfully loads content.

  > ***Note:*** This action is called only for the given request URL, but not for the subsequent web navigation requests.

- onPageStarted: The action is triggered when the specified page starts loading.
- onPageFinished: The action is triggered when the specified page has finished loading.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.

- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.
- onReceive: The action is triggered when the specified page has loaded and has an event callback such as digest authentication (Android).

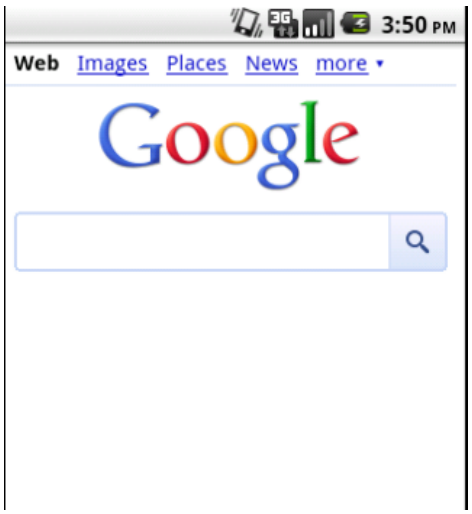
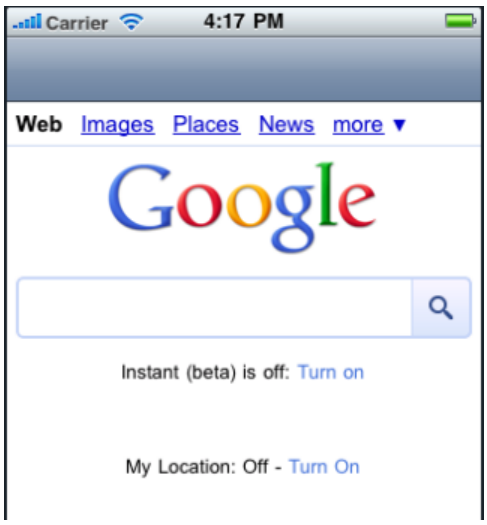
Placement Inside a Widget

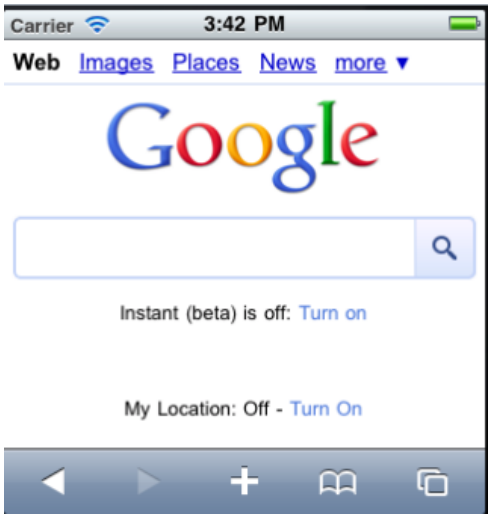
The following table summarizes where a Calendar widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation -Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Browser widget varies as follows:

Platform	Appearance
Android	
iOS	

Platform	Appearance
SPA	

Cordova Browser

Use a Cordova Browser widget to display Cordova-based HTML content in your application. The HTML content can be static or obtained from a URL.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

The following are important considerations for a Cordova Browser widget:

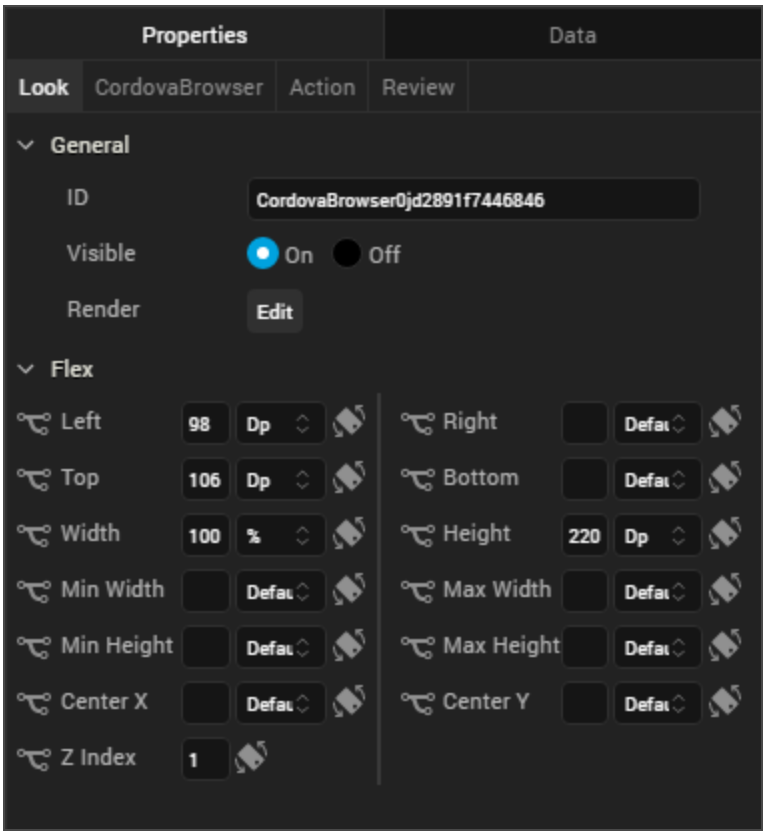
- For the iOS platform, use a Cordova Browser widget as a screen-level widget and move other widgets to the header or footer. Otherwise, dual scrolling bars may be displayed.

- The Cordova Browser widget is memory- and performance-intensive. Initial RAM usage is high, and the RAM usage grows in proportion to the size and number of rendered images and static text.
- If there are multiple instances of a Cordova Browser widget in the same application, information sharing (for example, cookies) may not behave as expected. Do not place multiple Cordova Browser widgets on a form, or more than two Cordova Browser widgets in an application.
- Do not use a Cordova Browser widget to display rich text. It should be used only to display large HTML content. Use a RichText widget to display rich text.
- Avoid using a Cordova Browser widget to create an application that looks and behaves like a web browser. Typically, users expect to use the native browser to browse web content.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

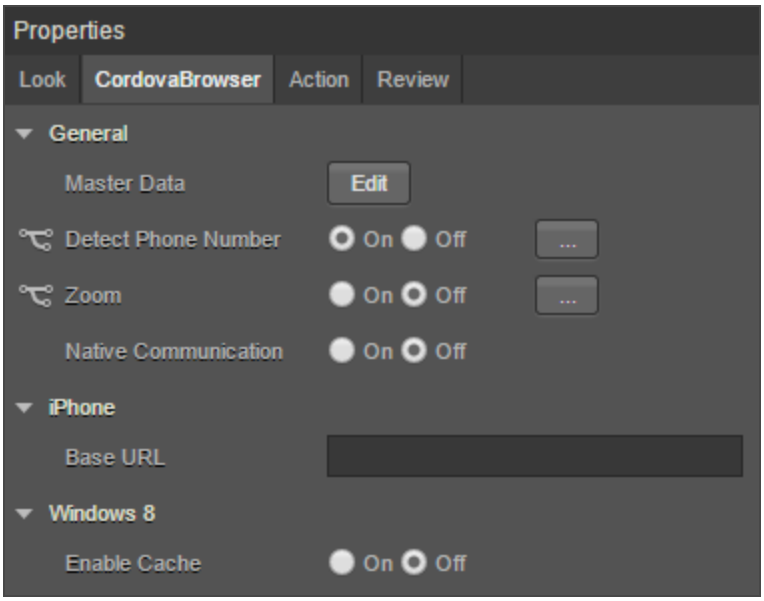
- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Cordova Browser Properties

Cordova Browser properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Master Data

Specifies the content source for the Cordova Browser widget. Click **Edit** button to open the **Master Data** dialog box, and select one of the following options:

- **Content.** Displays a text field where you enter or paste the HTML content to be displayed.
- **URL.** Displays a text box where you specify the URL of an HTML page to display. The URL must begin with http:// .
- **Local File.** Displays a text box where you specify a local web-based file for launching local HTML content, such as a web app. For more information, see Add Local HTML Content.

Detect Phone Number

Specifies whether the Cordova Browser widget supports the detection of phone numbers on the web page and displays them as clickable links. When a user clicks a phone link, the Phone application launches and dials the number.

The option you choose becomes the default for all the platforms. You can change the default option or provide a platform-specific option by forking the **Detect Phone Number** property. For more information, see Fork a Widget Property.

Native Communication

Specifies whether to enable a web app JavaScript module running in the Cordova Browser widget to execute JavaScript code in the Kony native context.

Zoom

Specifies whether enable the Zoom feature for the Cordova Browser widget. The Zoom feature provides the ability to change the scale of the view area.

The option you choose here becomes the default for all the platforms. You can change the default option or provide a platform-specific option by forking the **Zoom** property. For more information, see Fork a Widget Property.

Base URL

For the iOS platform, specifies a URL to provide additional web-based functionality.

Enable Cache

For the Windows 8 platform, specifies whether data is cached relative to the Cordova Browser widget. For more information, refer to *kony.evaluateJavaScriptInNativeContext* in the Kony Visualizer API Developer's Guide.

Actions

Actions define what happens when an event occurs. On a Cordova Browser widget, you can run an action when the following events occur:

- onFailure: The action is triggered when the specified Master Data URL fails to load content.

  > ***Note:*** This action is executed only for the given request URL, but not for the subsequent web navigation request failures.

- onSuccess: The action is triggered when the specified Master Data URL successfully loads content.

> ***Note:*** This action is called only for the given request URL, but not for the subsequent web navigation requests.

- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

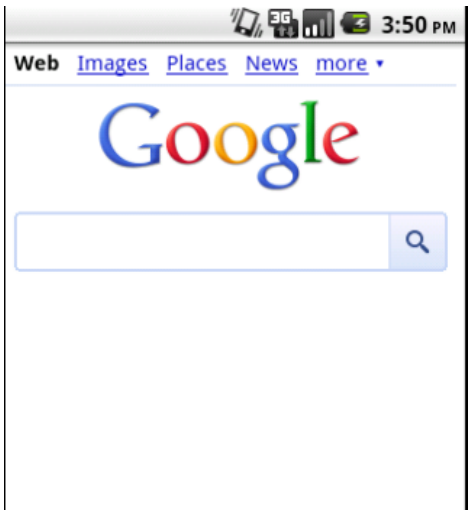
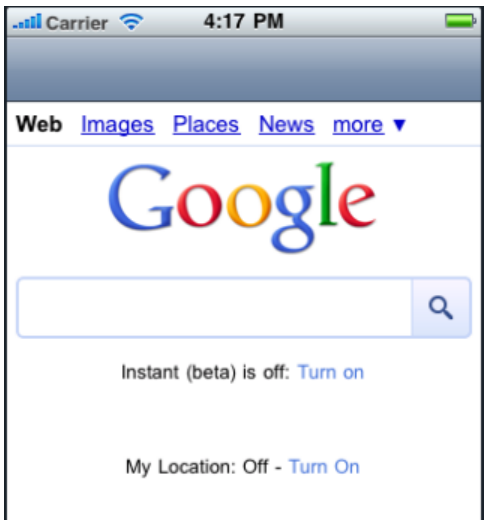
Placement Inside a Widget

The following table summarizes where a Calendar widget can be placed:

Flex Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
ScrollBox	Horizontal Orientation -Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Cordova Browser widget varies as follows:

Platform	Appearance
Android	
iOS	

Camera

Use a Camera widget to display a button that opens the device's native camera application to capture an image. By default, a saved image is stored as a PNG (Portable Network Graphics) image with the original size.

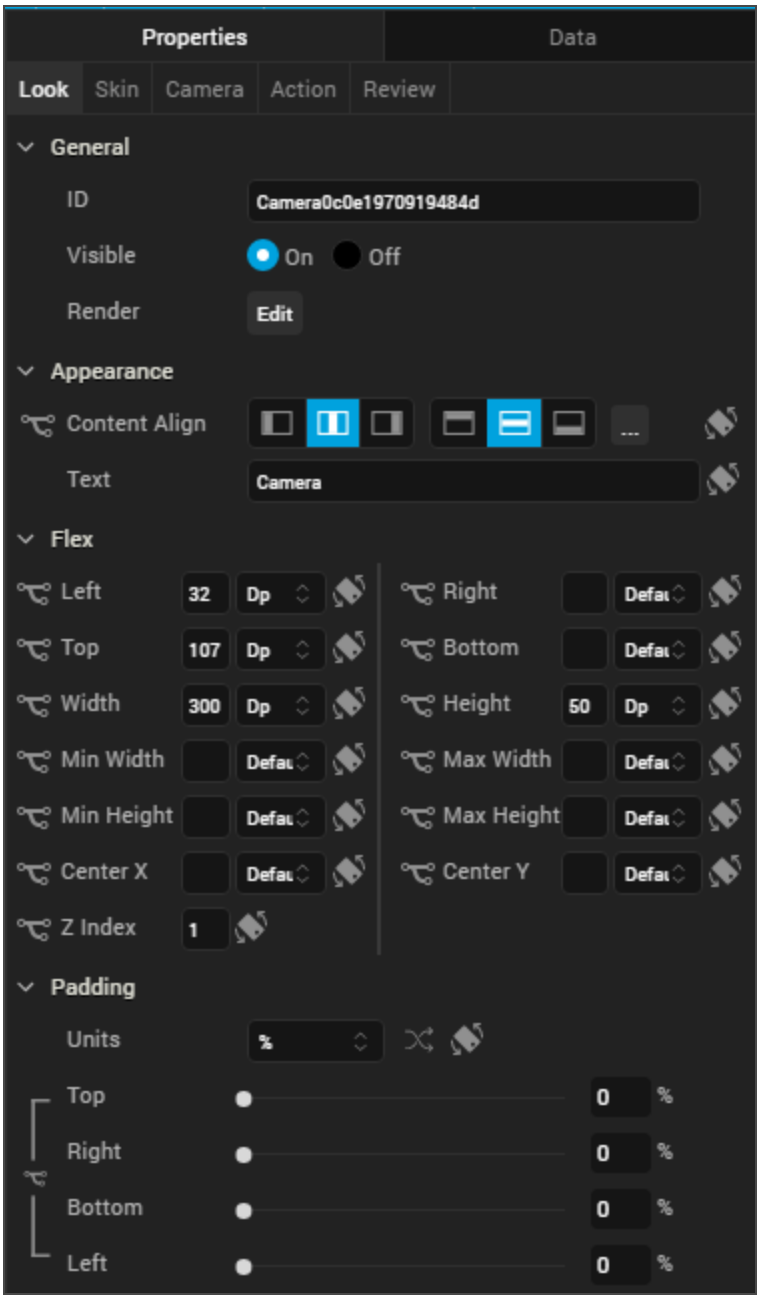
To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

> ***Note:*** The Camera widget is not supported on SPA platforms.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the Camera widget, you can apply a skin and its associated properties for the following states:

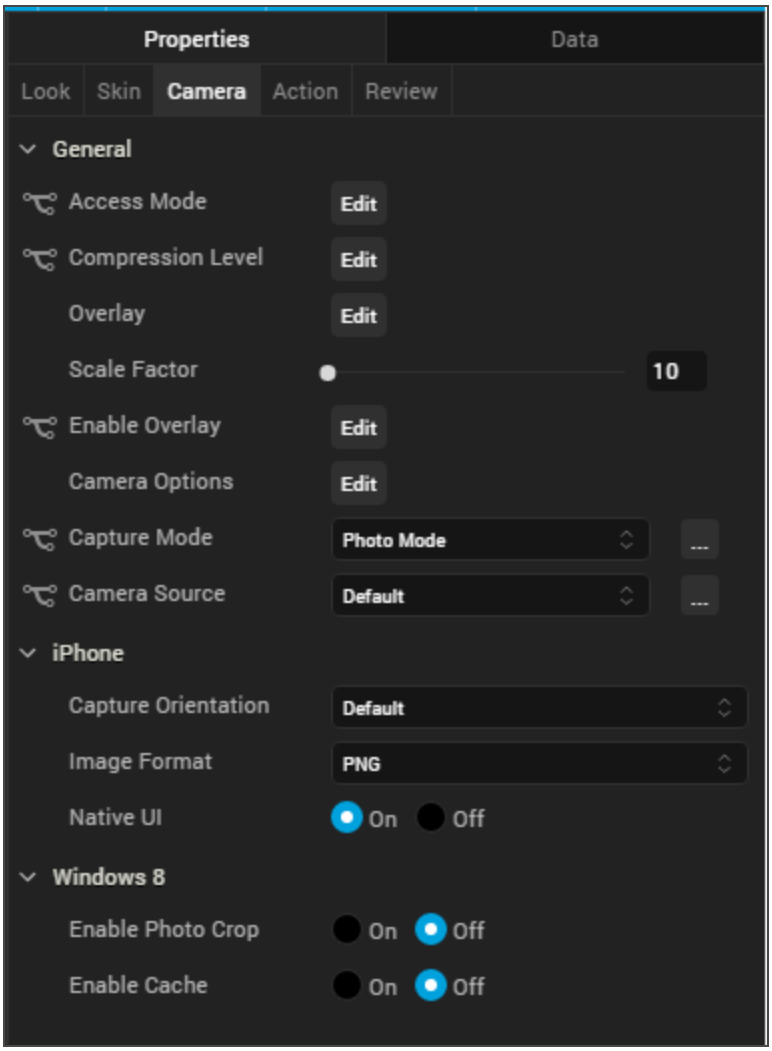
Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the widget has the focus.
Hover Skin	The look and feel of the widget when the cursor hovers over it. ***Note:*** Hover skins are available only on the Windows (native) Tablet platform.

For more information about applying skins, see Understanding Skins and Themes.

Camera Properties

Camera properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

Note: In this section, the properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Access Mode

Specifies how the captured image is stored.

Following are the options:

- Public: The captured image is stored on the device and is accessible to all the applications on the device. For example, the captured images are accessible in ImageGallery.
- Private: This is the default option for Windows. The captured image is stored on the device but is not accessible to any other application on the device and remains private to the application.

- In-Memory: The captured camera image is stored in memory and is never written to disk. Captured images are lost when the application closes.

To specify the access mode, click the **Edit** button to open the **Access Mode** dialog box. To change the default access mode, select Default and then select an access mode from the value list. To specify a platform-specific access mode, select the platform and then select an access mode from the list.

Compression Level

For JPEG images, specifies the compression level with which a captured image is stored. The compression level determines picture quality. You can specify a compression level value between 0 (best picture quality) and 100 (low picture quality).

Default: 0

To specify the compression level, click **Edit** to open the **Compression Level** dialog box. To change the default compression level, select Default and then select a compression level from the value list. To specify a platform-specific compression level, select the platform and then select a compression level from the list.

Overlay - Photo Mode

Applicable when the capture mode is photo mode.

Specifies the overlay configuration parameters for overlaying a form.

The following are the configurable properties available for various platforms:

iOS

- Overlay Form : Specifies the reference of the form to be rendered over the camera view. When this option is set, the Capture Orientation property is not respected.

  Default : None

- Cropping Reference Image : Specifies the reference of the Image widget in the Overlay Form which guides the camera to crop the captured image to the Reference Image Dimensions.

  Default : None

Android

- Overlay Form : Specifies the reference of the form to be rendered over the camera view. When this option is set, the Capture Orientation property is not respected.

  Default : None

- Cropping Reference Image : Specifies the reference of the Image widget in the Overlay Form which guides the camera to crop the captured image to the Reference Image Dimensions.

  Default : None

- Capture Button Skin : Specifies the skin for a captured button.
- Capture Button Text : Specifies the text for a captured button.
- Tap Anywhere: Specifies to capture an image with a tap on the camera overlay view.

  Default : false

Windows 8

- Overlay Form : Specifies the reference of the form to be rendered over the camera view. When this option is set, the Capture Orientation property is not respected.

  Default : None

- Cropping Reference Image : Specifies the reference of the Image widget in the Overlay Form which guides the camera to crop the captured image to the Reference Image Dimensions.

  Default : None

- Tap Anywhere: Specifies to capture an image with a tap on the camera overlay view.

  Default : false

Overlay - Video Mode

Applicable when the capture mode is video mode.

Specifies the overlay configuration parameters for overlaying a form.

The following are the configurable properties available for various platforms:

iOS

- Overlay Form : Specifies the reference of the form to be rendered over the camera view.

  Default : None

- Start Button Text: Specifies the text on the start button for the camera. You can enter text here.
- Stop Button Text: Specifies the text on the stop button for the camera. You can enter text here.
- Start Button Skin: Specifies the skin on the start button for the camera. Default is none. You can select a skin from the drop-down list.
- Stop Button Skin: Specifies the skin on the stop button for the camera. Default is none. You can select a skin from the drop-down list.
- Timer Control Skin: Specifies the skin on time controller for the camera. Default is none. You can select a skin from the drop-down list.

Android

- Overlay Form : Specifies the reference of the form to be rendered over the camera view. When this option is set, the Capture Orientation property is not respected.

  Default : None

- Start Button Text: Specifies the text on the start button for the camera. You can enter text here.
- Stop Button Text: Specifies the text on the stop button for the camera. You can enter text here.
- Start Button Skin: Specifies the skin on the start button for the camera. Default is none. You can select a skin from the drop-down list.
- Stop Button Skin: Specifies the skin on the stop button for the camera. Default is none. You can select a skin from the drop-down list.
- Timer Control Skin: Specifies the skin on time controller for the camera. Default is none. You can select a skin from the drop-down list.

Windows 8

- Overlay Form : Specifies the reference of the form to be rendered over the camera view. When this option is set, the Capture Orientation property is not respected.

  Default : None

- Cropping Reference Image : Specifies the reference of the Image widget in the Overlay Form which guides the camera to crop the captured image to the Reference Image Dimensions.

  Default : None

- Tap Anywhere: Specifies to capture an image with a tap on the camera overlay view.

  Default : false

Scale Factor

Specifies the ratio by which a captured image is scaled down. You can specify a scale factor between 10 percent of the captured image and 100 percent (no reduction).

Enable Overlay

Specifies whether to enable overlay of a form interface over the camera view.

Default: Disabled.

To specify the **Enable Overlay** property, click **Edit** to display the **Enable Overlay** dialog box. To change the default Enable Overlay setting, select Default and then select a value from the list. To specify a platform-specific Enable Overlay setting, select the platform and then select a value from the list.

Capture Mode

Specifies the capture mode of the camera. You can select Photo Mode or Video Mode.

Camera Source

Specifies the camera source, either Default, Rear, or Front.

Video Duration

When Capture Mode is set to Video Mode, specifies the duration of the video in seconds.

Capture Orientation

For the iOS platform, specifies the orientation of the captured image, either Default, Landscape, or Portrait.

Default: Default

Image Format

For the iOS platform, specifies whether the image is saved as a PNG (Portable Network Graphics) or a JPEG (Joint Photographic Experts Group) image.

Default: PNG

Native UI

For the iOS platform, specifies whether the camera uses the native interface with the default platform controls for the camera, or a user interface with custom options.

Default: On (the camera uses the native interface)

Video Format

For the iOS platform when Capture Mode is set to Video Mode, specifies whether the video is saved in Mp4 or MOV format.

Default: MP4

Video Stabilization

For the Android platform, specifies whether video stabilization is enabled.

Default: Off (video stabilization is disabled)

Enable Photo Crop

For the Windows platform, specifies whether the captured image can be cropped.

Default: Off (photo crop is disabled)

Tool Tip

For the Windows Tablet platform, specifies a message that displays when you hover the mouse pointer over the widget.

Actions

Actions define what happens when an event occurs. On a Camera widget, you can run an action when the following events occur:

- onCapture: The action is triggered when the user captures a picture.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.
- onFailure: The action is triggered when an error occurs using a Camera widget. For example, you set a camera source but it is not available on the device (iOS and Android).

For more information, see Add Actions.

Placement inside a Widget

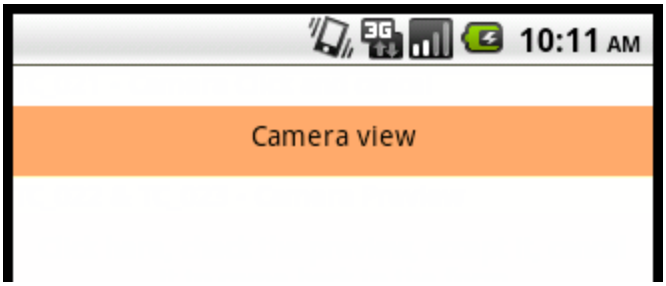
The following table summarizes where a Camera widget can be placed:

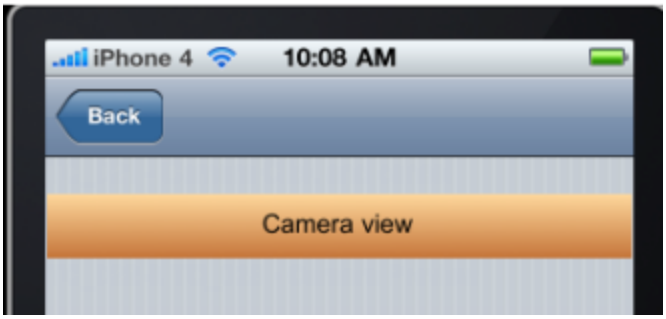
Form	Yes
HBox	Yes
VBox	Yes

ScrollBox	Horizontal Orientation -Yes Vertical Orientation- Yes
Tab	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Camera widget varies as follows:

Platform	Appearance
Android	

Platform	Appearance
iOS	

Map

Use a Map widget to display locations on a map. Platforms such as iPhone (above 3.0) and Android provide a native map widget, that can be displayed as part of an application. The following table shows the available mapping services:

Platform	Mapping Service
Android	Google Maps
iPhone	Google Maps
Mobile Web (advanced)	Google Static Maps, Native maps of the device, and interactive maps (Java script)

On platforms where a native map widget is not available, the Map widget integrates with Google Maps to display a static image with zoom and pan controls. You can customize the map view.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Note:

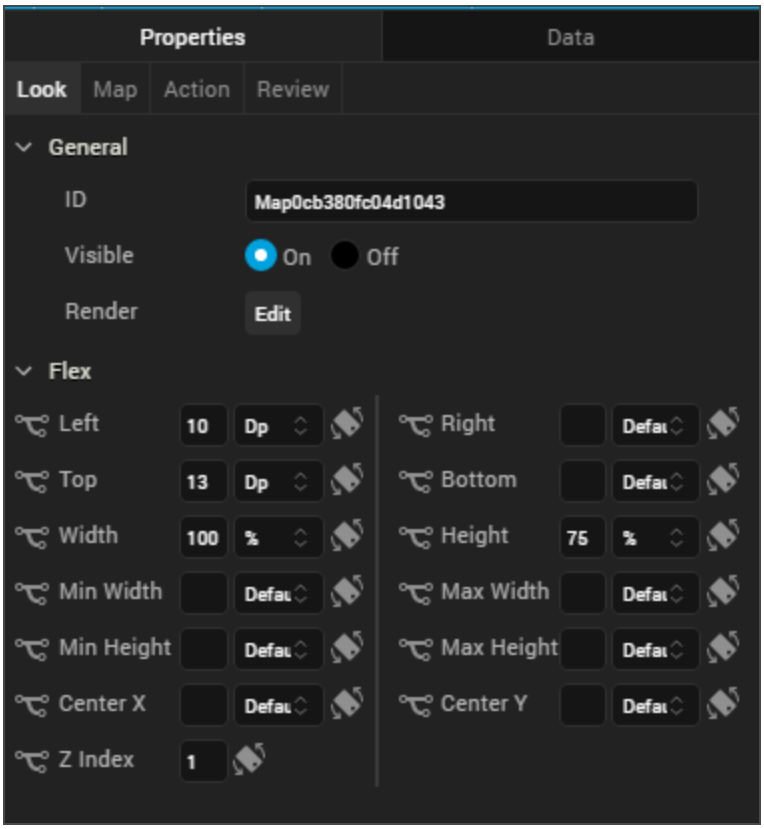
- Mobile Web (basic) and Non-Touch HTML devices support only static maps.

- On the Android platform, the Map widget is not available in a popup window.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.

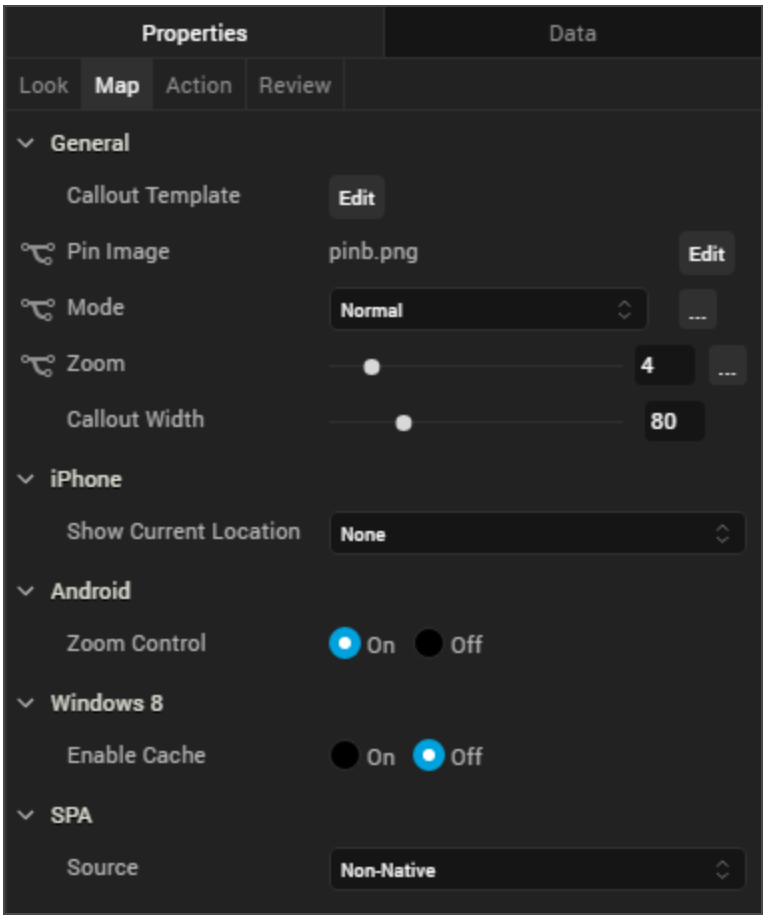


For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Map Properties

Map properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

> ***Note:*** In this section, the properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Callout Template

Specifies a template for a map callout (for example, a pin) that indicates a particular location on a map. The template can include Label, Link, RichText, Button, and Image widgets.

> ***Note:*** If a template is not specified, the platform-specific default callout is used.

For information about creating a callout template, see Templates:Maps.

To specify a template, click **Edit** to open the **Callout Templates** dialog box, and then choose a template from the list of available templates.

Pin Image

Specifies the pin image to use to indicate a location on map. A default image is provide by the system.

To specify a pin image, click the **Edit** button to open the **Pin Image** dialog box. To change the default pin image, select Default and then select a pin image from the value list. To specify a platform-specific pin image, select the platform and then select a pin image from the list.

Mode

Specifies the map viewing mode. The mode can be one of the following:

- Normal: A traditional map view of roads, parks, borders, etc.
- Satellite: A map showing aerial imagery.
- Street: A map with street-level imagery.
- Hybrid: A street map superimposed on satellite map.
- Terrain: A map showing the surface of the land in 3D view.
- Polygon: A map showing the polygonal area specified in the locationdata property.
- Traffic: A map showing streets with different colors to indicate traffic congestion. Green indicates low traffic, orange indicates medium traffic and red indicates heavy traffic.

Select a value from the **Mode** list to specify a default mode that is applied to all platforms. To provide a platform-specific mode, you can fork the **Mode** property. See Fork a Widget Property for more details.

Source

Specifies the map source. The map source can be one of the following:

- Native: The application uses mapKey and provider properties to fetch the map. The fetched map is interactive with the ability to zoom and pan.

> ***Note:*** Polygon view on the advanced Mobile Web platform is available only when the source is set to non-native.

- Non Native: The application uses the map that is on the device.

> ***Note:*** SPA platforms support only Google Static Maps as a source. Static maps are directly requested from Google for a given latitude and longitude. Kony Visualizerdoes not support any other option because the size of the get request URL could be larger than 256 characters, causing the request to fail.

Select a value from the **Map Source** list to specify a default map source that is applied for all the platforms. To provide a platform-specific map source, you can fork the **Map Source** property. See Fork a Widget Property for more details.

Zoom Level

Specifies the zoom level for the current map view. The range varies from platform to platform.

Select a value from the zoom level slider to specify a default zoom level that is applied for all the platforms. To provide a platform-specific zoom level, you can fork the **Zoom Level** property. See Fork a Widget Property for more details.

Full Screen Widget

Specifies whether the Map widget occupies the full screen.

> ***Note:*** This property is available only on VBox forms.

Callout Width

Specifies the width of the callout on the map. You can specify a value between 1 and 100 that represents a percentage relative to the width of the Map widget. For example, a callout width of 100 uses the full width of the widget. If the specified value is less than 1 or more than 100, the callout width is set to 80 percent.

Height

Specifies the height of the map as a percentage relative to the value of the Height Reference property.

> ***Note:*** This property is not available on Flex Forms.

Height Reference

Specifies how the map height is calculated:

- Form Reference: The map height is calculated as a percentage of the form height, excluding headers and footers. This option does not apply to a Map widget on a popup or template.
- Parent Width: If the Map widget is on a popup or template, the width is calculated based on the width of the parent container.

> ***Note:*** This property is not available on Flex Forms.

Show Current Location

Specifies whether and how the current location on the map is indicated:

- None: The current location is not indicated.
- Pin: The current location is indicated by a pin.
- Circle: The current location is indicated by a circle.

> ***Note:*** This property is specific to the iOS platform.

Zoom Control

Specifies whether to display the zoom control on the map.

> ***Note:*** This property is specific to the Android platform.

Actions

Actions define what happens when an event occurs. On a Camera widget, you can run an action when the following events occur:

- onClick: The action is invoked by the platform when the map is clicked.
- onPinClick: The action is triggered when a user clicks the map pin, passing the selected *locationdata* to the callback.
- onSelection: The action is triggered when a user clicks on a map callout.
- onMapLoaded: The action is invoked by the platform when map rendering is complete.
- onBoundsChanged: The action is invoked by the platform when the content of the map changes

For more information, see Add Actions.

Placement inside a Widget

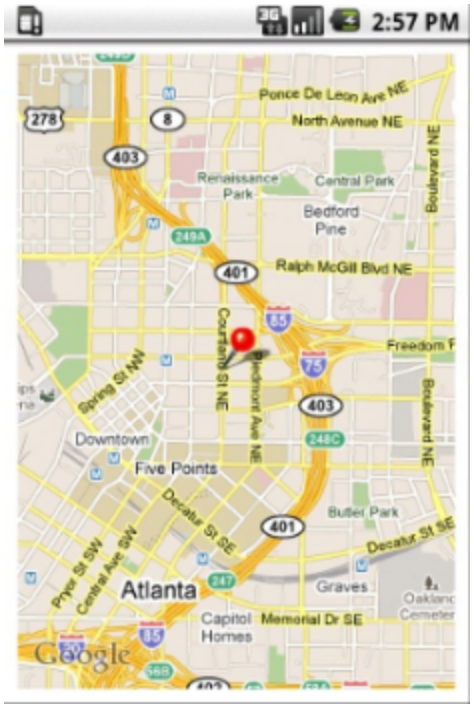
The following table summarizes where a Map widget can be placed:

Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation -Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes

Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Map widget varies as follows:

Platform	Appearance
Android	

Platform	Appearance
iOS	
SPA	

Phone

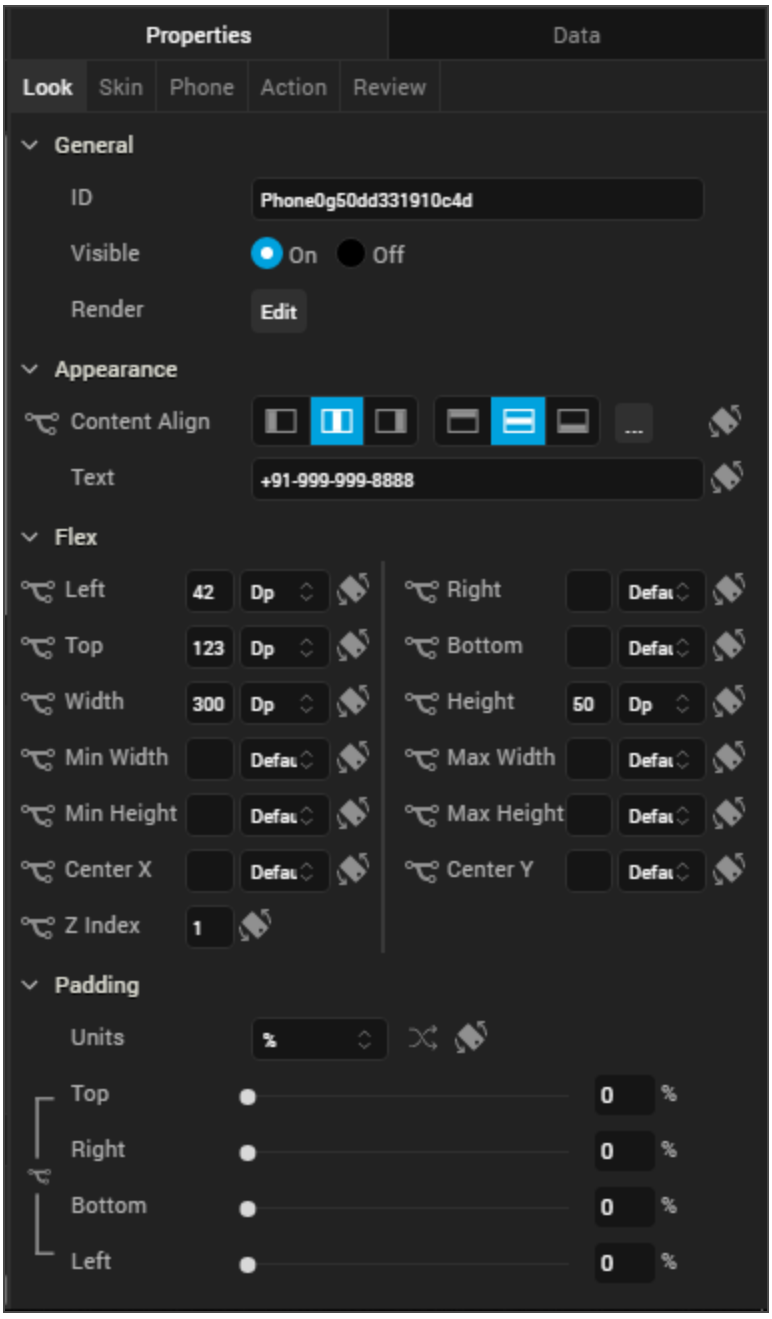
Use a Phone widget to launch the native phone dialer and initiate a phone call to a specified phone number. When a user clicks the Phone widget, the native dialer launches to make a call.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the Phone widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.
Focus	The skin applied when the widget has the focus.

For more information about applying skins, see Understanding Skins and Themes.

Actions

Actions define what happens when an event occurs. On a Phone widget, you can run an action when the following events occur:

- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

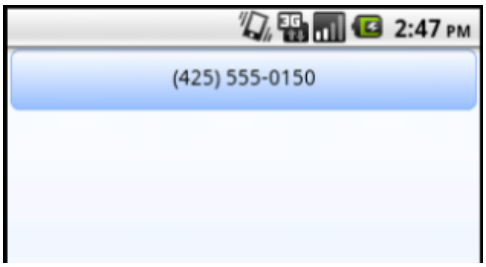
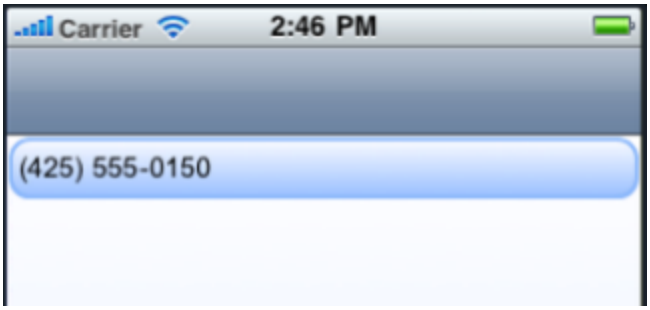
The following table summarizes where a Phone widget can be placed:

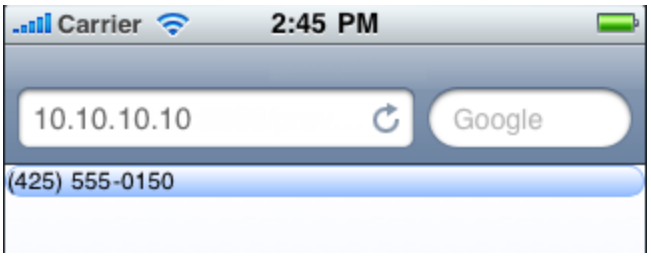
Form	Yes
HBox	Yes
VBox	Yes

ScrollBox	Horizontal Orientation -Yes Vertical Orientation- Yes
Tab	Yes
Segment	Yes
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the Phone widget varies as follows:

Platform	Appearance
Android	
iOS	

Platform	Appearance
SPA	

PickerView

Use a PickerView widget to enable a user to select a single combination of values from multiple sets of values. A user rotates through each set of values with a selection indicator. For example, a user can select a single date from lists of months, days of the month, and years.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

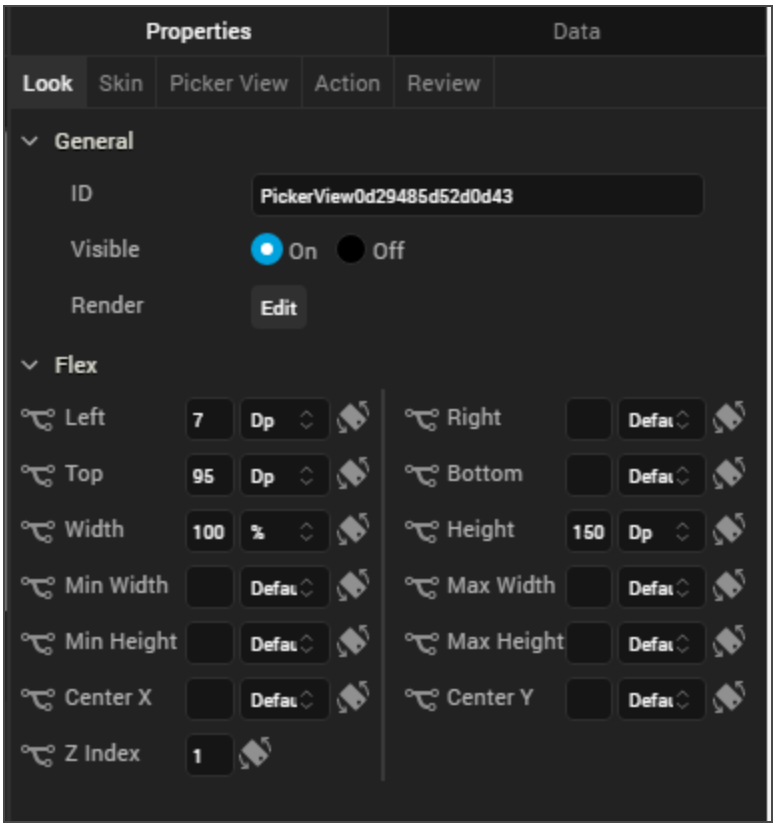
> ***Note:*** The PickerView widget is not available on SPA platforms.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.

- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the PickerView widget, you can apply a skin and its associated properties for the following states:

Skin	Definition
Normal	The default skin of the widget.

Skin	Definition
Focus	The skin applied when the widget has the focus.

For more information about applying skins, see Understanding Skins and Themes.

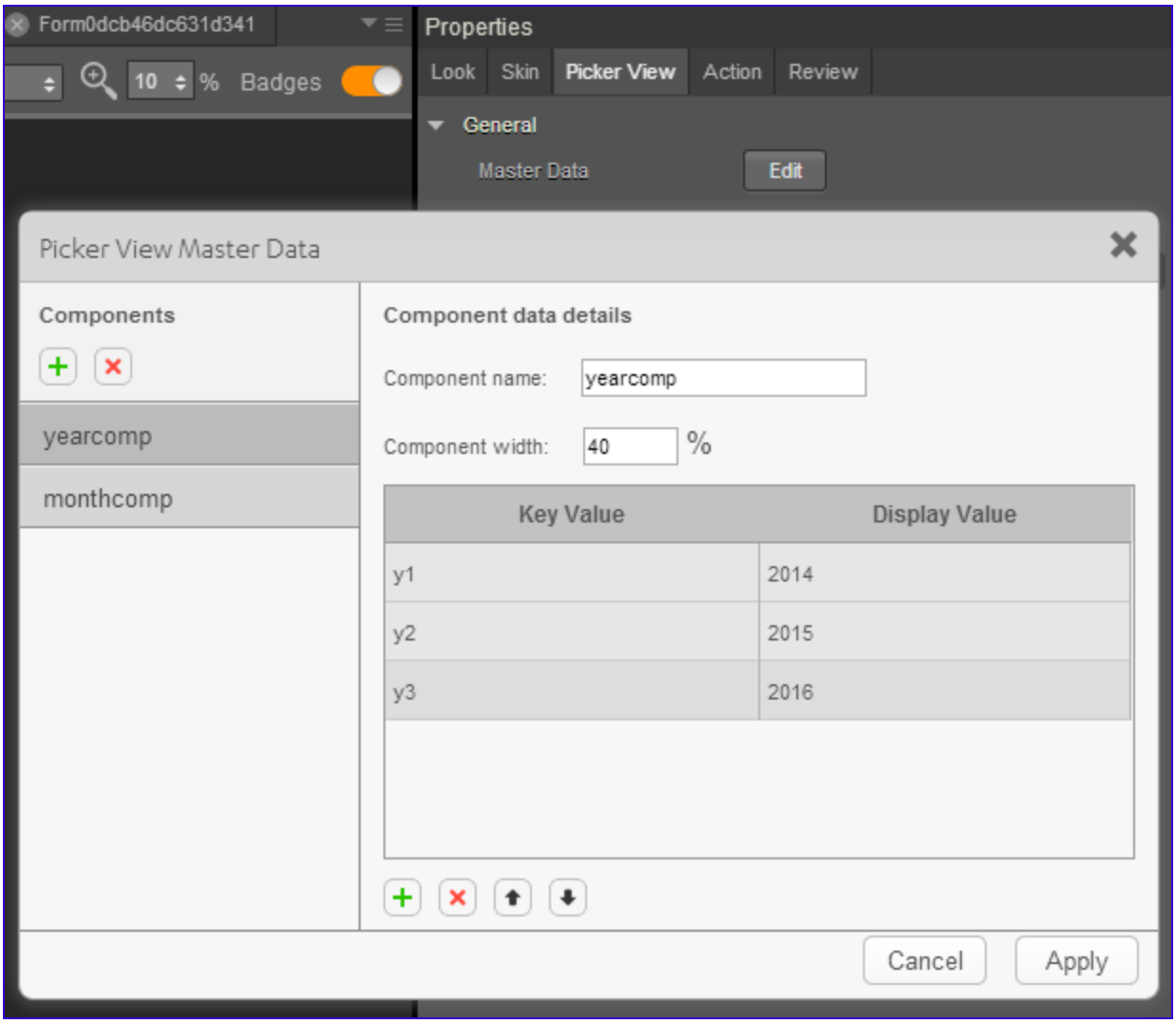
PickerView Properties

PickerView properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

Master Data

Specifies the values displayed in the PickerView widget.

To specify values, click the **Edit** button to open the **PickerView Master Data** dialog box.



Specify key and display values for each component in the PickerView. Each component represents a set of data; for example, years and months. You can add a component or a key/value pair by clicking the appropriate + button.

> ***Note:*** The combined component width should total 100 percent.

View Type

For the Android platform, specifies the picker view type.

Default: Flat

The following are the options:

- Flat: The picker appearance is flat.
- Wheel: The picker displays as a wheel.

Enable Cache

For the Windows 8 platform, specifies whether data is cached relative to the PickerView widget.

Default: Off

Actions

Actions define what happens when an event occurs. On a PickerView widget, you can run an action when the following event occurs:

- onSelection: The action is triggered when the component selection changes.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

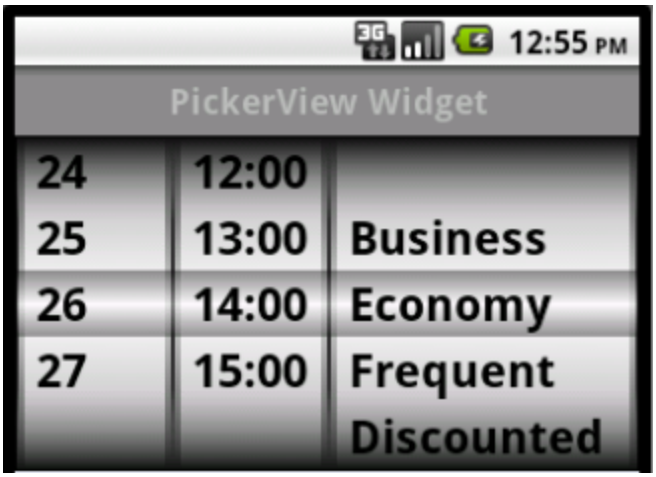
The following table summarizes where a PickerView widget can be placed:

Form	Yes
HBox	Yes
VBox	Yes

Scroll Box	Horizontal Orientation - Yes Vertical Orientation- Yes
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Widget Appearance on Platforms

The appearance of the PickerView widget varies as follows

Platform	Appearance
Android	

Platform	Appearance
iOS	

Segment

A Segment widget consists of multiple segments (rows or records) and each segment can have multiple child widgets. You can use Segment widgets to create menus and grouped lists in your applications.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

You can place a Segment widget within a number of parent widgets, and you can place a number of child widgets within a Segment widget.

You can add a Segment widget to the following widgets:

Flex Form	VBox Form
FlexContainer	FlexScrollContainer

ScrollBox (vertical orientation only)	Tab
Popup	VBox widget (but not an HBox)

You can place the following widgets within a Segment widget:

Button	Calendar	FlexContainer
Image2	Label	Phone
RichText	Slider	Switch
TextArea2	TextBox2	ListBox

Important Considerations

The following are important considerations for a Segment widget:

- Segment occupies memory from two perspectives:
  - The amount of data required by the number of rows. For example, if you set data for 100 rows, memory for all 100 records will be in memory.
  - The view hierarchy (Box and other supported widgets) in each segment row. If the View hierarchy is complex, the memory usage is high.

    > ***Note:*** iOS, Android, and Windows platforms, if your segment has large data sets (more than 20 records with each record having more than 15 widgets), set the segment as a Screen Level Widget.

- You cannot add any elements to the widgets dynamically, but you can hide any elements if you do not provide any data for that element.
- You can dynamically change the skin of the widgets in the segment.
- A Segment widget can be placed in a ScrollBox widget only if the ScrollBox widget has a vertical orientation.

- The height of the Segment widget is determined by the content of the widget. If you set the Screen Level Widget to **True**, then the height of the Segment widget is the form height excluding headers and footers.

Using the Segment Widget in a Component or Master

One you add a Segment widget to a component or master, you cannot drop other widgets onto that segment widget. To add a widget to a Segment widget in a component or master, create a segment row template, populate it with the widgets you want it to have, and then you configure the segment widget in the component or master to reference that row template. Once the segment widget in your component or master is populated width content from the row template, you can edit the segment widget's master data to have the per-row customized data that you need.

> ***Note:*** A segment template can only apply to one channel. To work around this, create the segment template for one channel, copy it, and then paste it into the other channels.

To use a Segment widget in a component or master:

1. Create a segment row template. To do so, in the Project Explorer, click the **Templates** tab, open the channel you want to create the segment template for (e.g. Mobile), click the context menu arrow of **Segments**, and then click **New Template**. The template is created and opens on the Visualizer Canvas. It might be a good idea to rename the template to something descriptive, such as rowMobile. To do so, on the **Templates** tab, click the context menu arrow of the newly-created template, and then click **Rename**.
2. Drag a FlexContainer widget onto the new template on the Visualizer Canvas, and configure it how you want.
3. Drag onto the FlexContainer the widgets that you want the segment row template to have, and configure them how you want.
4. Once you have the segment row template configured the way you want it, if you want to copy it to other channels, click the context menu arrow of the configured template, and then click **Copy**. Navigate to and open the channel you want to paste it to, click the context menu arrow of

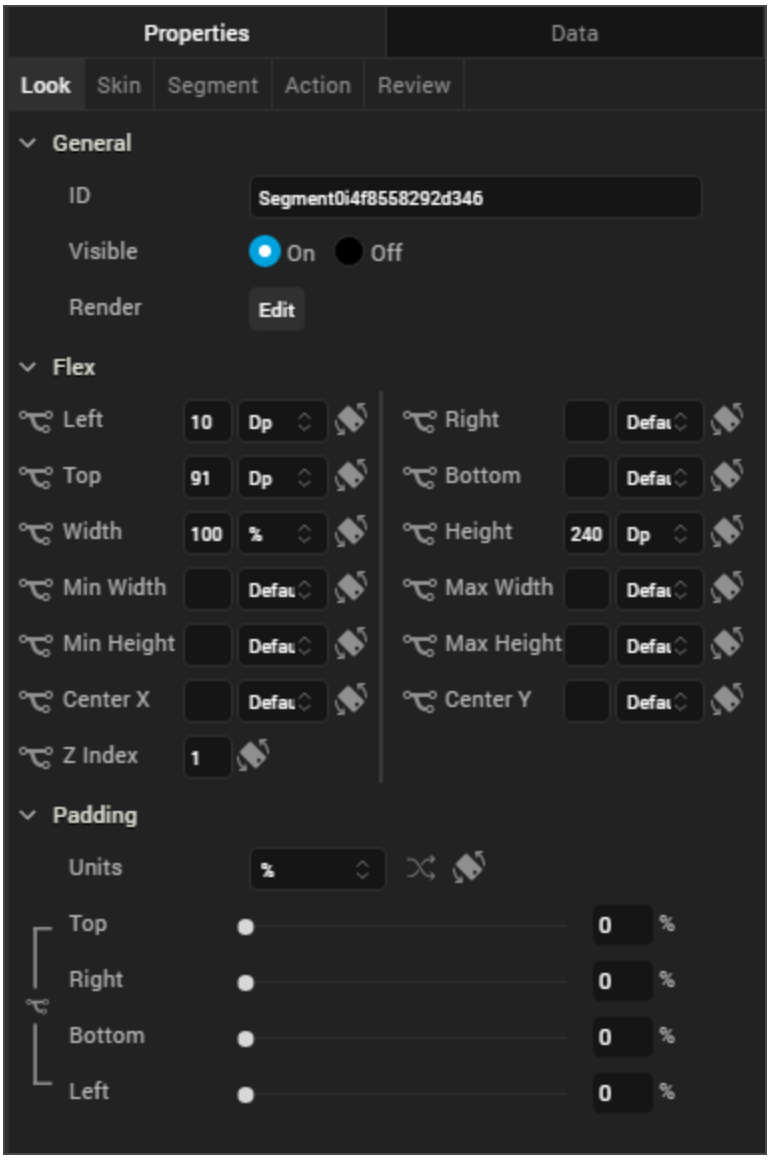
Segments, and then click **Paste**. Rename the template if you'd like, and then make any needed modifications so that it looks and behaves correctly in that channel.

5. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **Save All**.
6. Add a Segment widget to the component or master. To do so, in the Project Explorer, on the **Templates** tab, in the Components or Masters section, select the component or master so that it opens on the Visualizer Canvas, add (or select an existing) container widget in the component or master, and then drag a Segment widget onto it.
7. In the Properties Editor for the Segment widget you just added, click the **Segment** tab. Then from the Row Template drop-down list, select the row template you want to use.
8. From the **Segment** tab, click the **Edit** button for Master Data, and make any changes needed to the segment data.
9. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **Save All**.

Look Properties

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin Properties

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the Segment widget, you can apply a skin and its associated properties for the following states:

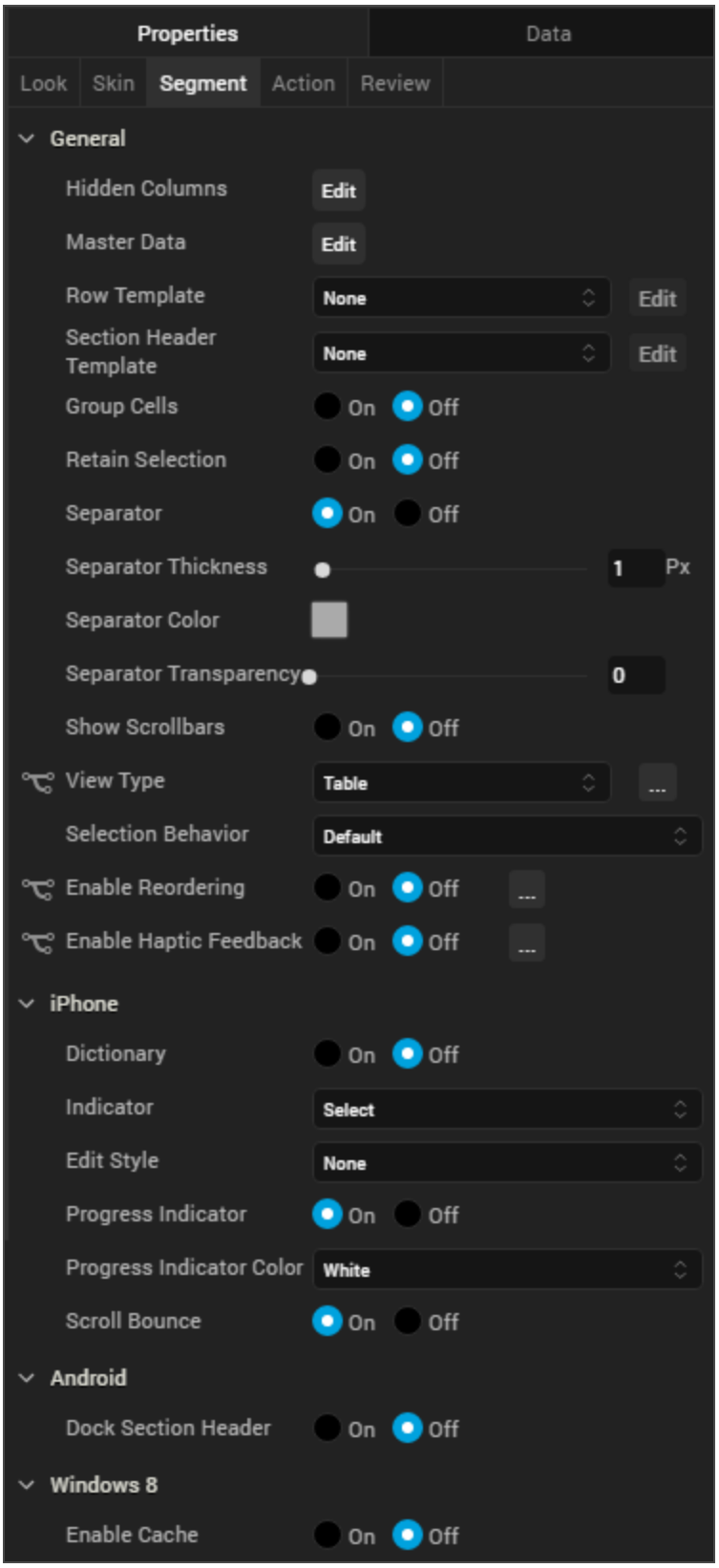
Skin	Definition
Row	The skin that is applied for each row.
Row - Focus	The skin that is applied when user focuses on a row.
Section Header	The skin that is applied to the Section Header of Segment widget.
Widget	The skin that is applied to the entire Segment.
Row - Alternate	The skin that is applied to every alternate *even numbered* row in the segment.
Blocked UI	The skin that is to block the interface until the action in progress (for example, a service call) is completed. ***Note:*** Blocked UI is available only for SPA platforms.
Pressed	The skin to indicate that the row of the segment is pressed or clicked.
Refresh - Pull	The look and feel of a widget when the scroll bar is pulled.
Refresh - Push	The look and feel of a widget when the scroll bar is pushed.

For more information about applying skins, see Understanding Skins and Themes.

Segment Properties

Segment properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.

> ***Note:*** Properties that can be forked are identified by an icon located to the left of the property. For more information, see Fork a Widget Property.



Master Data

The master data is enabled only when the widgets are added to the **Segment** widget.

The following example illustrates using the Master Data property to add an image and button to the Segment widget:

1. Select the Segment widget.
2. Drag an Image widget and Button widget onto the Segment widget.
3. From the **Segment** widget properties, click the **Segment** tab.
4. Click the **Edit** button of the **Master Data** option to open the **Master Data** dialog box.
5. Provide the image and button widget details as shown in the following image.
6. Click **OK**.

Row Template

Indicates the common template to be used for each row while creating the row and filling the data. This can be overridden at the row level when setting the data using the template key.You can create a new template without going to the Templates section using the **Create New** option from the drop-down list. You can also edit a template inline in the segment.

> ***Note:*** Only those templates that are created from **Project Explorer >Templates > Segments** are visible on the **Row Template** drop-down list.

Section Header Template

Specifies the common template to be used for each section while creating the section header and filling the data. This is optional parameter and if not provided the default template provided by each platform will be used. It can also be provided at each section level when setting the data. You can create a new template without going to the Templates section using the **Create New** option from the drop-down list. You can also edit a template inline in the segment.

> ***Note:*** Only those templates that are created from **Project Explorer >Templates > Segments** are visible on the **Row Header Template** drop-down list.

> ***Note:*** When a Section Header is provided along with the rows/items, the Section Header is "clamped" to the top of the scrollable area (on the Form) as one scrolls through a long list of items (for example, if you have a long list of contacts that all begin with the letter "A", the "A" header will be fixed at the top until you scroll down past the last "A" item). This behavior can be clearly seen iPhone's Contacts application.
> This behavior of Section Headers is available on iOS and Android platform and is enabled when the Screen Level Widget has been set to true.

Group Cells

Specifies whether all the rows in a segment should be grouped using a rounded corner background and border.

- If **On** is selected, the cells will not have rounded border.
- If **Off** is selected, the cells will have a rounded border.

Retain Selection

Specifies whether the segment should retain the selection made when the user navigates out of the form and revisits the form.

- If **On** is selected, the selection is retained when the user navigates to different form.
- If **Off** is selected, the selection is not retained.

Full Screen Widget

Specifies whether the widget should occupy the whole container or not. You must set the value to true if your segment has large data sets (more than 20 records with each record having more than 15 widgets) to facilitate a better reuse of the widgets and a different scrolling behavior.

- If **On** is selected, the widget occupies the whole container and there is a reduction in load time of the Segment as only few rows are loaded at the load time. The rest of the rows are loaded as

the user scrolls through the widget. But the scrolling speed reduces.

- If **Off** is selected, the widget does not occupy the whole container and load time of Segment increases because all the rows are loaded at the beginning. But the scrolling speed improves.

> ***Note:*** This property is available only when a Segment widget is placed in a VBox form.

Separator

Specifies if the segment should display the separator between the rows.

- If **On** is selected, the separator appears.
- If **Off** is selected, the separator is not displayed.

Separator Thickness

Specifies the thickness of the separator in pixels.

Separator Color

Specifies the color of the separator between rows of a Segment widget. Click the color sampler to open the color picker from where you can select a separator color.

Separator Transparency

Provide the desired transparency for the separator.

Show Scrollbars

Specifies if the scrollbars of the segment must be visible all the time.

- If **On** is selected, the scrollbars are displayed.
- If **Off** is selected, the scrollbars are not displayed.

Orientation

Specifies how you can stack the widgets within the Segment. You can set the orientation of the Segment as horizontal or vertical.

Following are the options:

- Horizontal: Enables you to stack the content within the Segment horizontally.
- Vertical: Enables you to stack the content within the Segment vertically.

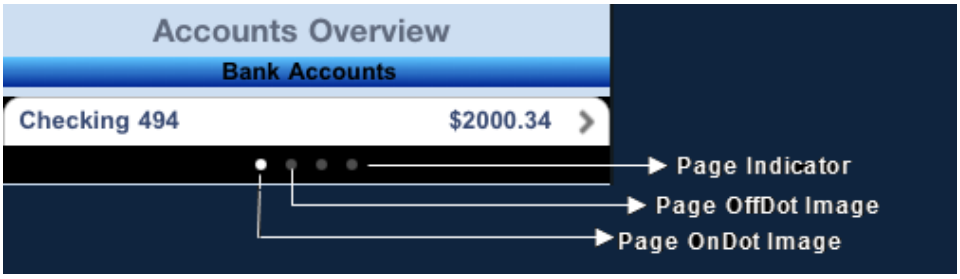
> ***Note:*** This property is available only when Segment widget is placed in a VBox form.

View Type

Specifies the view type of a segment. The following are the available view types that you can select and their appearances on iPhone native client:

- Table: The rows of the segment appear in a table as a list.

- Page: The rows of the segment appear in pages and you need to scroll through the pages to view the rows.



Layout Alignment

Specifies the direction in which the widgets are laid out.

Default: LEFT

The options are:

- Left: The widgets placed inside a Segment are aligned left.
- Center: The widgets placed inside a Segment are aligned center.
- Right: The widgets placed inside a Segment are aligned right.

> ***Note:*** This property is avialable only when Segment widget is placed in a VBox form.

Selection Behavior

Specifies whether the segment will support single or multiple selection.

Following are the options:

- Default: Indicates that the segment does not support either single or multiple selection. This option allows you to define an onRowClick event for the segment.
- Single Select: Indicates that you can make one selection when you have many choices in the segment (the behavior is similar to a RadioButtonGroup).
- Multi Select: Indicates that you can make more than one selection when you have many choices in the segment (the behavior is similar to a CheckBoxGroup).

Enable Reordering

For Kony Visualizer Version 7.3 and later, specifies whether to enable or disable reordering rows in a segment.

Autogrow Mode

This property is applicable only when the segment is placed inside a flex container and View Type is set as Table. It specifies the segment to grow when the new content is added.

Following are the options:

- None: Auto growth of a Segment is disabled.
- Autogrow-Height: Auto growth of a Segment is enabled.

Rules and Priorities of Autogrow-Mode property

- If the height of the Segment is not computable and **Autogrow Mode** property is configured as **Autogrow-Height**, then the height of the Segment will be the Preferred Height, and min and max constraints are applied on top of the Preferred Height computed.

  > ***Note:*** Preferred Height in the above statement refers to the cumulative height of the segment contents (rows - defined using templates, section headers / footers, separators etc.).

- If the height of the Segment is not computable and **Autogrow Mode** property is configured as **None**, then the height of the Segment will be the default value and min/max constraints are applied on top of the default value.
- The Autogrow Mode property gets preference, when the height of the Segment is specified as preferred and Autogrow Mode property is configured Autogrow-Height.
- If the Autogrow Mode property is not specified or the specified value is invalid, then the default value of the Autogrow Mode property is equal to specifying **None**.
- If Autogrow Mode property is specified, and the templates defined are specified in percentage, then the height of the row, which uses percentage template will be considered as the default row height of the template, which is 100dp.
- All the above rules are applied even if the template specified is a box template.
- When a Segment is generated using single bucket constructor, height property must be set as undefined for Autogrow Mode property to work.
- If the row template height is not specified and the Segment Autogrow Mode property is set as Autogrow-Height, then row height of the Segment is 220dp (preferred height of a flex container).

Height

Specifies the height of the Segment in terms of percentage. The percentage is with reference to the value of Height Reference property.

> ***Note:*** This property is unavailable on Flex Forms.

Height Reference

The Segment height percentage is calculated based on the option selected.

- Form Reference: The Segment height percentage is calculated based on the height of the form excluding headers and footers. This option is not respected if Segment is placed inside a popup or in templates.
- Parent Width: This option is used if the Segment is placed inside a popup or in templates. The width is calculated based on the width of the parent container.

> ***Note:*** This property is unavailable on Flex Forms.

Dictionary

Specifies whether the dictionary must be enabled for easy navigation.

If the dictionary property is enabled, alphabets from A to Z appear on the screen and when you select any alphabet, all the corresponding results that start with the selected alphabet are displayed.

> ***Note:*** This property is applicable if Screen Level Widget property is set to true and the section headers have been set.

- If **On** is selected, the dictionary is available.
- If **Off** is selected, the dictionary is not available.

> ***Note:*** This property is specific to the iOS platform.

Indicator

Specifies the indicator type as rowSelect, rowClick, or none. Based on your selection, the behavior is exhibited:

Following are the options:

If the user selects the indicator, the related content appears in the next screen .

- Click: Specifies the disclosure button. The button appears as follows:

If the user selects the disclosure button, the detailed content appears.

- None: No indicator or button appears.

> ***Note:*** This property is specific to the iOS platform.

Edit Style

Specifies the way in which the edit feature of Segment can be enabled.

Following are the options:

- Icon: An icon will be displayed on the left hand side of each row.
- Swipe: A delete or insert button will be shown on the right hand side of each row when the user performs a SWIPE gesture on the row. Whether an insert button or delete button is to be shown is controlled by the editmode property that is set using the data property of the Segment.
- None: No special edit styles are applied.

> ***Note:*** This property is specific to the iOS platform.

Progress Indicator

Specifies whether the progress indicator is displayed.

Default: true (the progress indicator appears on the widget)

- If **On** is selected, the progress indicator appears on the widget.
- If **Off** is selected, the progress indicator is not displayed on the widget.

> ***Note:*** This property is specific to the iOS platform.

Progress Indicator Color

Specifies the color of the progress indicator as white or grey.

- White: The progress indicator is white in color.
- Grey: The progress indicator is grey in color.

> ***Note:*** This property is specific to the iOS platform.

Scroll Bounces

Specifies whether the scroll view bounces past the edge of the content and back again.

- If **On** is selected, the scroll view bounce is applied.
- If **Off** is selected, the scroll view bounce is not applied.

> ***Note:*** This property is specific to the iOS platform.

Search Criteria

Specifies the search criteria to be applied when searching has been enabled.

> ***Note:*** This property applies only when the Full Screen Widget property is enabled, View Type is set to **Table**, and a template is selected for Search By property.

The options are:

- Starts With: The search is performed on the strings that start with the input string.
- Ends With: The search is performed on the strings that end with the input string.
- Contains: The search is performed on the strings that contain the input string.

> ***Note:*** This property is specific to the iOS platform.

Search By

Specifies the identifier of the widget placed inside the row of the Segment. Search will be performed against the content present inside the widget.

> ***Note:*** Note: This property is applicable only when Full Screen Widget property is enabled, **View Type** is set to **Table**, and a template is selected for Row Template property.

> ***Note:*** This property is specific to the iOS platform.

Dock Section Header

Specifies whether to dock the section header at the top of the segment while scrolling the section content. If you are scrolling the segment data, the next section header will be docked on top of the segment

> ***Note:*** This property applies only when the Full Screen Widget property is enabled and **View Type** is set to **Table**.

For example, if you scroll the segment data shown in the following figure, as the segment data scrolls up, the Samsung Phones docked header moves out oThe Samsung header f view. and is replaced with the HTC Phones section header, which is now docked.

> ***Note:*** This property is specific to the Android platform.

Swipe Config

From Kony Visualizer V9 onwards, you can configure swipe functionality for a Segment widget. The Swipe config settings enable users to dismiss a row or to reveal certain actions when they swipe a row to the left or right.

For more information on enabling the swipe functionality on the rows of a Segment using a row template, refer SwipeConfig.

Actions

Actions define what happens when an event occurs. On a Segment widget, you can run an action when the following events occur:

- **onRowClick:** The action is triggered when a user clicks any row of the widget.
- **onPull:** The action is triggered when the widget is pulled from the top.
- **onPush:** The action is triggered when the widget is pushed from the bottom.

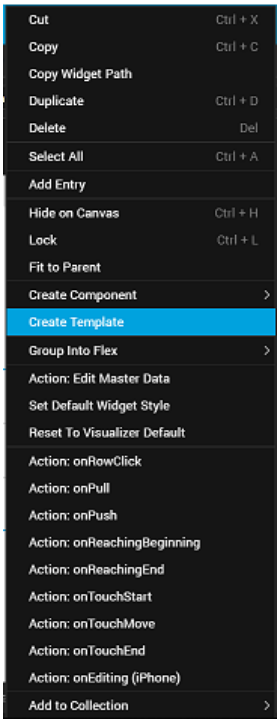
- **onReachingBegining:** The action is triggered when scrolling reaches the beginning of the widget.
- **onReachingEnd:** The action is triggered when scrolling reaches the end of the widget.
- **onTouchStart:** The action is triggered when the user touches the touch surface. This event occurs asynchronously.
- **onTouchMove:** The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- **onTouchEnd:** The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.
- **onEditing:** The action is triggered when a user edits the row. This action is only triggered if the Edit Style is set to SEGUI_EDITING_STYLE_ICON or SEGUI_EDITING_STYLE_SWIPE (iOS).

For more information, see Add Actions.

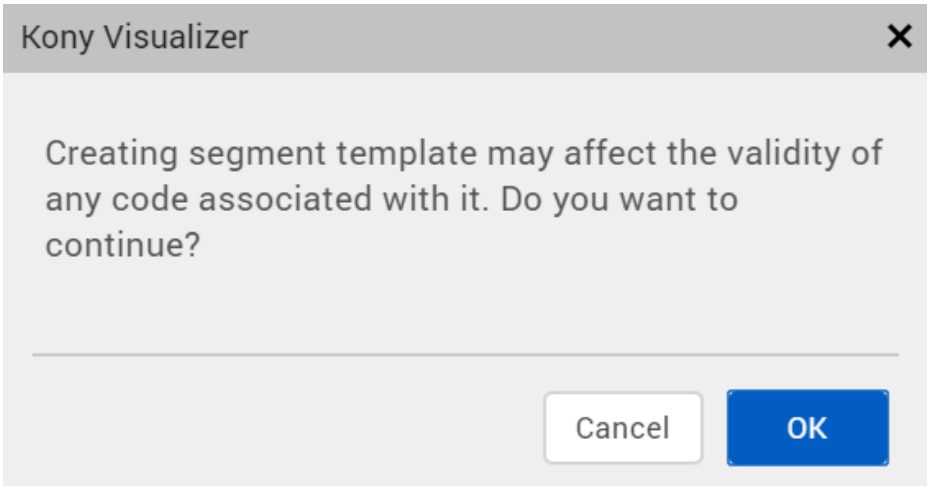
Create a User-defined Template from an Auto-generated Segment Template

To create a user-defined Template from an auto-generated Segment Template, follow these steps:

1. Right-click the Segment that contains the auto-generated Template and associated UI elements. A list of options appears.



2. Click **Create Template**. Kony Visualizer displays a dialog box stating that if you have added a code snippet for the widgets inside the Segment Template, the associated code will be cloned in the Segment Template; but, the references in the custom code will not be modified appropriately. You must manually make the changes to the references in the custom code.



3. Click **OK**. A new Segment Template, which is a clone of the selected Segment Template, is created and is available at Project Explorer > **Templates** > *channel* > **Segments**. Alternatively, you can right-click the original Segment Template and select the **Navigate to Template** option.

   All the properties of the Segment Template and its associated widgets are duplicated in the newly created Template.

   The cloned Template will be assigned as a Row Template of the selected Segment.

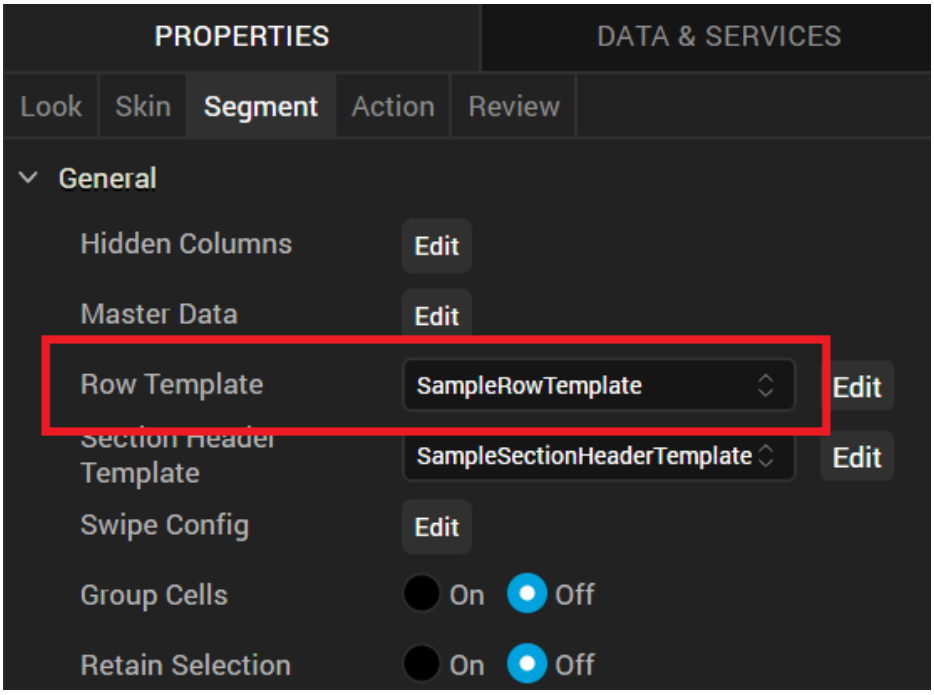
   You can only reuse the Segment Template in various forms of the same channel in which you created the Template.

Map Section Header Template Widgets by using Mapping Editor

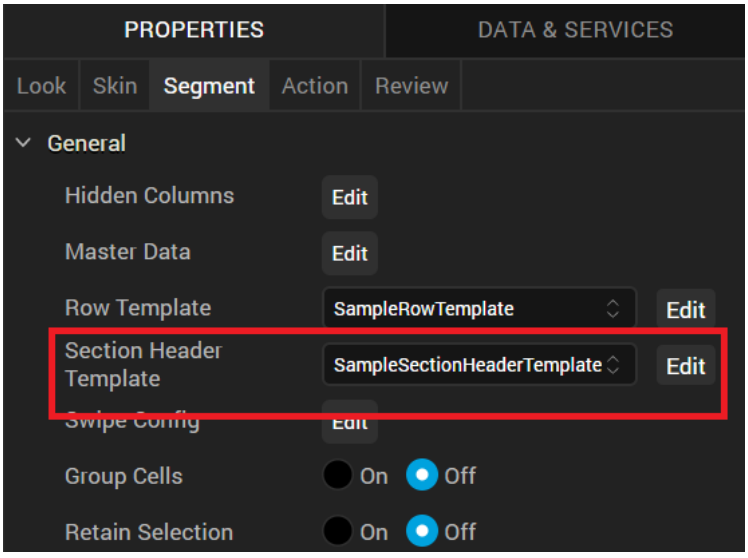
To map the widgets of the Section Header Template of a Segment by using Mapping Editor, follow these steps:

1. In Kony Visualizer, add a Segment widget to a form.
2. Select the Segment, and then go to **Properties** panel > **Segment**.

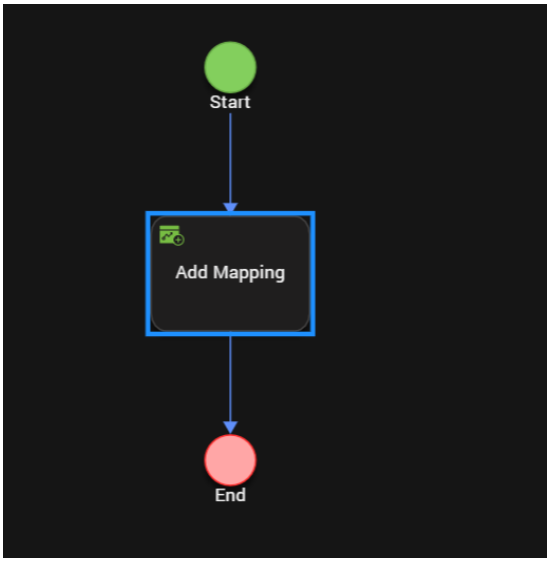
3. In the **Row Template** list box, select the **SampleRowTemplate** option.



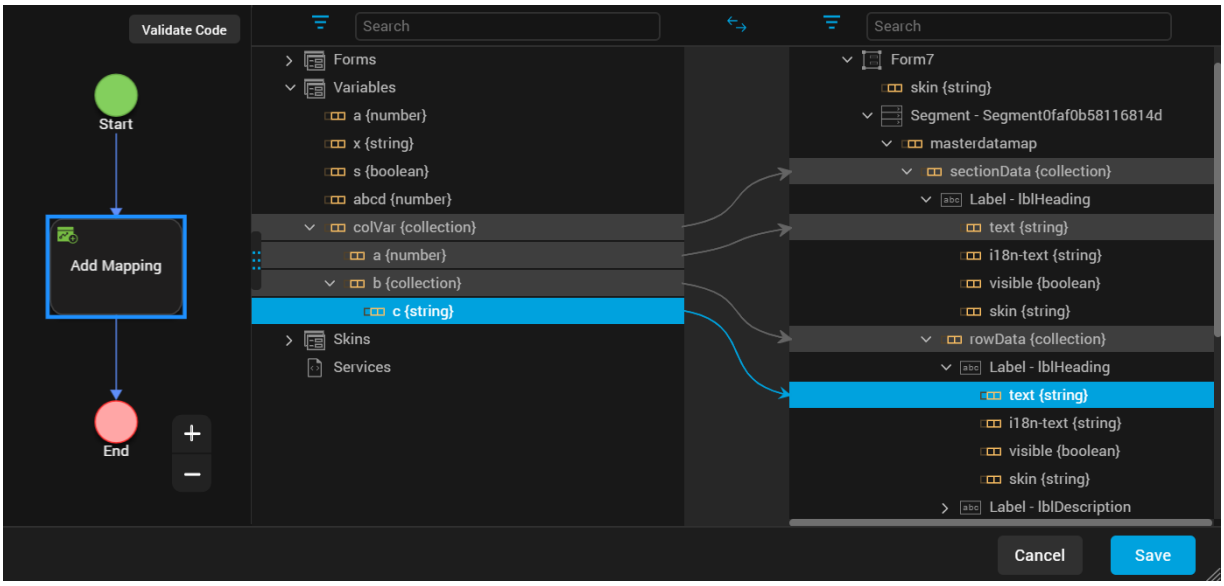
4. In the **Section Header Template** list box, select the **SampleSectionHeaderTemplate** option.



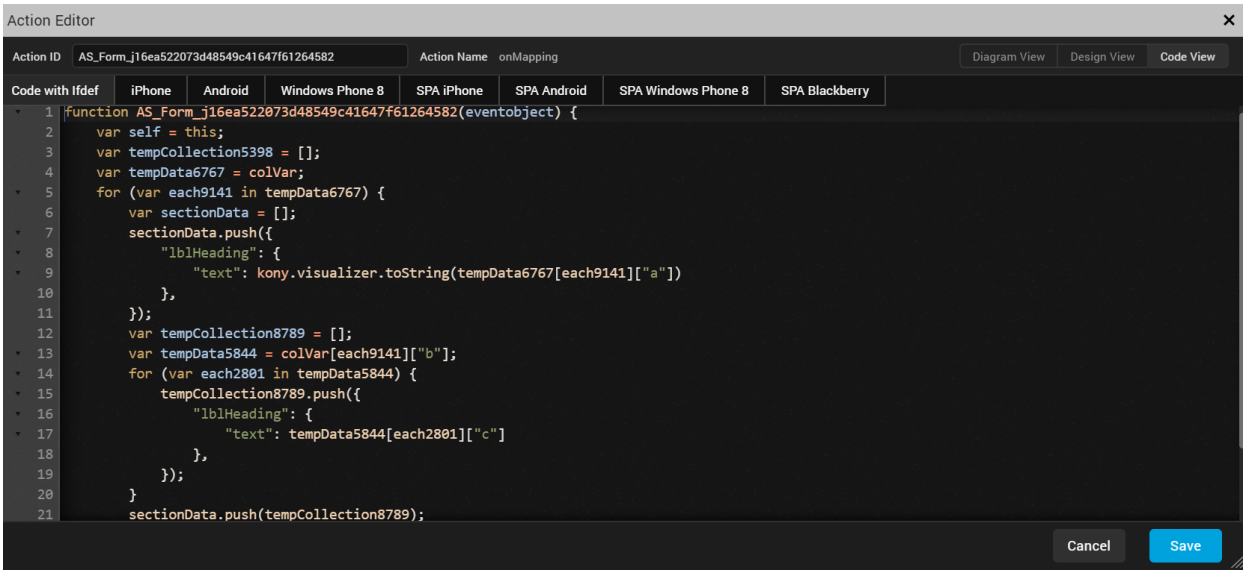
5. In Project Explorer, select the form to which you had added the Segment and go to **Properties** panel > **Action**.
6. For any Event (for example, *onMapping*), click **Edit**. The **Action Editor** window appears, with **Diagram View** open by default.
7. On the left pane of Action Editor, locate and click the **Add Mapping** action. The Add Mapping action is added to the flow diagram, as shown here.



8. Select **Add Mapping** from the flow diagram. Mapping Editor opens on the right pane of Action Editor.
9. Map the required global variables, services, and other items to the Segment **rowData** and **sectionData** as well as to associated rowData and sectionData widgets.



10. Click **Code View** to see the code details of the data mappings.



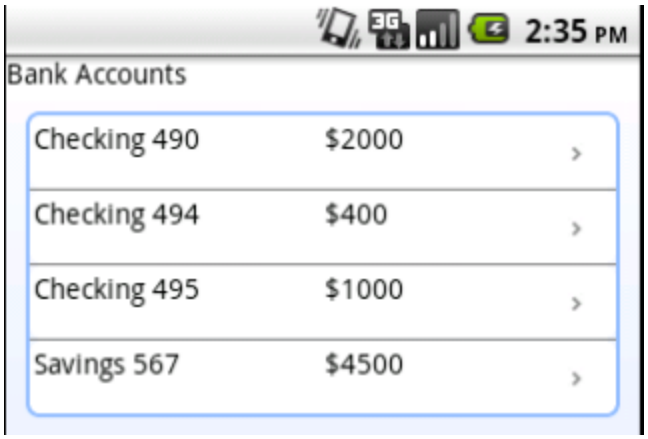
11. Click **Save**. You have successfully mapped the widgets of the Section Header Template and Row Data of a Segment by using Mapping Editor.

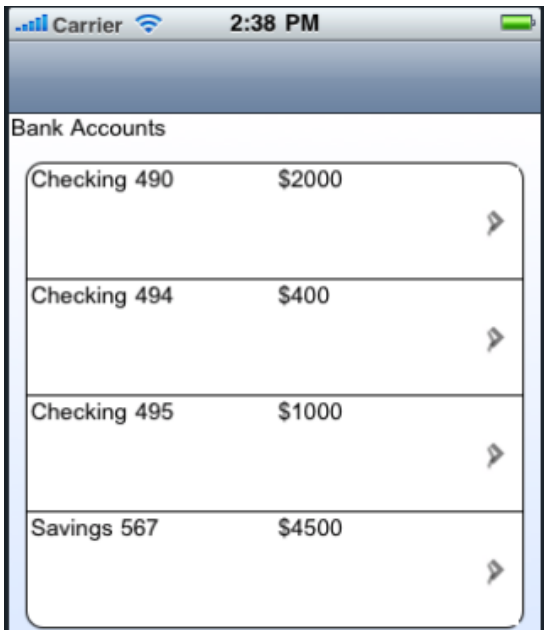
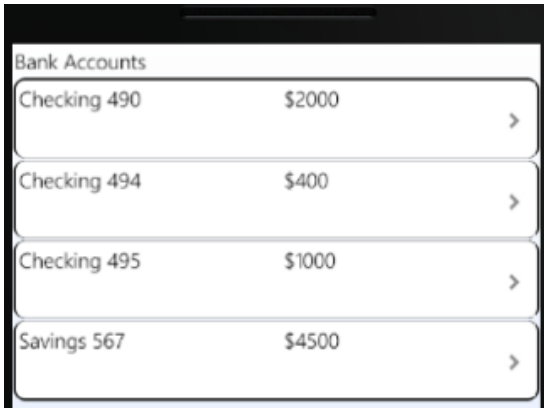
Remarks

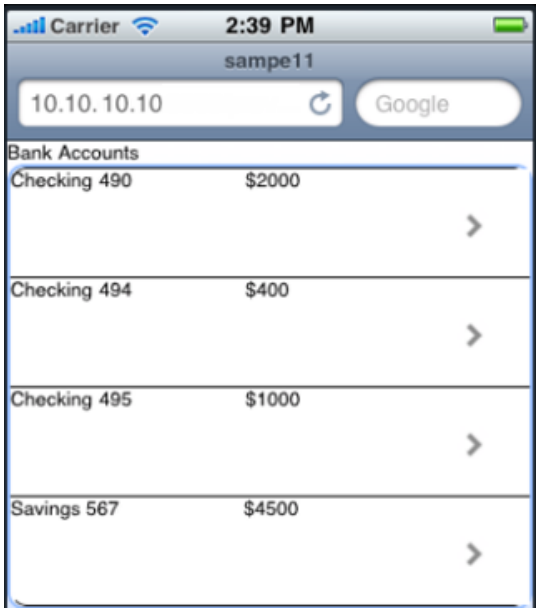
- For Kony Visualizer V8 SP4 or earlier projects, the elements that had been mapped to the widgets of Segment Master Data will now be mapped to the widgets of Segment Section Data.
- If you change any of the Segment Templates, all their associated mappings will be automatically deleted.

Widget Appearance on Platforms

The appearance of the Segment widget varies as follows.

Platform	Appearance
Android	

Platform	Appearance
iOS	
Windows Phone	

Platform	Appearance
SPA	

Switch

Use a Switch widget to let a user choose between two mutually exclusive choices or states, similar to the Switch control on an iPhone .

The Switch widget displays the value that is currently in effect. Slide the control to select or reveal the other value.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

> ***Note:*** The Switch widget is supported on iOS, Android, and Windows platforms, both Native and SPA.

Important Considerations

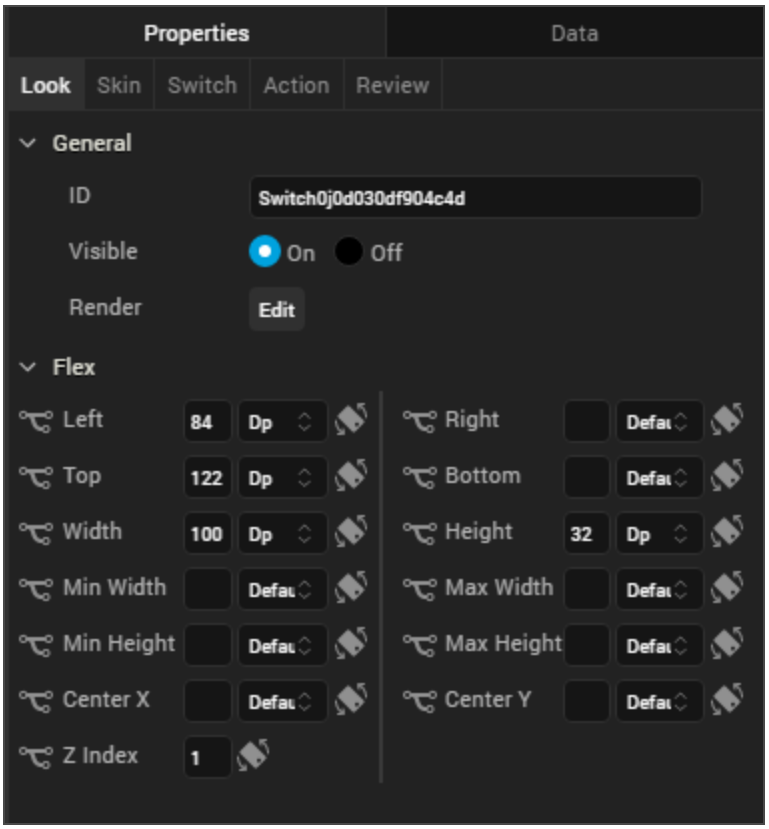
The following are important considerations for a Switch widget:

- Use a predictable pair of values so that the user does not have to slide the switch to know the other value.
- Consider using the Switch widget to change the state of other user interface elements in the view. For example, in an airline application booking screen you could use a Switch widget to let the user select between One Way and Round Trip, displaying a different set of user interface elements for each option.

Look

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



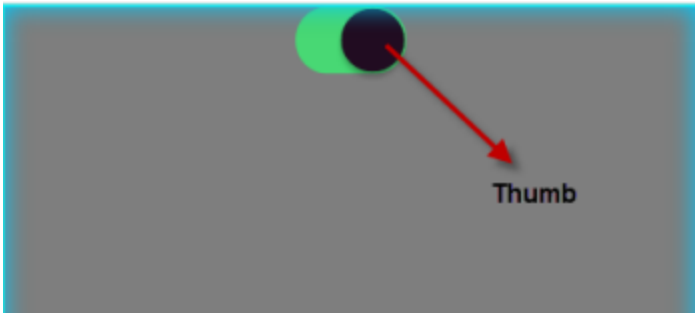
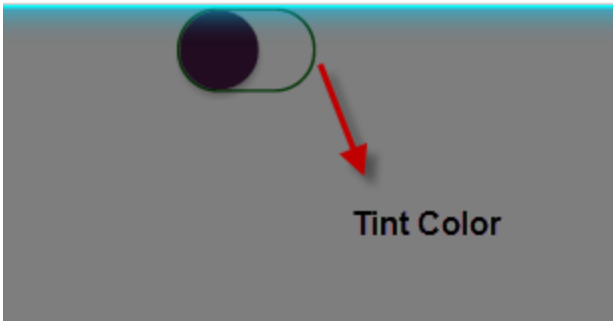
For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

For the Switch widget, you can apply a skin and its associated properties for the following states:

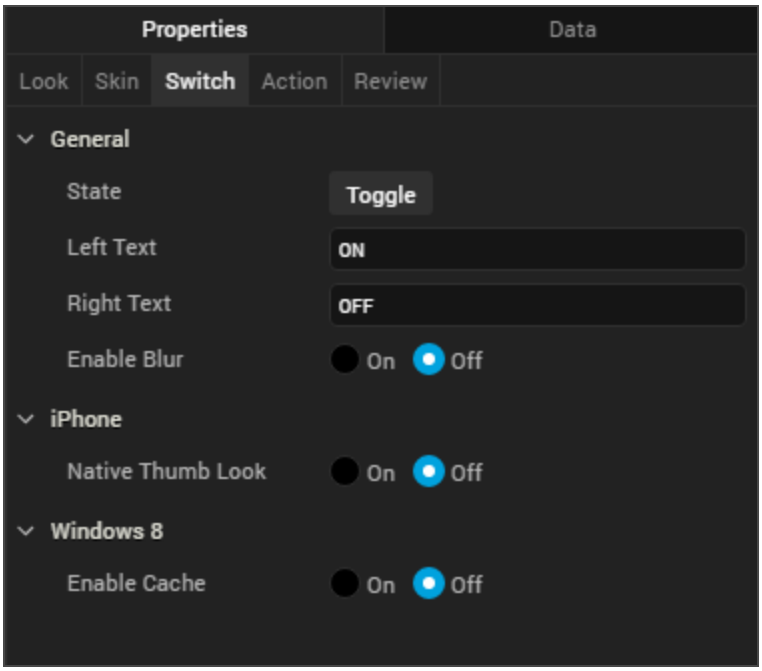
Skin	Definition
Normal	The default skin of a widget.
Focus	The skin applied when the focus is on a widget.

Skin	Definition
Thumb Color	The color of switch's slider button. This property is available under BackGround when you fork the skin for iPhone Native, and also when you fork the entire form for Android Native. 
Tint Color	This property is available under BackGround only when you fork the skin for iPhone Native. 

For more information about applying skins, see Understanding Skins and Themes.

Switch Properties

Switch properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



State

Lets you toggle the switch colors.

Left Text

Specifies the text to be displayed on the left portion of the switch.

Right Text

Specifies the text to be displayed on the right portion of the switch.

Actions

Actions define what happens when an event occurs. On a Switch widget, you can run an action when the following events occur:

- onSlide: The action is triggered when there is a change in the default selected value.
- onTouchStart: The action is triggered when the user touches the touch surface. This event occurs asynchronously.

- onTouchMove: The action is triggered when the touch moves on the touch surface continuously until movement ends. This event occurs asynchronously.
- onTouchEnd: The action is triggered when the user touch is released from the touch surface. This event occurs asynchronously.

For more information, see Add Actions.

Placement Inside a Widget

The following table summarizes where a Switch widget can be placed:

Widget	Switch placement inside a widget
Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation - yes Vertical Orientation- Yes
Tab	Yes
Segment	Yes
Popup	Yes
Template	Header- No Footer- No

Video

Use a Video widget to stream video within a form.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Important Considerations

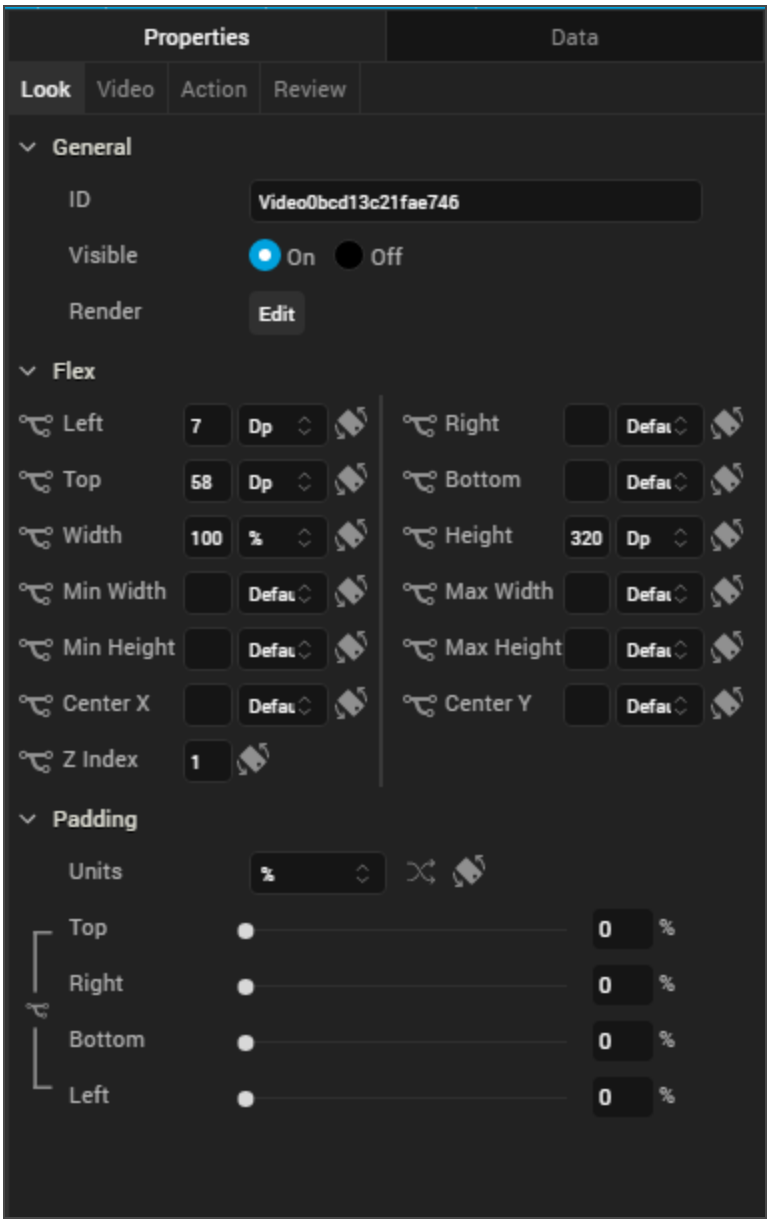
The following are important considerations for a Video Widget:

- When the form contains dockable components such as headers, footers, or a menu bar, scrolling the video widget on iPhone or iPad does not scroll the form since Video controls on and iPhone or iPad do not respond to custom touch events when media controls are present. To enable scrolling of the form, apply left or right margins to the either side of the video widget.
- You can play only one video at a time.
- Video widgets are available only for SPA platforms.

Look

Look properties define the appearance of the widget. The following are the major properties you can set:

- Whether the widget is visible.
- The platforms on which the widget is rendered.
- How the widget aligns with its parent widget and neighboring widgets.
- If the widget displays content, where the content appears.



For descriptions of the properties available on the Look tab of the Properties pane, see Look.

Skin

Skin properties define a skin for the widget, including background color, borders, and shadows. If the widget includes text, you can also specify the text font.

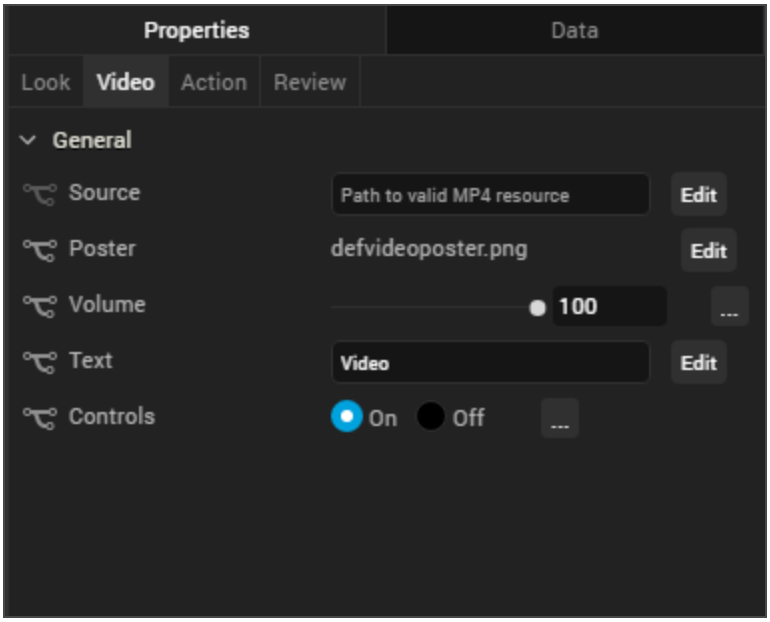
For a Video widget, you can apply a skin and its associated properties for the following state:

- Normal: The default skin of a widget.

For more information about applying skins, see Understanding Skins and Themes.

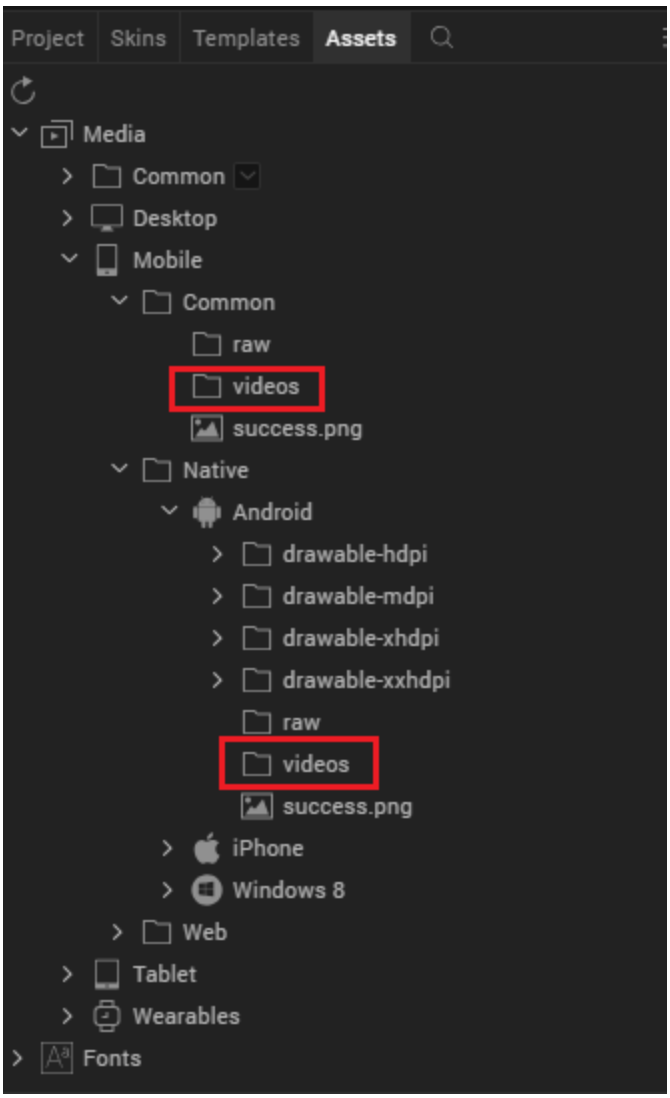
Video Properties

Video properties specify properties that are available on any platform supported by Kony Visualizer, and assign platform-specific properties.



Source

Specifies the video's source. You can add the videos that you want to be part of the app in its respective channels common folder. For example, **Native>Android>Common>Video**.



Prior to Kony Visualizer V8 SP1 release, video files were inside the **raw** folder. Video files are now moved into a separate **video** folder. Project built on versions lower that 8.1 will continue to work without any changes.

> ***Note:*** The supported video formats are MP4, WebM and OGV. The device will play video only if the format is compatible with the underlying SDK

Controls

Specifies whether to display the default video controls. When this property is set to false, video is playable by clicking anywhere on the Poster.

> ***Note:*** This property is specific to the SPA and Desktop Web platforms.

Poster

Specifies an image to be displayed as a poster or as a starting image for the video. The image location must point to an external URL. For example, www.kony.com/sites/all/themes/kony/logo.png

> ***Note:*** This property is specific to the SPA and Desktop Web platforms

Review

Displays review feedback. For more information, see Capture Product Requirements with Review Notes.

Placement Inside a Widget

The following table summarizes where a Video widget can be placed:

Widget	Video placement inside a widget
Flex Form	Yes
VBox Form	Yes
FlexContainer	Yes
FlexScrollContainer	Yes
HBox	Yes
VBox	Yes
ScrollBox	Horizontal Orientation -Yes Vertical Orientation- Yes

Widget	Video placement inside a widget
Tab	Yes
Segment	No
Popup	Yes
Template	Header- No Footer- No

Specifying a Video

To specify a video:

1. Click the **Edit** button to open the Source dialog box.
2. Do one of the following:
    - Locate and select the video you want from the list of available videos.
    - In the video URL text box, enter a video URL.
3. Click **Open**. The Source property displays the file name or URL of the video.

Add Watch Widgets

Widgets are the building blocks of a screen (form) in a digital application, and each one has a specific purpose, such as user interaction or animation. Kony Visualizer provides you with built-in widgets that help you achieve your required functionality. You can configure every widget based on your needs.

This topic covers widgets for the Watch channel. For information regarding widgets for the Mobile, Tablet, and Desktop channels, see Populate Screens with Widgets.

Kony Visualizer provides three types of widgets for the Apple Watch: Container, Basic, and Advanced.

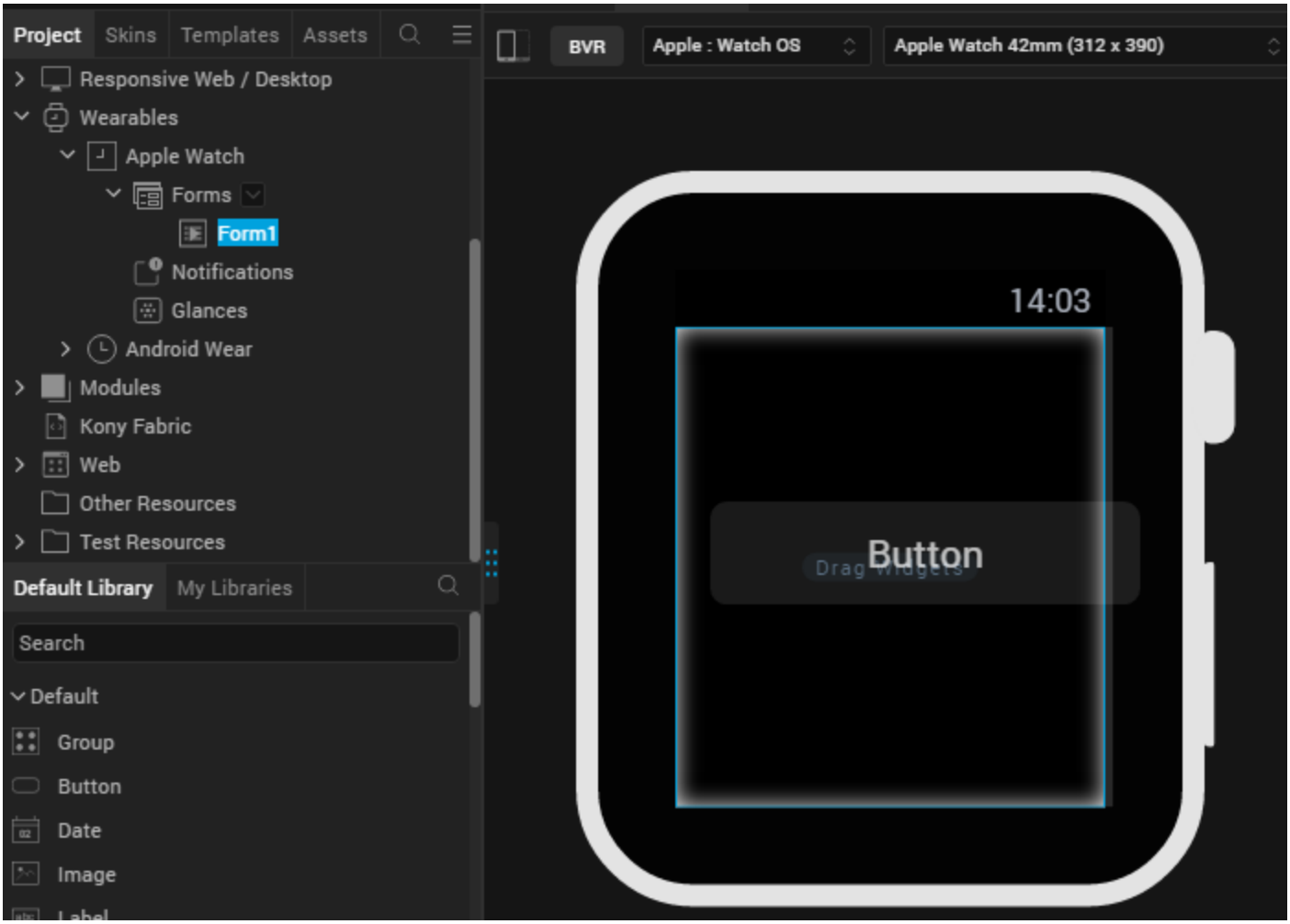
Click a topic for more information.

Access widgets from the Widget Tab

The Three Types of Widgets

Access Widgets from the Widget Tab

The widgets are located on the Widget tab of the Kony Library pane. To add widgets to a form, simply drag and drop them into place. As you do so, alignment guides display to guide your positioning of the widget.



Dragging a button widget onto a Watch form.

The Three Types of Widgets

Kony Visualizer provides you with three types of widgets.

Container Widget

Basic Widgets

Advanced Widgets

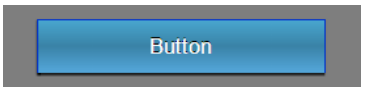
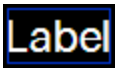
Container Widget

The container widget category for the Watch channel consists of a single widget, the Group container widget. You can add any of the Watch widgets directly onto a Watch form without using the Group container. But if you want to structure your form with grouped widgets that align on either a horizontal or vertical axis, you can use the Group container. This widget behaves similarly to HBox and VBox forms, which were the de facto method for adding widgets to a form in Kony Studio 6.0 and earlier. A Group container can orient the widgets it contains either horizontally, or vertically. By nesting one or more vertically-oriented Group containers within a horizontally-oriented Group container, and then adding widgets to those containers, you can create a grid of widgets. Similarly, you can also nest horizontally-oriented Group containers within a vertically -oriented Group container. With the right combination and alignment of horizontal and vertical Group containers, you can position a widget anywhere on the Watch form. For more information about the Group container widget, see Group Widget.

Two vertically-oriented Group containers, each containing four button widgets, nested within a single, horizontally-oriented Group container.

Basic Widgets

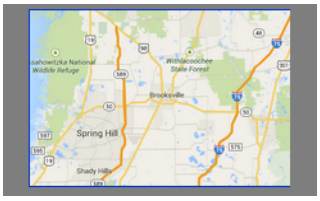
Basic Widgets help you build the user interface in your application. Along with designing the application, you can also configure events to these widgets or write code snippets to create functionality.

Widget	Description	Image
Button	The Button widget provides input to an application or to trigger an event. For example, you can navigate to a form, interact with a dialog box, or confirm an action.	
Date	The Date widget can display a date, a time of day, or both. It does not have an Actions tab.	
Image2	The Image2 widget is a non-interactive widget that you can use to display a local image file. You can use an Image2 widget in scenarios such as displaying your company's logo, displaying a snapshot, and providing an illustration.	
Label	The Label widget displays non-editable text on a form and is non-interactive.	
Line	The Line widget is a non-interactive widget that provides a means of visually separating widgets from one another on the Watch screen.	
Slider	With the Slider widget, you select a value from a defined range of values by moving the thumb (an indicator) in a horizontal direction.	

Widget	Description	Image
Timer	The Timer widget displays a non-editable countdown on a form and is non-interactive.	

Advanced Widgets

Advanced widgets provide you the capability to achieve the most commonly used functionality in your application. These Widgets are developed by Kony. Also, the option to configure properties of the widgets is provided.

Widget	Description	Image
Map	The Map widget displays locations defined by latitude and longitude on an on-screen map.	
Segment2	The Segment2 widget consists of multiple segments (rows or records), and each segment (row or record) can have multiple child widgets.	
Switch	The Switch widget presents two mutually exclusive choices or states.	

Group Widget

The container widget category for the Watch channel consists of a single widget, the Group container widget. You can add any of the Watch widgets directly onto a Watch form without using the Group container. But if you want to structure your form with grouped widgets that align on either a horizontal or vertical axis, you can use the Group container. This widget behaves similarly to HBox and VBox

forms, which were the de facto method for adding widgets to a form in Kony Studio 6.0 and earlier. A Group container can orient the widgets it contains either horizontally, or vertically. By nesting one or more vertically-oriented Group containers within a horizontally-oriented Group container, and then adding widgets to those containers, you can create a grid of widgets. Similarly, you can also nest horizontally-oriented Group containers within a vertically -oriented Group container. With the right combination and alignment of horizontal and vertical Group containers, you can position a widget anywhere on the Watch form.

Two vertically-oriented Group containers, each containing four button widgets, nested within a single, horizontally-oriented Group container.

Click any of the following to learn about the properties found on the tabs of the Group container widget.

Look Tab

Skin Tab

Group Tab

Action Tab

Review Tab

Look Tab

On the Look tab, you define properties and behaviors related to a Group container's appearance and position. The following sections describe each of the properties.

ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** text box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

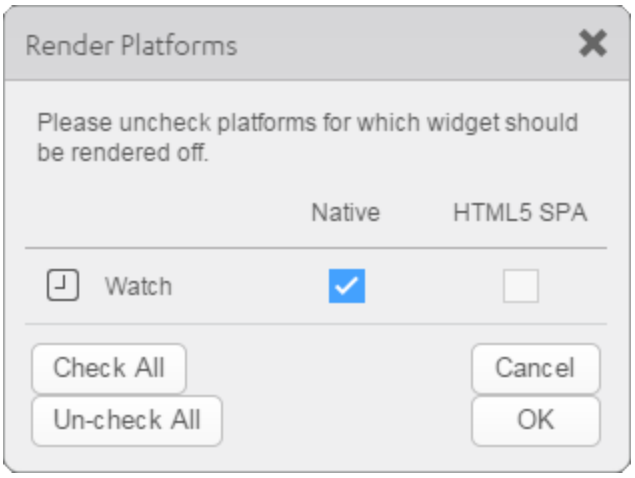
Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how a widget's boundaries are aligned with respect to its parent. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

Width determines the width of the widget as measured along the x-axis.

Following are the options that can be used as units of width:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Height

Height determines the height of the widget as measured along the y-axis (height of the parent). You can use any of the following options:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Padding

Defines the space between the content of the widget and the widget boundaries. You can use this option to assign the top, left, right, and bottom distance between the widget content and the widget's boundaries.

> *Important:* Default padding for Android is not set, as the devices are manufactured with predefined padding values.

Property	Definition	Action
Top	Top padding	Move the slider to adjust the top padding of the widget.
Bottom	Bottom padding	Move the slider to adjust the bottom padding of the widget.
Left	Left padding	Move the slider to adjust the left padding of the widget.
Right	Right padding	Move the slider to adjust the right padding of the widget.

> ***Notes:***
>
> - When you click the Uniform Padding button  , changing the value for one padding boundary changes all of them to the same value.
> - Modifying a widget's padding affects the padding of its parent and its children.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.

In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one button state: Normal.

General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Background

Under the Background section of the **Skin** tab, you can set the type of background you want to use, and set the color and its opacity.

Type

For the Watch channel, the Group widget is capable of two types of backgrounds:

Background Type	Description
Single Color	Applies a uniform, single color as the background of the skin that you choose.
Image	Applies an image of your choosing as the background of the skin. The image stretches to fill the dimensions of whatever widget the skin is applied to.

Color

If you select Single Color as the background type, you can configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing.

Opacity

Similarly, with Single Color as the background type, you can configure the opacity of the background color. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

Border

The Border section on the Skin tab of the Group container widget determines the degree to which the corners of the Group widget are rounded. The style of the rounding is set as **Custom** in the **Style** drop-down list. By changing the radius of the border, you can change how rounded the corners are. The higher the number in pixels, the more rounded the border. To see the rounding, make sure you have the background color set to something other than black, and the opacity higher than 20% or so.

Group Tab

Widgets placed within a Group container widget cannot overlap each other. They are aligned either horizontally, like a row of widgets, or vertically, like a column. On the Group tab, you can set the following properties:

Orientation. Specifies whether the widgets inside the Group widget are oriented horizontally, like a row, or vertically, like a column.

Spacing. Specifies the amount of space in Dp (device-independent pixels) you want to have in between the widgets that are inside the Group widget.

Opacity. Specifies the degree to which the Group widget is transparent or opaque. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Action Tab

On this tab, you define the events that are executed when an action is run. A Group widget supports only one action.

onClick. This action is invoked by the platform when the user performs a click action on the button.

For more information, see Add Actions.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Button Widget for Watch

A button is a control that you use to provide input to an application or trigger an event. For example, navigating to a form, interacting with a dialog box, or confirming an action.

Click any of the following to learn about the properties found on the tabs of the Button widget.

Look Tab

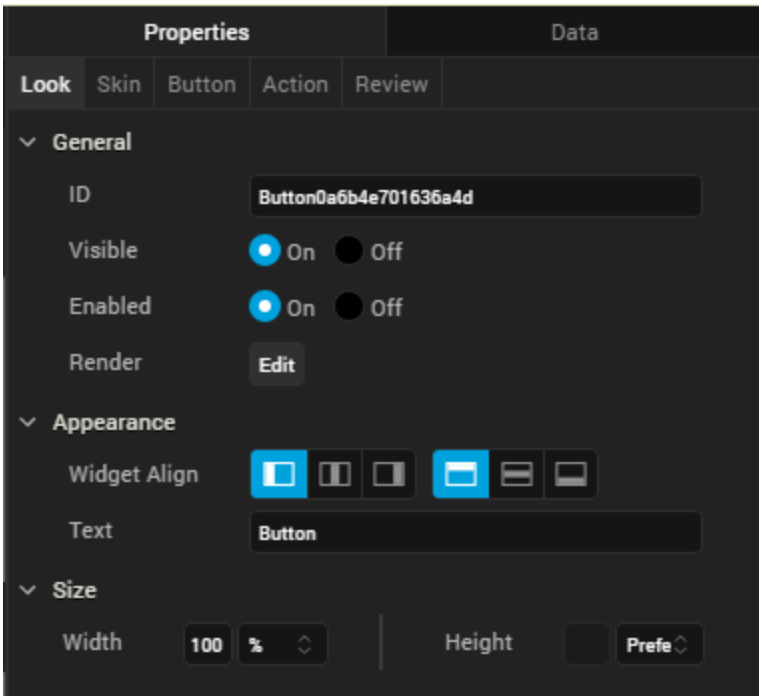
Skin Tab

Button Tab

Review Tab

Look Tab

On the **Look** tab, you define properties and behaviors related to a button widget's appearance and position. The following sections describe each of the properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On** .
- To make a widget invisible, click **Off**.

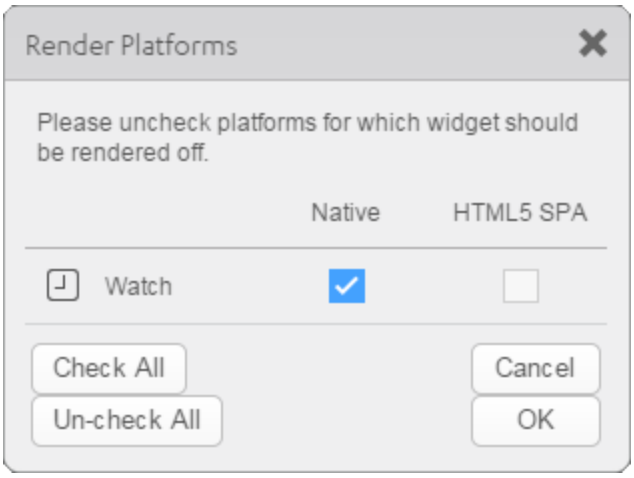
Enabled

Controls whether the widget is functional or not. You can programmatically make the widget functional or nonfunctional through an action sequence, triggered by the user.

- To enable the widget, click **On**.
- To disable the widget, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how the edges of the Button widget are aligned with respect to its parent's edges. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Text

Specifies the text that the user sees when running the app.

Width

The Width property sets the x-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the width of the widget.

% Specifies the width as a percentage of the parent's dimensions.

Dp Specifies the width in terms of device independent pixels.

Preferred Size Specifies an optimal width for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Height

The Width property sets the y-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the height of the widget.

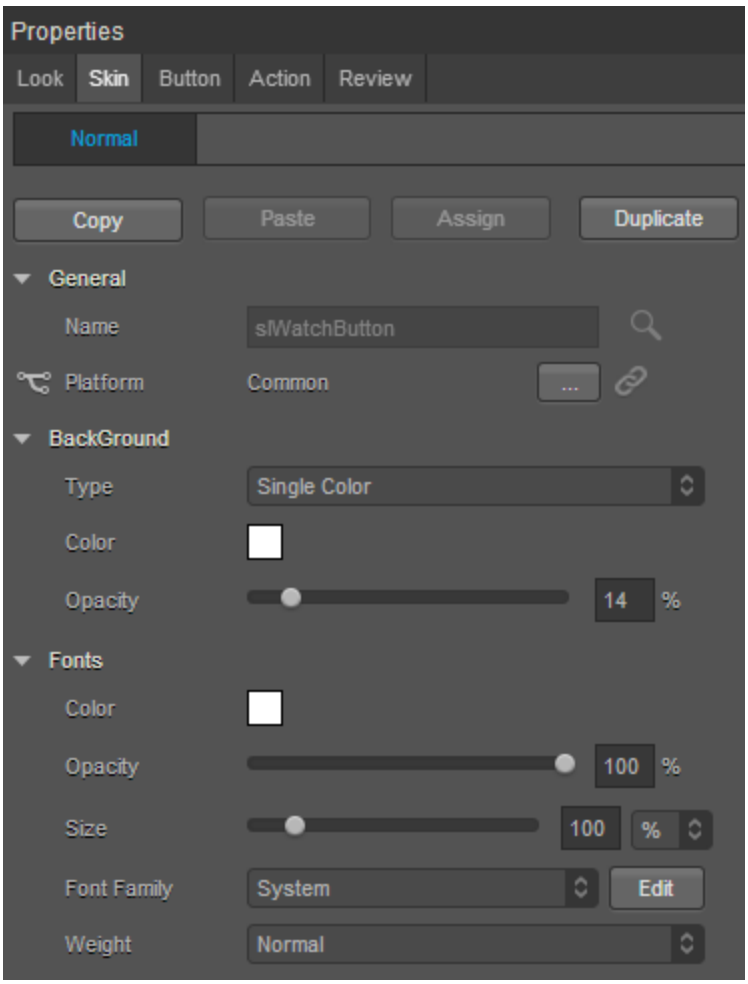
%. Specifies the height as a percentage of the parent's dimensions.

Dp. Specifies the height in terms of device independent pixels.

Preferred Size. Specifies an optimal height for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.



In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one button state: Normal.

General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available button skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Background

Under the Background section of the **Skin** tab, you can set the type of background you want to use, and set the color and its opacity.

Type

For the Watch channel, the Button widget is capable of two types of backgrounds:

Background Type	Description
Single Color	Applies a uniform, single color as the background of the skin that you choose.
Image	Applies an image of your choosing as the background of the skin. The image stretches to fill the dimensions of whatever widget the skin is applied to.

Color

If you select Single Color as the background type, you can configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing.

Opacity

Similarly, with Single Color as the background type, you can configure the opacity of the background color. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

Fonts

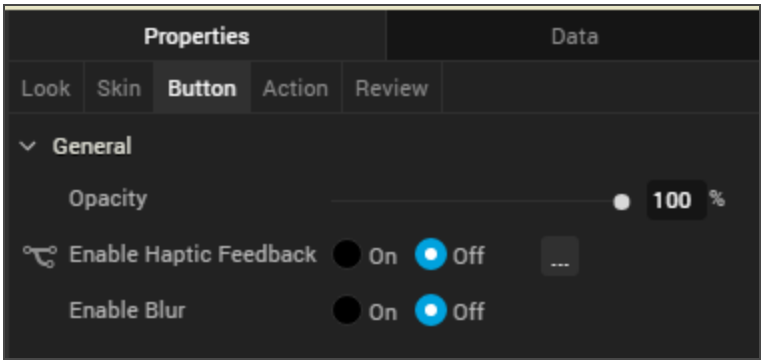
Under the Fonts section of the **Skin** tab, you can set the following properties.

Property	Description
Color	Sets the color that you want the font to be. You configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing
Opacity	Sets the degree to which the background color is transparent or opaque. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.
Size	You can set the font size by pixels (0 to 600) or as a percentage (0 to 600) of the baseline font size of 28 pixels.
Font Family	Currently, the System font is the only font family supported by Kony Visualizer for the Label widget.
Weight	You can set the weight of the font either to Normal, which is the default, or Bold.

For more information, see Understanding Skins and Themes.

Button Tab

On this tab, you configure properties unique to the Button widget. Currently, there is only one.



Opacity. Sets the degree to which the widget is transparent or opaque. By default, the opacity is set to 100, making the widget completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Action Tab

On this tab, you define the events that are executed when an action is run. On a Button widget, you can run the following action:

- **onClick.** This action is invoked by the platform when the user performs a tap action on the button.

For more information on using this action, see Add Actions.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Date Widget for Watch

The Date widget can display a date, a time of day, or both. It does not have an **Actions** tab.

Click any of the following to learn about the properties found on the tabs of the Date widget.

Look Tab

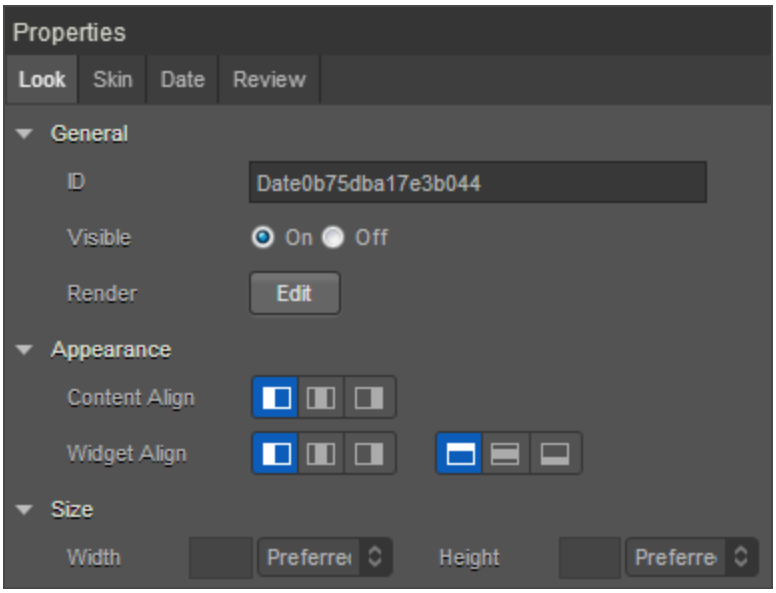
Skin Tab

Date Tab

Review Tab

Look Tab

On the **Look** tab, you define properties and behaviors related to a Date widget's appearance and position. The following sections describe each of its properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** text box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

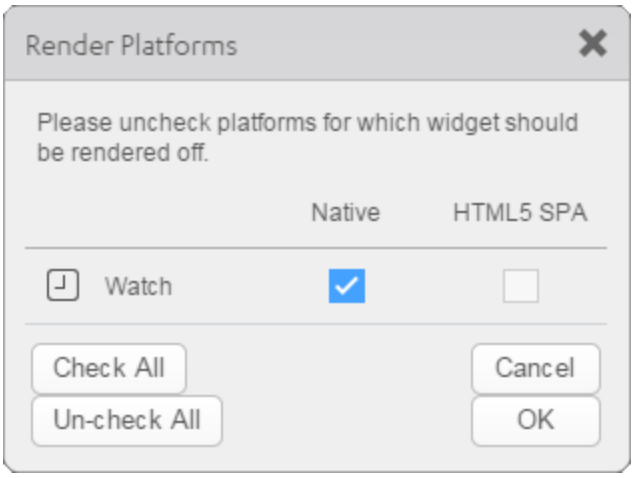
Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how a widget's boundaries are aligned with respect to its parent. The following alignment options are available:

Content Alignment

	Aligns the left edge of the content with the left edge of the widget.
	Aligns the horizontal center of the content with the horizontal center of the widget.
	Aligns the right edge of the content with the right edge of the widget.

Widget Alignment

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

Width determines the width of the widget as measured along the x-axis.

Following are the options that can be used as units of width:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Height

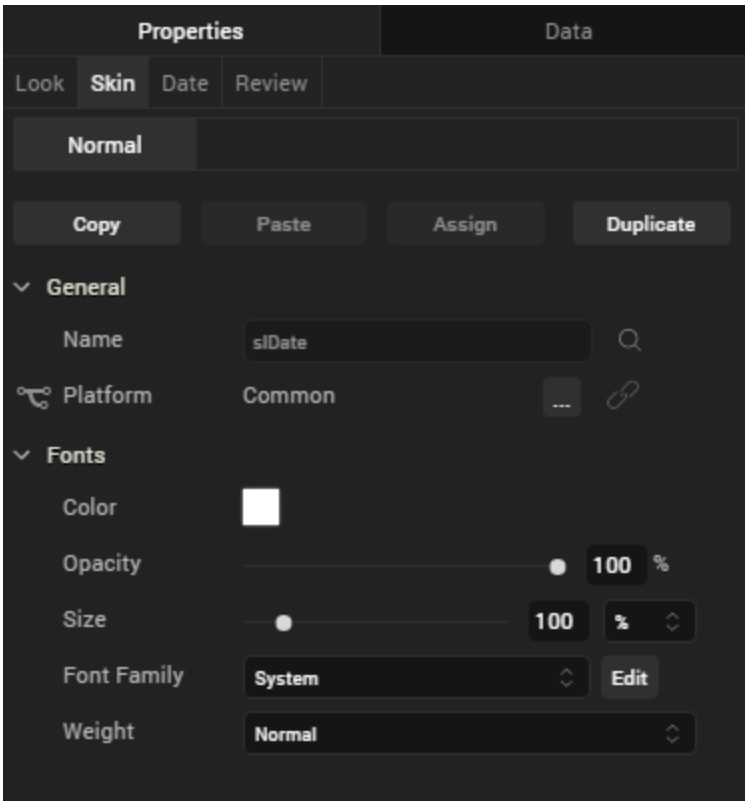
Height determines the height of the widget as measured along the y-axis (height of the parent). You can use any of the following options:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.

In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one state: Normal.



General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available date skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Fonts

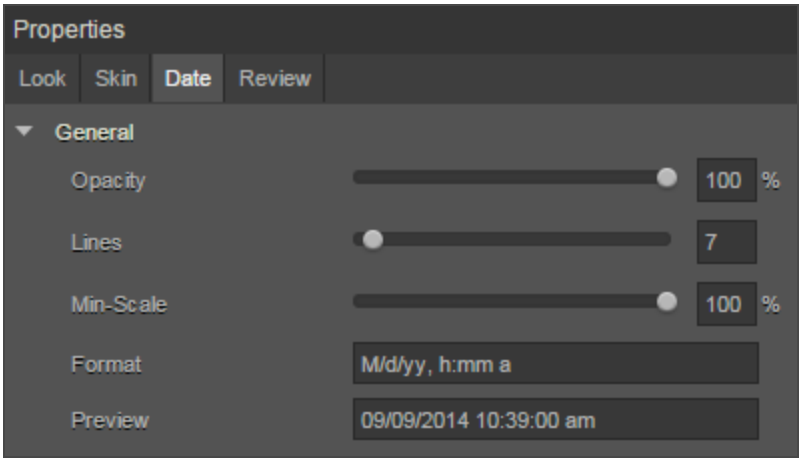
Under the Fonts section of the **Skin** tab, you can set the following properties.

Property	Description
Color	Sets the color that you want the font to be. You configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing
Opacity	Sets the degree to which the background color is transparent or opaque. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.
Size	You can set the font size by pixels (0 to 600) or as a percentage (0 to 600) of the baseline font size of 28 pixels.
Font Family	Currently, the System font is the only font family supported by Kony Visualizer for the Date widget.
Weight	You can set the weight of the font either to Normal, which is the default, or Bold.

For more information, see Understanding Skins and Themes.

Date Tab

On this tab, you configure properties unique to the Date widget.



Opacity. Sets the degree to which the widget is transparent or opaque. By default, the opacity is set to 100, making the widget completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Lines. Sets the maximum number of lines allowed for the label text. Text that does not fit on the specified number of lines is truncated.

Min-Scale. Sets the amount by which the font may be scaled to accommodate text. Values must be 100% or less. Specifying a value of 0 causes the Apple WatchKit to use the default scaling behavior, which allows the font to be scaled to no less than 80% of the original font size.

Format. Sets the manner in which the date and time is displayed. The following table outlines the available formats. These values are case-sensitive. You can mix and match these date and time values however you wish, such as `MM/dd/yyyy` or `yy-d-M`.

Element	Value	Description	Example
Day	d dd	Displays days 1-9 with one digit and 10-31 with two digits. Displays all days with two digits, placing a zero in front of days 1-9.	7 12 07 12
Month	M MM	Displays months 1-9 with one digit and 10-12 with two digits. Displays all months with two digits, placing a zero in front of months 1-9.	3 10 03 10

Year	yy or YY yyyy or YYYY	Displays the year using its last two digits. Displays the year as a four digit number.	16 2016
Hour	h H hh HH	Displays in 12-hour time hours 1-9 with one digit and 10-12 with two digits. Displays in 24-hour time hours 1-9 with one digit and 10-12 with two digits. Displays in 12-hour time all hours with two digits, placing a zero in front of hours 1-9. Displays in 24-hour time all hours with two digits, placing a zero in front of hours 1-9.	8 08
Minute	mm	Displays all minutes with two digits, placing a zero in front of minutes 1-9.	21
Seconds	ss	Displays all seconds with two digits, placing a zero in front of seconds 1-9.	03
Other	a v G Z	Displays either AM or PM, depending on the value in the Preview. Indicates either Standard or Daylight Time, depending on the time of year. Indicates AD (anno Domini). Indicates the hour in Greenwich Mean Time.	PM Eastern Standard Time AD -0400

Preview. Used for entering a placeholder value to see how the date and time are rendered in the Date widget on the Visualizer Canvas.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Image2 Widget for Watch

The Image2 widget is a non-interactive widget that you can use to display a local image file. Kony Visualizer version 7.3 suppcrts PNG, JPEG, and GIF image formats. Kony Visualizer version 7.2 and earlier supported PNG image format only.

To use images in Kony Visualizer, you copy the images to a specific folder in the Workspace and then use the Image2 widget to insert the image in a form. You can see what images are a part of your project on the **Assets** tab of the Project Explorer. You can also specify images for a specific channel and then use them in your application. For more information, see Adding and Managing Images and Other Media.

Be sure you name your images correctly by following the guidelines in Image Naming Conventions.

Click any of the following to learn about the properties found on the tabs of the Image2 widget.

Look Tab

Skin Tab

Image Tab

Review Tab

Image Naming Conventions

If an image file has an invalid file name, Kony Visualizer does not list it in the Media folder of the **Assets** tab. In Kony Visualizer version 7.2 and earlier, Kony Visualizer does not list it the file in the Media folder of the **Assets** tab if it is not a PNG file. You will want to be sure to use the following conventions when naming image files:

- For Kony Visualizer version 7.2 and earlier, the file format for images must be PNG.
- The file name must contain only lowercase characters.
- The file name must start with a letter.
- Numbers are allowed as long as they are not the first character in the file name.

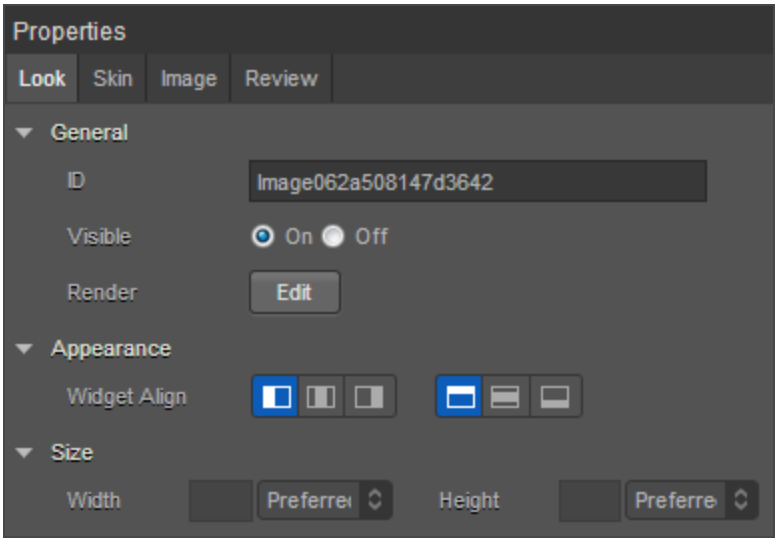
- Do not use special characters, such as dashes and underscores.
- Do not use any reserved JavaScript words or keywords (of JavaScript) as the file name for images.

The following table shows a few examples of valid and invalid file names for images:

Valid File Names	Invalid File Names	Remarks
myicon.png	Myicon.png	The invalid file name contains an uppercase character.
icon2.png	icon_2.png	The invalid file name contains an underscore.
accntsummary.png	accnt&summary.png	The invalid file name contains a special character.
accountdetails.png	2details.png	The invalid file name begins with a number.
companylogo.png	company logo.png	The invalid file name contains a space.
flightstatus123.png	continue.png	The invalid file name contains a Java keyword.

Look Tab

On the **Look** tab, you define properties and behaviors related to a Image2 widget's appearance and position. The following sections describe each of its properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** text box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

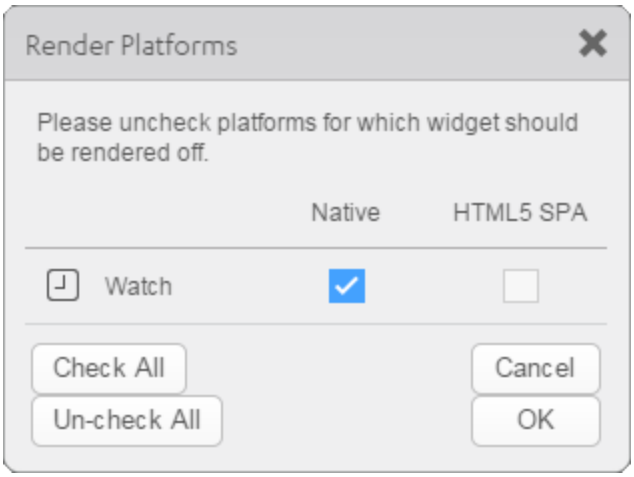
Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how a widget's boundaries are aligned with respect to its parent. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

Width determines the width of the widget as measured along the x-axis.

Following are the options that can be used as units of width:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Height

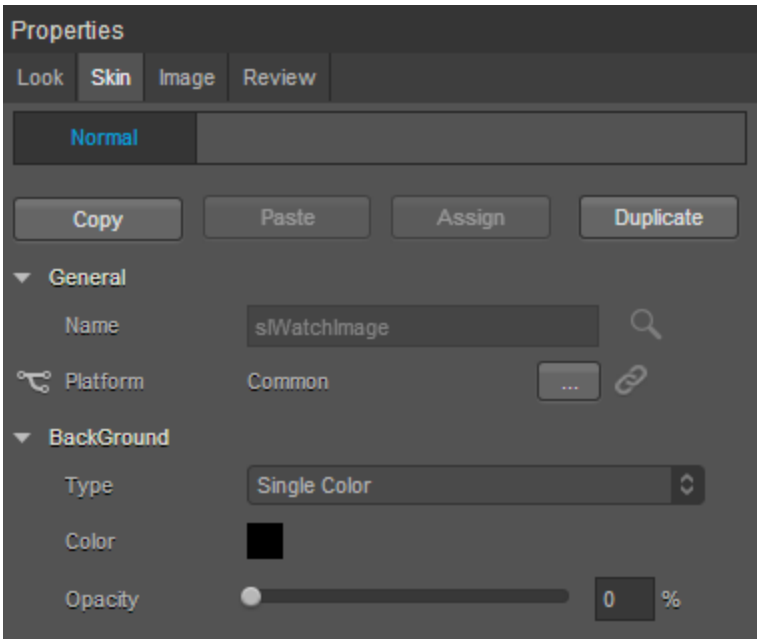
Height determines the height of the widget as measured along the y-axis (height of the parent). You can use any of the following options:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background.

In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one state: Normal.



General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available Image2 skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Background

Under the Background section of the **Skin** tab, you can set the type of background you want to use, and set the color and its opacity.

Type

For the Watch channel, the Group widget is capable of two types of backgrounds:

Background Type	Description
Single Color	Applies a uniform, single color as the background of the skin that you choose.
Image	Applies an image of your choosing as the background of the skin. The image stretches to fill the dimensions of whatever widget the skin is applied to.

Color

If you select Single Color as the background type, you can configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing.

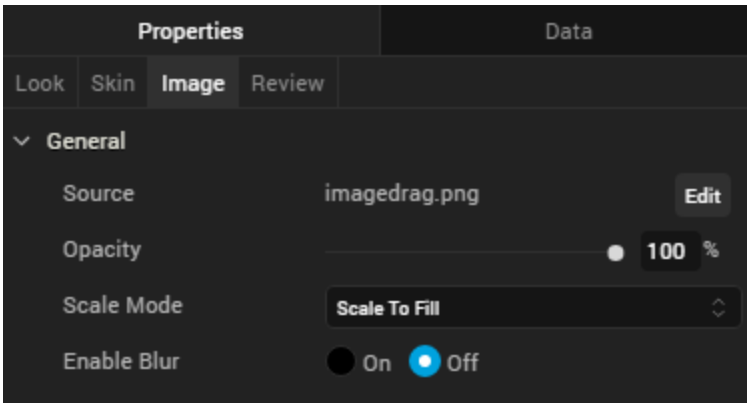
Opacity

Similarly, with Single Color as the background type, you can configure the opacity of the background color. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

For more information, see Understanding Skins and Themes.

Image Tab

On this tab, you configure properties unique to the Image2 widget.



Source

The Source property specifies the file name of the image to be displayed. When you click the **Edit** button, the Source dialog box lists those images located in the media folder. Different platforms have their own media folder; the folder that's opened is set on the **Skin** tab using the Platform property. By default, the platform is Common; images in the Common media folder are accessible to all platforms. If you select a specific platform, the images listed include those from that platform's media folder, and also from the Common media folder.

Opacity

The Opacity property sets the degree to which the image is transparent or opaque. By default, the opacity is set to 100, making the image completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Scale Mode

The Scale Mode property determines how the image's width, height, aspect ratio, and alignment are set when the dimensions and aspect ratio of the source image varies from the dimensions of the Image2 widget itself. The available values are as follows:

Scale to Fit. Fills the entire area of the Image2 widget with the image without regard for maintaining the aspect ratio.

Aspect Fit. Scales the image to fit the width of the Image2 widget while maintaining the image's aspect ratio, resulting in unused space in the widget above and below the image itself.

Aspect Fill. Fills the entire area of the Image2 widget while maintaining the image's aspect ratio

Redraw. Refreshes the image such that it fills the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Center. Displays the center of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Top. Displays the center top edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Bottom. Displays the center bottom edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Left. Displays the center left edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Right. Displays the center right edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Top Left. Displays the top left edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Top Right. Displays the top right edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Bottom Left. Displays the bottom left edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Bottom Right. Displays the bottom right edge of the image, filling the entire area of the Image2 widget without regard for maintaining the aspect ratio.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Label Widget for Watch

The Label widget displays non-editable text on a form and is non-interactive. If the text in the Label widget is occupying more space than the allocated height of the widget, the widget is stretched vertically to accommodate the full text (infinite wrapping). It does not stretch horizontally.

Click any of the following to learn about the properties found on the tabs of the Label widget.

Look Tab

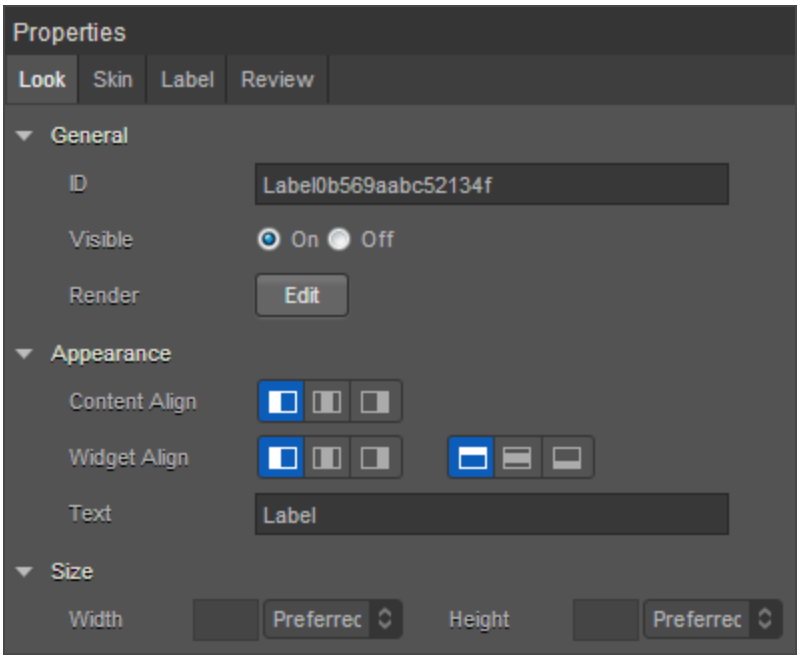
Skin Tab

Label Tab

Review Tab

Look

On the Look tab, you define properties and behaviors related to a Label widget's appearance and position. The following sections describe each of its properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** text box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

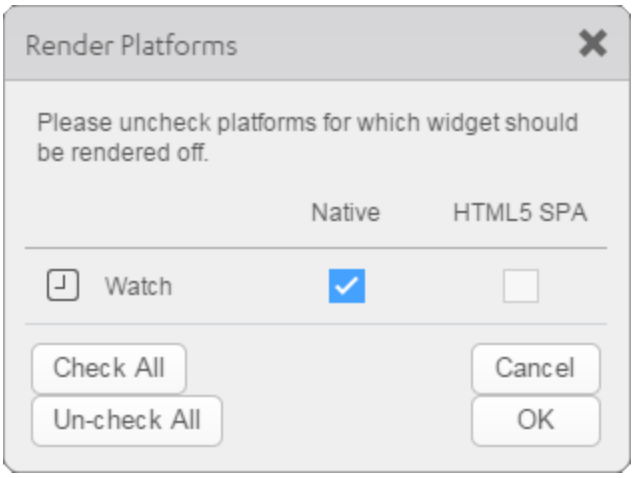
Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how a widget's boundaries are aligned with respect to its parent. The following alignment options are available:

Content Alignment

	Aligns the left edge of the content with the left edge of the widget.

	Aligns the horizontal center of the content with the horizontal center of the widget.
	Aligns the right edge of the content with the right edge of the widget.

Widget Alignment

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

Width determines the width of the widget as measured along the x-axis.

Following are the options that can be used as units of width:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Height

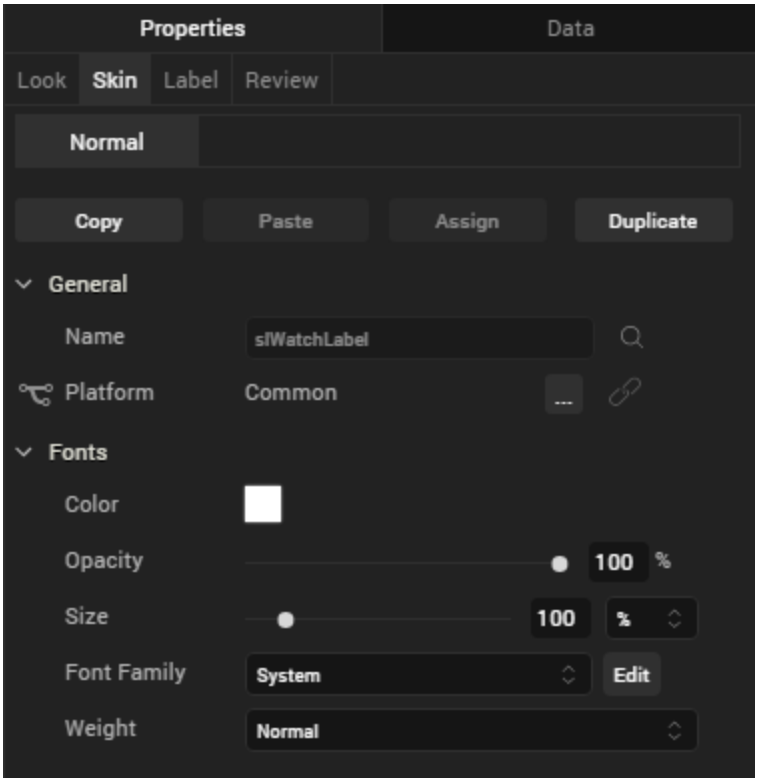
Height determines the height of the widget as measured along the y-axis (height of the parent). You can use any of the following options:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Skin

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.

In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one label state: Normal.



General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available label skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Fonts

Under the Fonts section of the **Skin** tab, you can set the following properties.

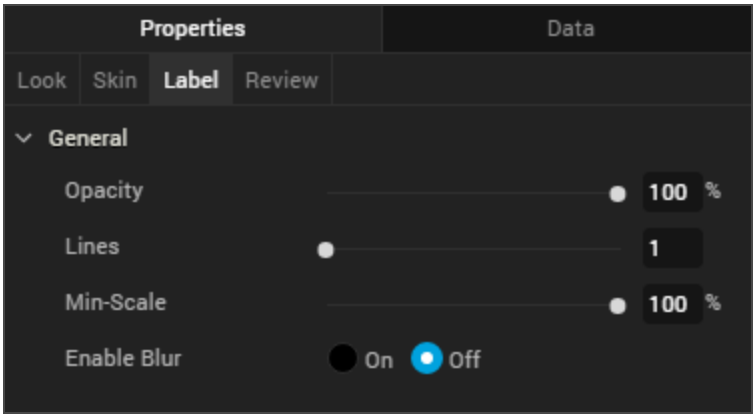
Property	Description

Color	Sets the color that you want the font to be. You configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing
Opacity	Sets the degree to which the background color is transparent or opaque. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.
Size	You can set the font size by pixels (0 to 600) or as a percentage (0 to 600) of the baseline font size of 28 pixels.
Font Family	Currently, the System font is the only font family supported by Kony Visualizer for the Label widget.
Weight	You can set the weight of the font either to Normal, which is the default, or Bold.

For more information, see Understanding Skins and Themes.

Label Tab

On this tab, you configure properties unique to the Label widget.



Opacity. Sets the degree to which the widget is transparent or opaque. By default, the opacity is set to 100, making the widget completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Lines. Sets the maximum number of lines allowed for the label text. Text that does not fit on the specified number of lines is truncated.

Min-Scale. Sets the amount by which the font may be scaled to accommodate text. Values must be 100% or less. Specifying a value of 0 causes the Apple WatchKit to use the default scaling behavior, which allows the font to be scaled to no less than 80% of the original font size.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Line Widget for Watch

The Line widget is a non-interactive widget that provides a means of visually separating widgets from one another on the Watch screen.

Click any of the following to learn about the properties found on the tabs of the Line widget.

Look Tab

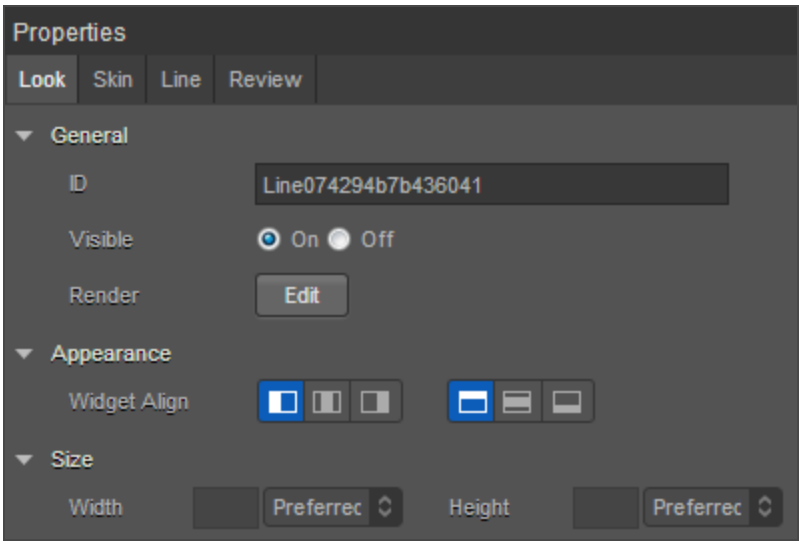
Skin Tab

Line Tab

Review Tab

Look

On the **Look** tab, you define properties and behaviors related to a Line widget's appearance and position. The following sections describe each of its properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** text box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

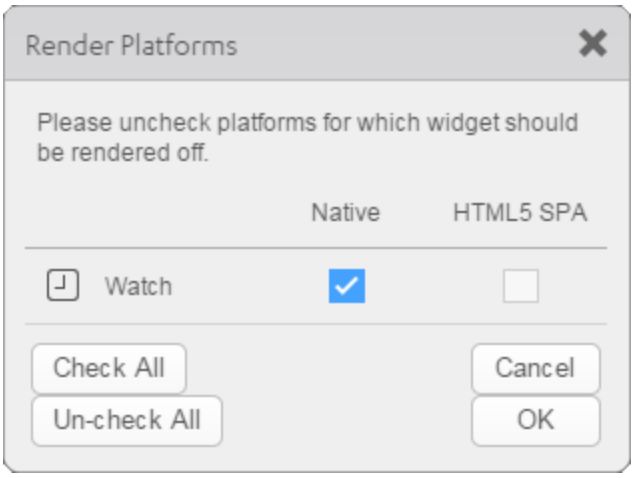
Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how a widget's boundaries are aligned with respect to its parent. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

Width determines the width of the widget as measured along the x-axis.

Following are the options that can be used as units of width:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Height

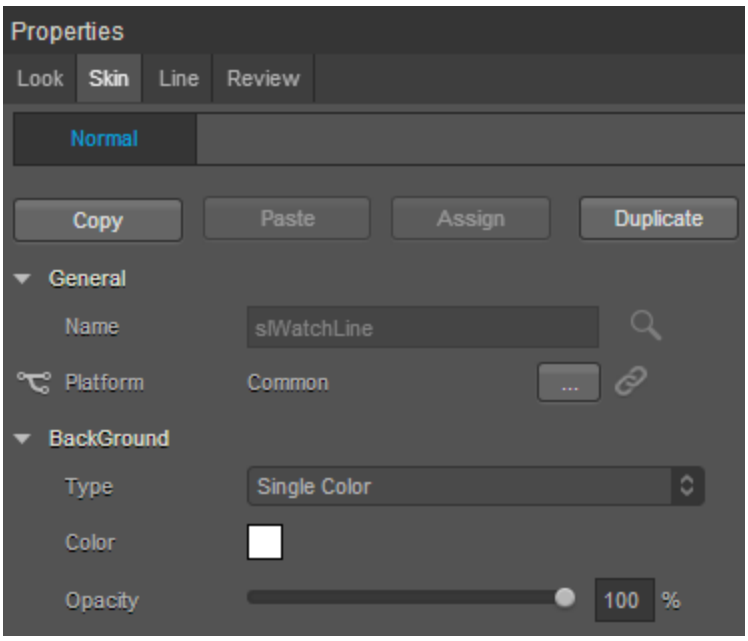
Height determines the height of the widget as measured along the y-axis (height of the parent). You can use any of the following options:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Skin

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.

In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one state: Normal.



General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available line skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Background

Under the Background section of the **Skin** tab, you can set the type of background you want to use, and set the color and its opacity.

Type

For the Watch channel, the Line widget is capable of a Single Color background.

Color

With Single Color as the background type, you can configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing.

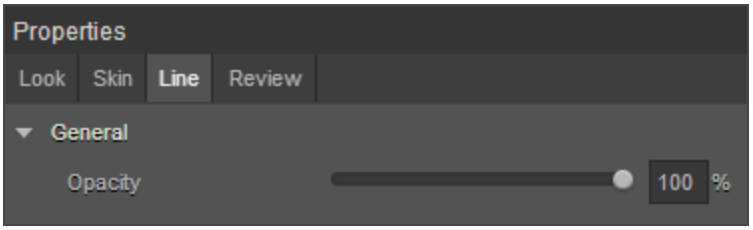
Opacity

Similarly, with Single Color as the background type, you can configure the opacity of the background color. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

For more information about applying skins, see Understanding Skins and Themes.

Line Tab

On this tab, you configure properties unique to the Line widget. Currently, there is only one.



Opacity. Sets the degree to which the widget is transparent or opaque. By default, the opacity is set to 100, making the widget completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Slider Widget for Watch

With the Slider widget, you can select a value from a defined range of values by moving the thumb (the control that you slide) in a horizontal direction along the seek bar, which is also known as the track. Optionally, you can display a minimum value and a maximum value. By adding an action sequence, the value or process can update continuously, appearing on the track when the thumb is moved.

The Slider widget

Unlike other platforms, the Apple Watch does not display the minimum and maximum values programmatically at either end of the slider, but you can compensate for this by using the Min Image and the Max Image properties to indicate the values. By default, these two values are represented by a minus symbol image and a plus symbol image.

Click any of the following to learn about the properties found on the tabs of the Slider widget.

Look Tab

Skin Tab

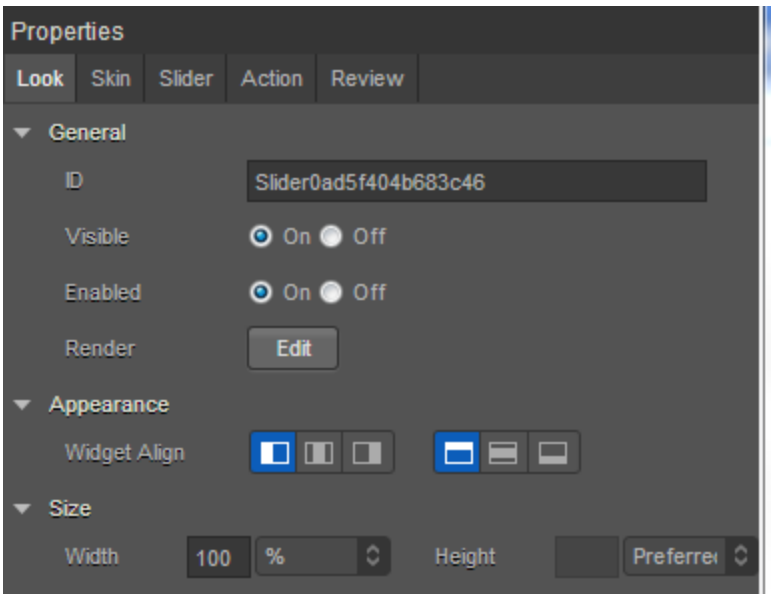
Slider Tab

Action Tab

Review Tab

Look Tab

On the **Look** tab, you define properties and behaviors related to a Slider widget's appearance and position. The following sections describe each of the properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On** .
- To make a widget invisible, click **Off**.

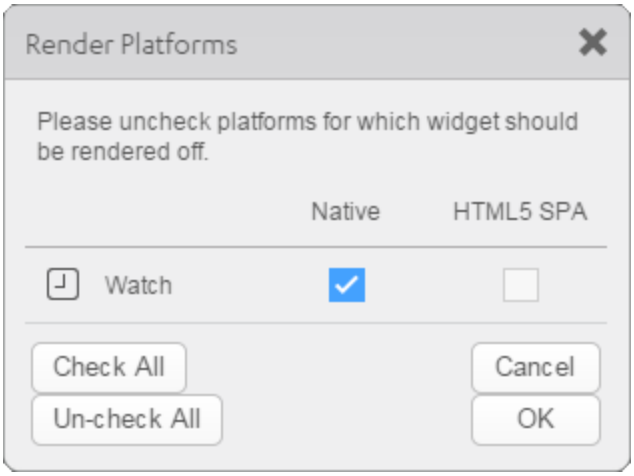
Enabled

Controls whether the widget is functional or not. You can programmatically make the widget functional or nonfunctional through an action sequence, triggered by the user.

- To enable the widget, click **On**.
- To disable the widget, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how the edge of the widget is aligned with respect to its parent's edge. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.

	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Text

Specifies the text that the user sees when running the app.

Width

The Width property sets the x-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the width of the widget.

%. Specifies the width as a percentage of the parent's dimensions.

Dp. Specifies the width in terms of device independent pixels.

Preferred Size. Specifies an optimal width for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Height

The Width property sets the y-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the height of the widget.

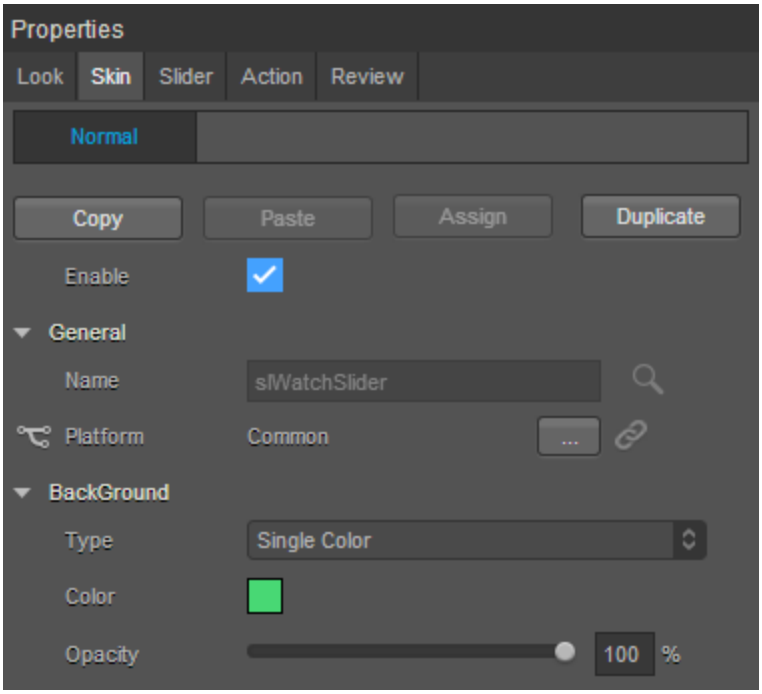
%. Specifies the height as a percentage of the parent's dimensions.

Dp. Specifies the height in terms of device independent pixels.

Preferred Size. Specifies an optimal height for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.



In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one slider state: Normal.

Enable

Depending on the current task at hand on the Watch screen, it may make sense for the Slider widget to be disabled, even as it's visible. A particular action taken by the user can then trigger the Slider widget to become enabled. Depending on the needs of your Watch app, you can set the Slider app to be initially enabled or not. By default, the **Enabled** checkbox is checked.

General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available slider skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Background

Under the Background section of the **Skin** tab, you can set the type of background you want to use, and set the color and its opacity.

Type

For the Watch channel, the Slider widget is capable of a Single Color background.

Color

With Single Color as the background type, you can configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing.

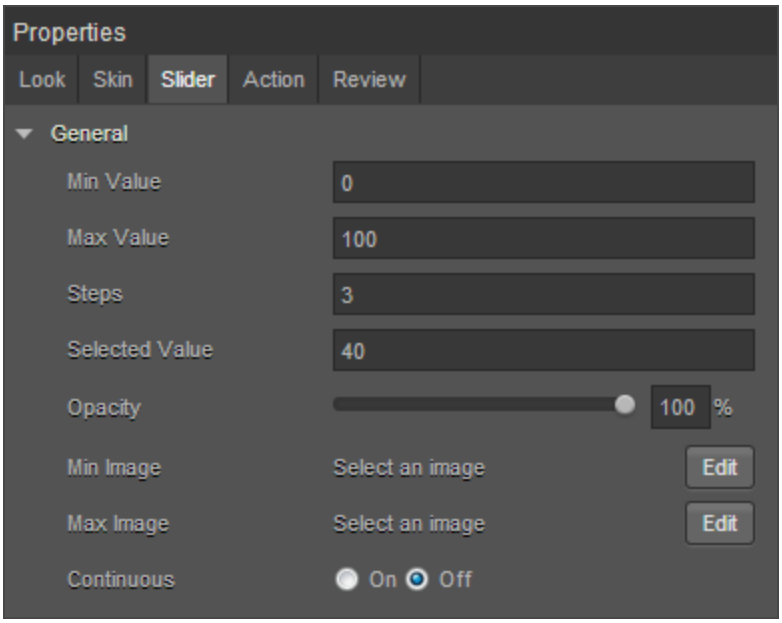
Opacity

Similarly, with Single Color as the background type, you can configure the opacity of the background color. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

For more information about applying skins, see Understanding Skins and Themes.

Slider Tab

On this tab, you configure properties unique to the Slider widget.



Min Value

The Min Value property specifies the minimum value on the Slider widget that you can select. The default value is 0 (zero). You can set any positive or negative number you want, up to 4 bytes in length.

Max Value

The Max Value property specifies the minimum value on the Slider widget that you can select. The default value is 100. You can set any positive or negative number you want, up to 4 bytes in length.

Step

The Step property specifies the number of increments by which the slider value is increased or decreased between the minimum and maximum slider values you've set. You can specify a number between 1 and 5. The default value is 3. If you enter a number that is not between 1 and 5, Kony Visualizer uses the default value of 3.

Selected Value

The Selected Value property specifies the value representing the position of the thumb along the slider when the widget is initially presented to the user. You can either set this value yourself, or let the Kony Visualizer determine an initial value for you. In the absence of you specifically setting the Selected Value property, Kony Visualizer calculates it as the Min Value plus half the difference between the Max Value and the Min Value (Selected Value = Min Value + (Max Value - Min Value/2)).

In the absence of you setting any values for Min Value, Max Value, and Selected Value, the default setting for the Selected Value property is 40.

If you set the Selected Value property to a number lower than the Min Value number, Kony Visualizer will calculate the Selected Value to be equal to the Min Value. If you set the Selected Value property to a number higher than the Max Value number, Kony Visualizer will calculate the Selected Value to be equal to the Max Value.

Min Image

Specific to the iOS platform, the Min Image property specifies the image for the minimum value of the slider. If you do not specify an image, Kony Visualizer uses a minus symbol. The aspect ratio for this image is 1:1, and any image that does not have this ratio will be stretched and scrunched to the 1:1 ratio.

To set the Min Image property, click its **Edit** button. From the Min Image dialog box, you can either select one of the listed images, or provide a URL. The listed images are located in the following folders:

```
<Workspace>\<ProjectName>\resources\common
<Workspace>\<ProjectName>\resources\watch\common
<Workspace>\<ProjectName>\resources\watch\native\watchos
```

For more information, see Add and Manage Images and Other Media.

Max Image

Specific to the iOS platform, the Max Image property specifies the image for the maximum value of the slider. If you do not specify an image, Kony Visualizer uses a plus symbol. The aspect ratio for this image is 1:1, and any image that does not have this ratio will be stretched and scrunched to the 1:1 ratio.

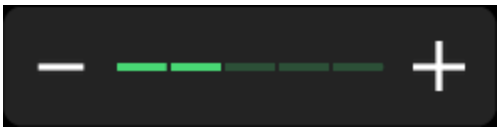
To set the Max Image property, click its **Edit** button. From the Max Image dialog box, you can either select one of the listed images, or provide a URL. The listed images are located in the following folders:

```
<Workspace>\<ProjectName>\resources\common
<Workspace>\<ProjectName>\resources\watch\common
<Workspace>\<ProjectName>\resources\watch\native\watchos
```

For more information, see Add and Manage Images and Other Media.

Continuous

The Continuous property specifies whether the slider displays the seek bar as segmented or not. The number of segments is set by the Step property. By default, the Continuous property is set to **Off**, displaying both the progress-made portion and progress-remaining portion of the seek bar as segmented. When the Continuous property is set to **On**, the progress-made portion has a very subtle segmentation, while the progress-remaining portion isn't segmented at all.



Continuous property set to Off

Continuous property set to On

Action Tab

On the **Action** tab, you define the events that are executed when an action is run. On a Slider widget, you can run the following action:

- **onValueChange.** This action is invoked when the progress on the Slider widget changes.

For more information on using this action, see the topic, Add Actions.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Timer Widget for Watch

The Timer widget displays a non-editable countdown on a form and is non-interactive.

Click any of the following to learn about the properties found on the tabs of the Timer widget.

Look Tab

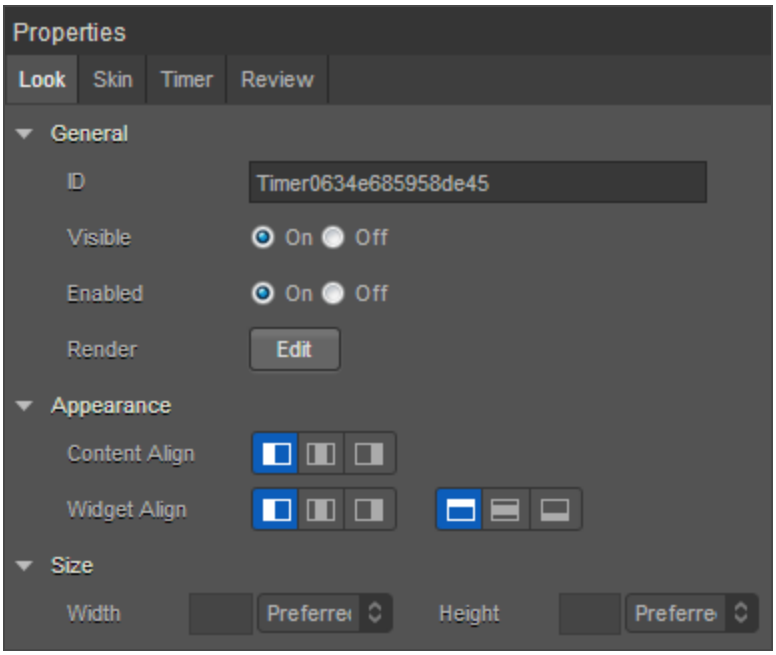
Skin Tab

Timer Tab

Review Tab

Look Tab

On the **Look** tab, you define properties and behaviors related to a Timer widget's appearance and position. The following sections describe each of its properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** text box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

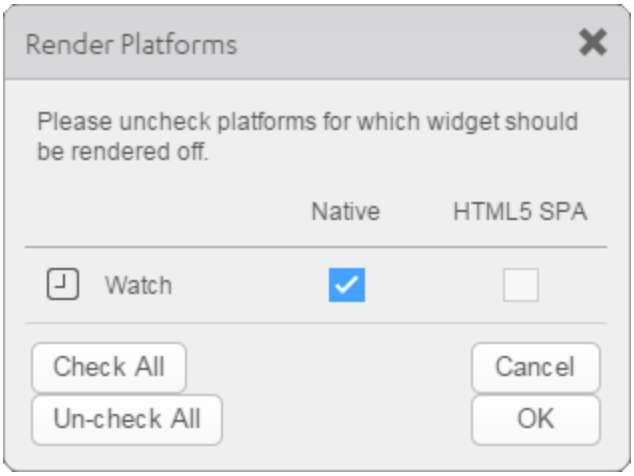
Enabled

Controls whether the widget is functional or not. You can programmatically make the widget functional or nonfunctional through an action sequence, triggered by the user.

- To enable the widget, click **On**.
- To disable the widget, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how a widget's boundaries are aligned with respect to its parent. The following alignment options are available:

Content Alignment

	Aligns the left edge of the content with the left edge of the widget.
	Aligns the horizontal center of the content with the horizontal center of the widget.
	Aligns the right edge of the content with the right edge of the widget.

Widget Alignment

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

Width determines the width of the widget as measured along the x-axis.

Following are the options that can be used as units of width:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Height

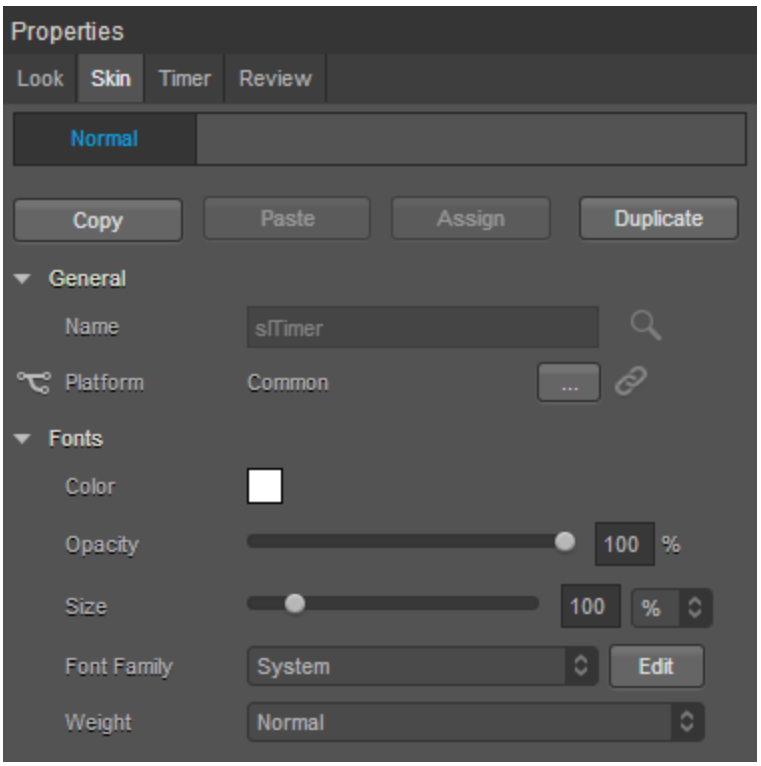
Height determines the height of the widget as measured along the y-axis (height of the parent). You can use any of the following options:

- **%.** Specifies the values in percentage relative to the parent dimensions.
- **Dp.** Specifies the values in terms of device independent pixels.
- **Preferred.** When this option is specified, the layout uses preferred height of the widget as height and preferred size of the widget is determined by the widget and may varies between platforms.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.

In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one Timer state: Normal.



General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available timer skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Fonts

Under the Fonts section of the **Skin** tab, you can set the following properties.

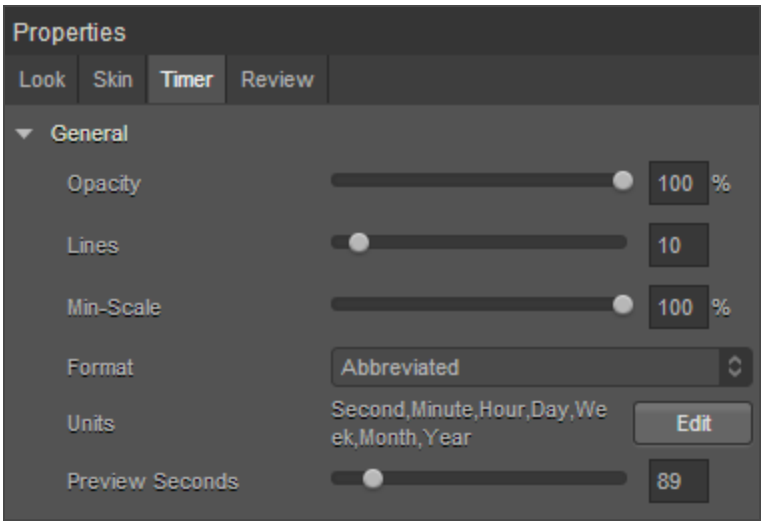
Property	Description

Color	Sets the color that you want the font to be. You configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing
Opacity	Sets the degree to which the background color is transparent or opaque. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.
Size	You can set the font size by pixels (0 to 600) or as a percentage (0 to 600) of the baseline font size of 28 pixels.
Font Family	Sets the font that you want the timer display to use. You can select one of the following fonts for the Timer widget: Arial Helvetica HelveticaNeue System System-Italic
Weight	You can set the weight of the font either to Normal, which is the default, or Bold.

For more information, see Understanding Skins and Themes.

Timer Tab

On this tab, you configure properties unique to the Timer widget.



Opacity. Sets the degree to which the widget is transparent or opaque. By default, the opacity is set to 100, making the widget completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Lines. Sets the maximum number of lines allowed for the timer text. Text that does not fit on the specified number of lines is truncated.

Min-Scale. Sets the amount by which the font may be scaled to accommodate text. Values must be 100% or less. Specifying a value of 0 causes the Apple WatchKit to use the default scaling behavior, which allows the font to be scaled to no less than 80% of the original font size.

Format. Working in conjunction with the Units property and Preview Seconds property, the Format property sets the manner in which the timer's numbers and text are displayed. The following table outlines the available formats.

Format Type	Description	Example
Positional	Displays from largest to smallest, the various units that you have set the timer to display using the Units property. The example displays the Positional format for a timer that includes hours, minutes, and seconds, with the Preview Seconds set to 200 (equal to 3 minutes and 20 seconds).	0:03:20
Abbreviated	Displays with one-letter unit descriptors, in as few units as possible using conventional time formatting, the amount of time you have set the Preview Seconds property to display, using the time units selected in the Units property. The example displays the Abbreviated format for a timer that includes hours, minutes, and seconds, with the Preview Seconds set to 200 (equal to 3 minutes and 20 seconds).	3m 20s
Short	Displays with three or four-letter unit descriptors, in as few units as possible using conventional time formatting, the amount of time you have set the Preview Seconds property to display, using the time units selected in the Units property. The example displays the Short format for a timer that includes hours, minutes, and seconds, with the Preview Seconds set to 200 (equal to 3 minutes and 20 seconds).	3min 20secs

Full	Displays with full-word unit descriptors, in as few units as possible using conventional time formatting, the amount of time you have set the Preview Seconds property to display, using the time units selected in the Units property. The example displays the Full format for a timer that includes hours, minutes, and seconds, with the Preview Seconds set to 200 (equal to 3 minutes and 20 seconds).	3 minutes 20 seconds
Spelled Out	Displays with words only (no numbers), in as few units as possible using conventional time formatting, the amount of time you have set the Preview Seconds property to display, using the time units selected in the Units property. The example displays the Spelled Out format for a timer that includes hours, minutes, and seconds, with the Preview Seconds set to 200 (equal to 3 minutes and 20 seconds).	three minutes twenty seconds

Units. Used for selecting the units of time that you want the timer to display. To change the units, click the property's **Edit** button. The available units are as follows:
Year, Month, Week, Day, Hour, Minute, Second

Preview Seconds. Used for entering a placeholder value to see how the timer is rendered in the Timer widget on the Visualizer Canvas. You can set this property to a value between 0 and 600 seconds (10 minutes). The default is 600.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Map Widget for Watch

A Map widget provides you the capability to display pre-defined locations (latitude and longitude) on an onscreen map. The iOS platform (above 3.0) provides a native map widget that can be displayed as part of the application.

The Map widget for Watch renders maps using Google Maps.

Click any of the following to learn about the properties found on the tabs of the Map widget.

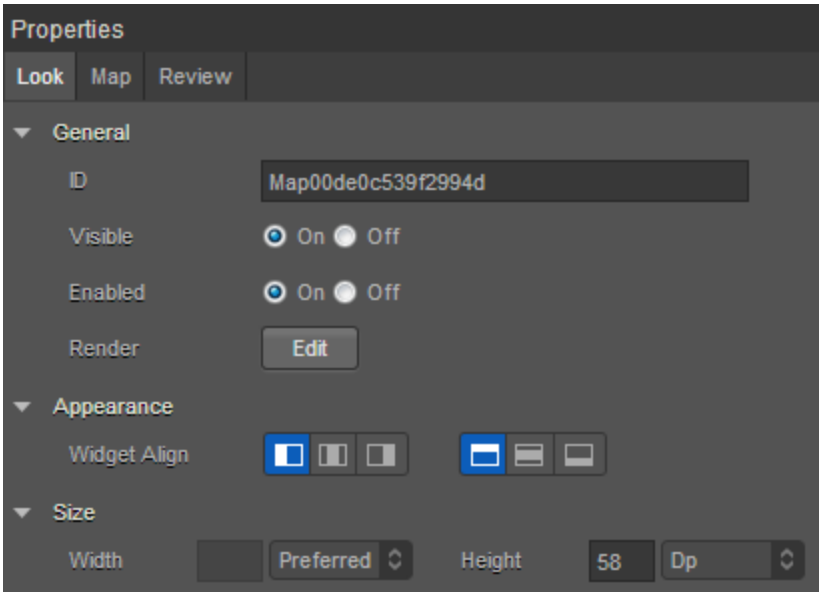
Look Tab

Map Tab

Review Tab

Look Tab

On the **Look** tab, you define properties and behaviors related to a Map widget's appearance and position. The following sections describe each of the properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On**.
- To make a widget invisible, click **Off**.

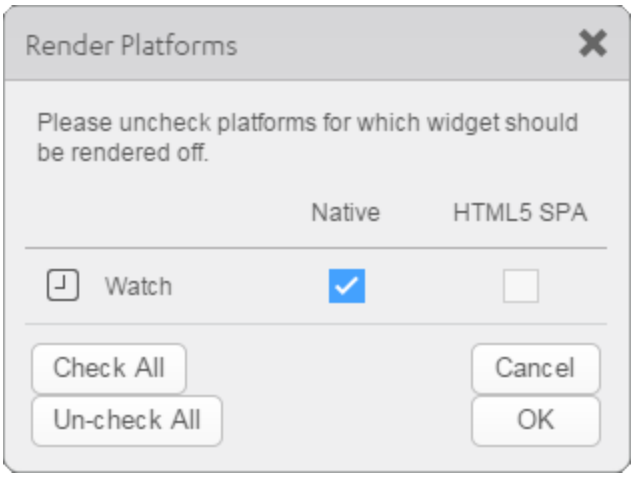
Enabled

Controls whether the widget is functional or not. You can programmatically make the widget functional or nonfunctional through an action sequence, triggered by the user.

- To enable the widget, click **On**.
- To disable the widget, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how the edges of the Map widget are aligned with respect to its parent's edges. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Width

The Width property sets the x-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the width of the widget.

%. Specifies the width as a percentage of the parent's dimensions.

Dp. Specifies the width in terms of device independent pixels.

Preferred Size. Specifies an optimal width for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Height

The Width property sets the y-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the height of the widget.

%. Specifies the height as a percentage of the parent's dimensions.

Dp. Specifies the height in terms of device independent pixels.

Preferred Size. Specifies an optimal height for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Map Tab

On this tab, you configure properties unique to the Map widget. Currently, there is only one.

Opacity. Sets the degree to which the widget is transparent or opaque. By default, the opacity is set to 100, making the widget completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Switch Widget for Watch

The Switch widget is identical to the Switch Control in iPhone (an on-off switch that is non customizable) and presents two mutually exclusive choices or states.

The Switch widget displays the value that is currently in effect. You must slide the control to select (or reveal) the other value.

Click any of the following to learn about the properties found on the tabs of the Switch widget for Watch.

Look Tab

Skin Tab

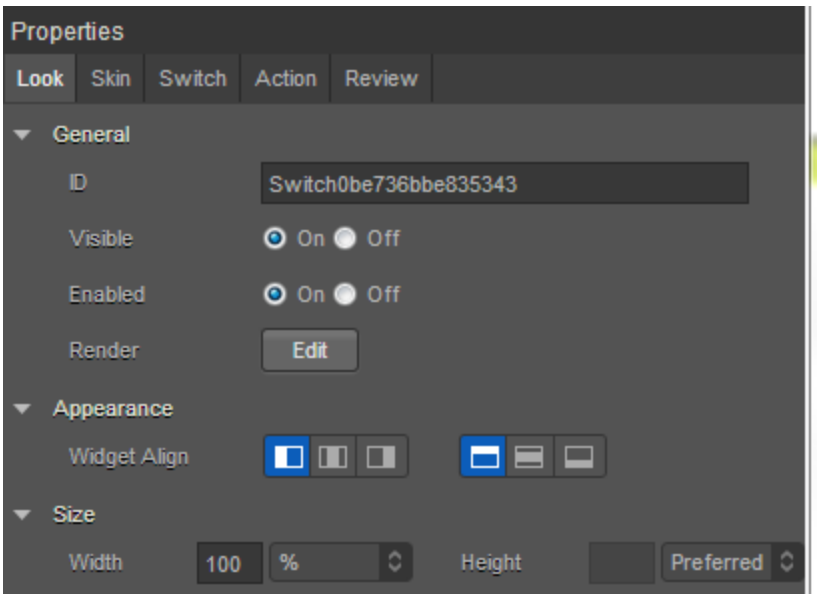
Switch Tab

Action Tab

Review Tab

Look Tab

On the **Look** tab, you define properties and behaviors related to a Switch widget's appearance and position. The following sections describe each of the properties.



ID

Denotes the name of a widget. When a widget is added to a form, a unique name is assigned to the widget. You can rename a widget by entering a new name in the **ID** box.

> ***Note:*** You can also rename a widget from the Project Explorer by right-clicking a widget, and then clicking **Rename**.

Visible

Controls whether or not the user of the app can see the widget.

- To make a widget visible, click **On** .
- To make a widget invisible, click **Off**.

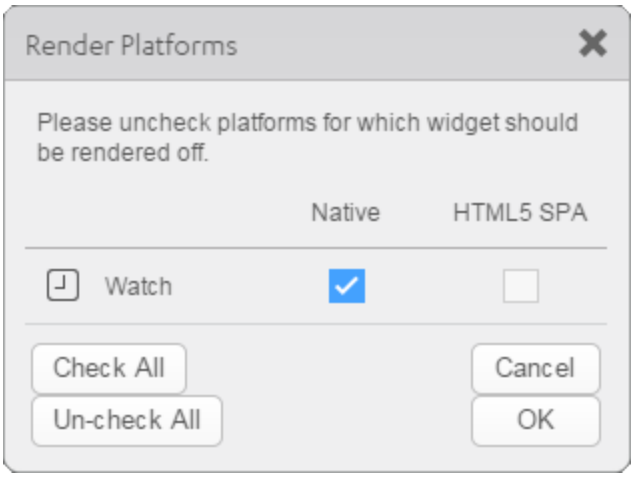
Enabled

Controls whether the widget is functional or not. You can programmatically make the widget functional or nonfunctional through an action sequence, triggered by the user.

- To enable the widget, click **On**.
- To disable the widget, click **Off**.

Render

Defines whether a widget appears on a specific platform. Currently, the Watch channel supports only the Apple Watch Native and HTML5 SPA platforms. Clicking the Render property's **Edit** button opens the **Render Platforms** dialog box.



Clear the check box of the platforms for which the widget should not be rendered.

The Difference between Visible and Render

- When a Widget is *not* rendered for a platform, it implies that the widget is hidden from that specific platform.
- Whereas, when a widget is set as invisible, it implies that the widget is available, but is invisible. This feature is useful when you wanted to display a widget based on certain conditions.

Widget Align

The Widget Align property specifies how the edge of the widget is aligned with respect to its parent's edge. The following alignment options are available:

	Aligns the left edge of the widget with the left edge of its parent.
	Aligns the horizontal center of the widget with the horizontal center of its parent.
	Aligns the right edge of the widget with the right edge of its parent.
	Aligns the top edge of the widget with the top edge of its parent.
	Aligns the vertical center of the widget with the vertical center of its parent.
	Aligns the bottom edge of the widget with the bottom edge of its parent.

Text

Specifies the text that the user sees when running the app.

Width

The Width property sets the x-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the width of the widget.

%. Specifies the width as a percentage of the parent's dimensions.

Dp. Specifies the width in terms of device independent pixels.

Preferred Size. Specifies an optimal width for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Height

The Width property sets the y-axis dimension of the widget using the numeric quantity and type of unit that you specify.

You can use the following units of measure to set the height of the widget.

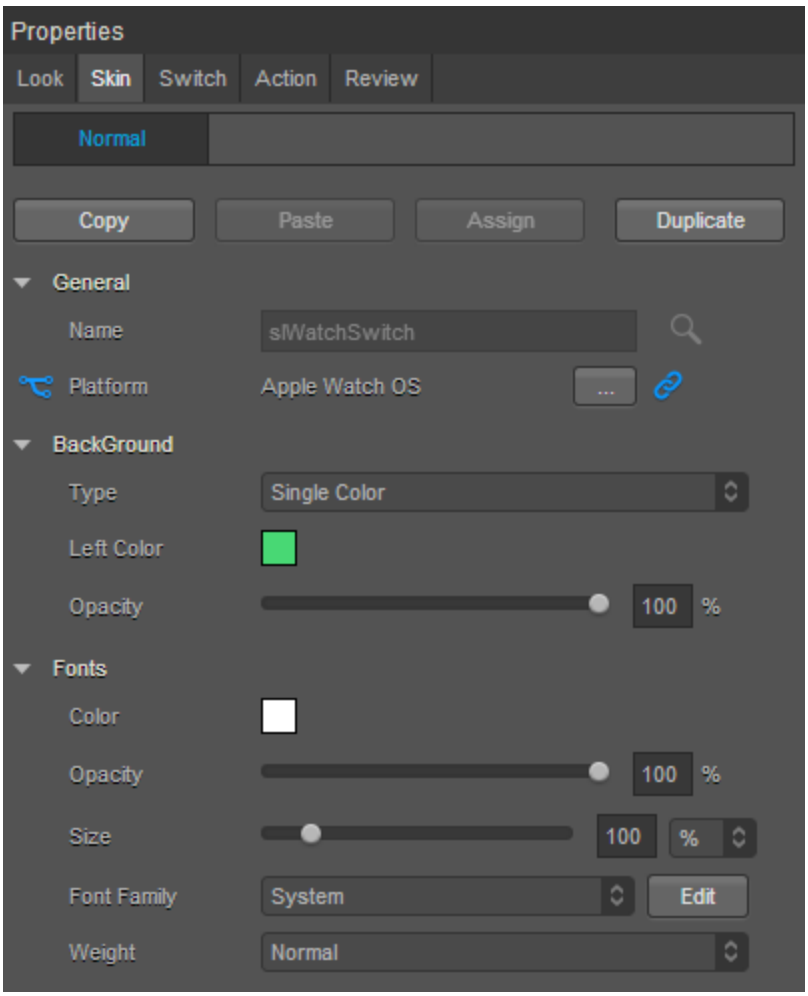
%. Specifies the height as a percentage of the parent's dimensions.

Dp. Specifies the height in terms of device independent pixels.

Preferred Size. Specifies an optimal height for the widget given the size of the font and the amount of text that the widget displays. The Preferred Size can vary from one platform to another.

Skin Tab

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. On the **Skin** tab, you can select to use a specific skin for your widget. In addition, you can configure the widget's background and font.



In the Mobile, Tablet, and Desktop channels, a widget may have a number of states, such as Normal (when it's not being interacted with), Focus (e.g. when it's been tabbed to), or Pressed. However, the Watch channel has only one state: Normal.

General

Under the General section of the **Skin** tab, you can change the name of the skin currently applied (if it's not one of the default skins), or you can select from the other available switch skins by clicking the magnifying glass icon next to the **Name** text box.

Platform

In channels that support multiple platforms, it's possible to fork a skin by clicking the Platform ellipsis button  , and then selecting the platforms that you want to fork the widget to. In the case of the Watch channel, currently the only platform available is Watch (Native). For more information, see Forking.

Background

Under the Background section of the **Skin** tab, you can set the type of background you want to use, and set the color and its opacity.

Type

For the Watch channel, the Switch widget is capable of two types of backgrounds:

Background Type	Description
Single Color	Applies a uniform, single color as the background of the skin that you choose.
Image	Applies an image of your choosing as the background of the skin. The image stretches to fill the dimensions of whatever widget the skin is applied to.

Color

If you select Single Color as the background type, you can configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing.

Opacity

Similarly, with Single Color as the background type, you can configure the opacity of the background color. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

Fonts

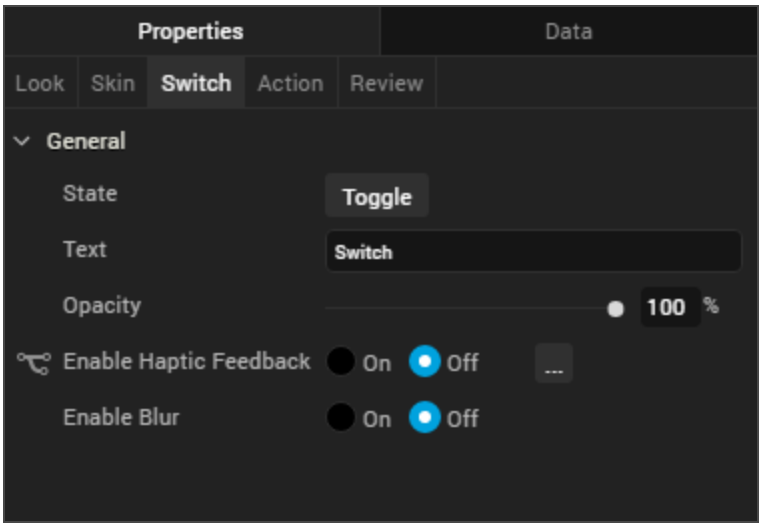
Under the Fonts section of the **Skin** tab, you can set the following properties.

Property	Description
Color	Sets the color that you want the font to be. You configure the hue you want by clicking the square color icon and dragging the cursor to the color of your choosing
Opacity	Sets the degree to which the background color is transparent or opaque. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.
Size	You can set the font size by pixels (0 to 600) or as a percentage (0 to 600) of the baseline font size of 28 pixels.
Font Family	Currently, the System font is the only font family supported by Kony Visualizer for the Label widget.
Weight	You can set the weight of the font either to Normal, which is the default, or Bold.

For more information, see Understanding Skins and Themes.

Switch Tab

On this tab, you can assign properties that are run on any platform (which is supported by Kony Visualizer) and also, assign properties specific to a platform.



State

Allows you to swap the colors.

Text

Specifies the text that the user sees to identify the switch.

Opacity

The Opacity property sets the degree to which the switch is transparent or opaque. By default, the opacity is set to 100, making the switch completely opaque with no transparency. However, if you want it to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the **Opacity** text box, or drag the opacity slider to the degree of opacity that you want.

Action Tab

On this tab, you define the events that are executed when an action is run. On a Switch widget, you can run the following action:

- **onStateChange.** This action is invoked when the state of the Switch widget changes from enabled to disabled, or vice versa.

For more information on using the above action, see the topic, Add Actions.

Review Tab

On this tab, you can add and review notes. With the Review Notes feature, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables.

For more information, see Capture Product Requirements with Review Notes.

Using the Native UI Container

Applies to *Kony Visualizer Classic*.

In Kony Visualizer, you can use native UI widgets in addition to the existing Kony cross-platform widgets. These native UI widgets are platform-specific widgets to either iOS or Android, and can be used in an application by placing them in a native UI container widget called a NativeContainer. A native UI container can only be used for native UI widgets; it cannot contain any Kony cross-platform widgets.

You can place a NativeContainer widget in a FlexContainer, FlexScrollContainer, component, or master. However, you cannot add it to the Tab and TabPane container widgets, nor can you add it to templates. Even a component or master containing a NativeContainer widget cannot be placed in a template.

Once it's added, to be able to see a NativeContainer widget listed in the widget library, you have to select that platform's canvas on the Visualizer Canvas. For example, the iOS NativeContainer will not display in the widget library if the platform of choice on the Visualizer Canvas is *Android: Native*. You'd need to select *Apple-iOS: Native*.

To add a NativeContainer or native UI widget to an app, you first must download it and add it to Kony Visualizer. The following procedure describes how to do so, along with how to activate the NativeContainer and native UI widget once it's downloaded. For more information about SSMs and NativeContainers, see the *Kony Visualizer API Developer's Guide*.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Download and Activate the Assets for the Native Container

To download and activate the native function UI API assets for NativeContainer widgets, do the following:

1. Download the Native Function UI API you want from the Kony install site. They are located at the bottom of the page under the heading, *Other Downloads*.
2. On the **Project** menu of Kony Visualizer, click **Add JS Modules**. The **Add JS Modules** dialog box opens.
3. In the **Add JS Module** dialog box, click **Browse**, and then navigate and select an native function UI API zip file that you downloaded.

4. Click **Finish**. A message box appears indicating that the API was successfully added to your app. Click **OK**.
5. Repeat steps 2 through 4 for each API that you want to add.
6. Click the **Project Settings** icon. The **Project Settings** window opens with the **Self Sufficient Module** tab added. Click the **Self Sufficient Module** tab. The tab opens with the name of the native function API that you imported, displayed with a check box under the platform to which it is applicable.
7. Select the check boxes of the native function APIs that you want to use as widgets with the native UI container, and then click **Finish**.
8. On the Visualizer Canvas, select the platform of the SSM that you just added to Kony Visualizer.
9. In the widget library, scroll to the bottom. You will see a new category of widget called *NativeContainer*, and listed under it is the widget itself. You can now drag the native UI container onto your app and add the imported native UI widgets to it.

Add or Import Native Function APIs for the Native Container

To add native function APIs for NativeContainer widgets, do the following:

1. On the **Edit** menu, click **Add Native Function API**. The Native Function Interface dialog box displays, and lists any native APIs you have already added or imported into the project.
2. Click the platform you want to download an API for, either iOS or Android.
3. Click **Add**. Kony Visualizer retrieves a list of native APIs available for the platform you chose.
4. In the Search text box, type keywords that help narrow the scope of the APIs listed.
5. From the list, select an API you want, and then click **Add**.
6. Repeat steps 3 through 5 until you have added all the native APIs you want.

7. When you are finished adding native APIs, close the dialog box by clicking the **X** in the upper right corner.
8. On the Visualizer Canvas, select the platform that corresponds to the NativeContainer widget you want to add.
9. In the widget library, scroll to the bottom. You will see a new category of widget called *NativeContainer*, and listed under it is the widget itself. You can now drag the native UI container onto your app and associate the added native APIs to it.

To import native function APIs for NativeContainer widgets, do the following:

1. On the **Edit** menu, click **Add Native Function API**. The Native Function Interface dialog box displays, and lists any native APIs you have already added or imported into the project.
2. Click the platform you want to download an API for, either iOS or Android.
3. Click **Import**.
4. Navigate to the location of the native API, select it, and then click **OK**. The API is added to the list of native APIs associated with your project.
5. Repeat steps 3 and 4 until you have imported all the native APIs you want.
6. When you are finished importing native APIs, close the dialog box by clicking the **X** in the upper right corner.
7. On the Visualizer Canvas, select the platform that corresponds to the NativeContainer widget you want to add.
8. In the widget library, scroll to the bottom. You will see a new category of widget called *NativeContainer*, and listed under it is the widget itself. You can now drag the native UI container onto your app and associate the imported native APIs to it.

Add and Use Third-Party and Custom Widgets

Kony Visualizer comes with a broad variety of native widgets. But you might see a widget elsewhere not available in Kony Visualizer that you want to incorporate into your Kony app. With Kony Visualizer, you can import such third-party and custom widgets from external libraries and frameworks. The process for doing so varies by platform. For instructions on how to import and use third-party and custom widgets, choose from the following platforms:

iOS

Android

Windows

SPA and Desktop Web

For a more hands-on approach on how to use Custom Widgets, import and preview the Rating Panel using Custom Widget app by using Kony Visualizer.

Importing Custom Widgets for iOS

You can import, create, and integrate third party custom widgets for the iOS platform. To help you do so, Kony Visualizer defines the following:

- A custom widget protocol. Kony Platform defines a custom widget protocol called CWIWidget. The widget protocol consists of mandatory methods that help you initialize, render, define properties, or update properties, for your widget. As part of custom widget implementation, a class must confirm to this protocol, so that the platform can communicate with the custom widget for initialization.
- A Kony Environment class. The Kony environment class consists of a property model, which enables you to fetch data from the custom widget protocol and to send data between the application and custom widget protocol.

To successfully define and import an iOS custom widget into Kony Visualizer, you must do the following:

1. Create a custom widget class implementation (mandatory). This implementation communicates with Kony Visualizer's custom widget protocol *CWIWidget*.
2. Create a protocol for the Kony Custom Event Delegate (optional). This protocol consists of events to be invoked for the custom widget.
3. Import the widget.

Custom Widget Protocol for iOS

The following comprise the widget protocol:

- **(id) initWithEventDelegate: (id) eventDelegate withKonyEnvironment:(id) env**: Initializes any object of the custom widget set. Initializing the Kony Environment and Custom Widget instance.
- **(CGSize)getPreferredSizeForGivenSize:(CGSize)givenSize**: Fetch the preferred size (Height and width) for the widget. The CGSize parameter is the parent width and height of the widget.
- **(void)setWidgetViewFrame:(CGRect)frame**: Set the position of the custom widget with respect to the parent widget. Provides x, y coordinates, height, and width for the custom widget with reference to the parent widget.
- **(UIView*) getWidgetView**: Returns the view defined for the custom widget. This view will be defined in a custom widget class.
- **(void) modelUpdatedForProperty:(NSString*) propertyName withOldValue:(id) oValue newValue:(id) nValue;**: Invoked when there are any updates in the end user defined properties.
- **(void)willBeginLayout** Invoked before the widget is laid out.
- **(void)didEndLayout** :Invoked after the widget is laid out.
- **(void)didCreateWidget** Invoked on creation of a widget.
- **(void)willDestroyWidget** : Invoked when the custom widget is destroyed.

The custom widget protocol provided by Kony is as follows:

```
//
//  CWIWidget.h
//  VM
//
//  Created by Amba Babjee Dhanisetti on 19/08/12.
//  Copyright (c) 2012 Kony Solutions. All rights reserved.
//

#
import < Foundation / Foundation.h >

@protocol CWIWidget < NSObject >

@required
/***
 Internally, will be used to initialize
 ****/
- (id) initWithEventDelegate: (id) eventDelegate withKonyEnvironment:
(id) env;

// View related methods
- (UIView * ) getWidgetView;

- (CGSize) getPreferredSizeForGivenSize: (CGSize) givenSize; - (void)
setWidgetViewFrame: (CGRect) frame;


@
optional

// Model updates
- (void) modelUpdatedForProperty: (NSString * ) propertyName
```

```
withOldValue: (id) oValue newValue: (id) nValue;

// Layout related methods
- (void) willBeginLayout; - (void) didEndLayout;

// Widget lifecycle events
- (void) didCreateWidget; - (void) willDestroyWidget;

@
end


// KonyEnvironment
/***
1. Forcing the Layout
    -(void) forceLayout;

2. Model exposing - Key Value compliant
(valueForKey:,setValue:forKey:) - to access properties in model
    KonyEnvironmentInstace.model - access the mapped properties

3. model class : save/get private data methods
    - (void) setNonModelStateValue:(id)value forKey:(NSString *)key;
    - (id) getNonModelStateValueForKey:(NSString *)key;

***/
```

Kony Environment Protocol

Kony provides a Kony Environment protocol. The model property in the Kony Environment Protocol fetches the data from the custom widget protocol and to send across data between the application and custom widget protocol.

```
@interface KonyCWIEnvironment: NSObject
```

```
@property(nonatomic, retain) id model;

- (void) forceLayout;

@end
```

Sample Custom Widget Implementation

The following is a sample implementation for the custom widget, Range Slider. This consists of all the basic data about the Range Slider Widget.

```
//
//  RangeSlider.m
//  RangeSlider
//
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "RangeSlider.h"

@interface RangeSlider(PrivateMethods) - (float) xForValue: (float)
value; - (float) valueForX: (float) x; - (void) updateTrackHighlight;@
end

@implementation RangeSlider

@synthesize minimumValue, maximumValue, minimumRange,
selectedMinimumValue, selectedMaximumValue;

@synthesize konyEnviroment, rangeSliderDelegate;@
synthesize minThumbImageName, maxThumbImageName,
trackBackgroundImageName, trackHighlightImageName;
```

```
- (id) init {
    self = [super init];
    if (self) {
        _minThumbOn = false;
        _maxThumbOn = false;
        _padding = 20;

        _trackBackground = [
            [
                [UIImageView alloc] initWithImage: [UIImage
imageNamed: @"bar-background.png"]
            ] autorelease
        ];
        [self addSubview: _trackBackground];

        _track = [
            [
                [UIImageView alloc] initWithImage: [UIImage
imageNamed: @"bar-highlight.png"]
            ] autorelease
        ];
        [self addSubview: _track];

        _minThumb = [
            [
                [UIImageView alloc] initWithImage: [UIImage
imageNamed: @"handle.png"] highlightedImage: [UIImage imageNamed:
@"handle-hover.png"]
            ] autorelease
        ];
        _minThumb.contentMode = UIViewContentModeCenter;
        [self addSubview: _minThumb];
```

```
        _maxThumb = [
            [
                [UIImageView alloc] initWithImage: [UIImage
imageNamed: @"handle.png"] highlightedImage: [UIImage imageNamed:
@"handle-hover.png"]
            ] autorelease
        ];
        _maxThumb.contentMode = UIViewContentModeCenter;
        [self addSubview: _maxThumb];
    }

    return self;
}

- (void) updateWithFrame: (CGRect) frame {

    self.frame = frame;
    _trackBackground.frame = CGRectMake(0, 0, frame.size.width, _
trackBackground.image.size.height);
    _track.frame = CGRectMake(0, 0, frame.size.width, _
track.frame.size.height);
    _trackBackground.center = self.center;
    _track.center = self.center;
    _minThumb.frame = CGRectMake(0, 0, self.frame.size.height,
self.frame.size.height);
    _maxThumb.frame = CGRectMake(0, 0, self.frame.size.height,
self.frame.size.height);

}


- (void) layoutSubviews {
    // Set the initial state
```

```
    _minThumb.center = CGPointMake([self xForValue:
selectedMinimumValue], self.center.y);

    _maxThumb.center = CGPointMake([self xForValue:
selectedMaximumValue], self.center.y);

    [self updateTrackHighlight];

}

- (float) xForValue: (float) value {
    return ((self.frame.size.width - (_padding * 2)) * ((value -
minimumValue) / (maximumValue - minimumValue)) + _padding);
}

- (float) valueForX: (float) x {

    return minimumValue + (x - _padding) / (self.frame.size.width - (_
padding * 2)) * (maximumValue - minimumValue);
}

- (BOOL) continueTrackingWithTouch: (UITouch * ) touch withEvent:
(UIEvent * ) event {
    if (!_minThumbOn & amp; & amp; !_maxThumbOn) {
        return YES;
    }

    CGPoint touchPoint = [touch locationInView: self];
    if (_minThumbOn) {
        _minThumb.center = CGPointMake(MAX([self xForValue:
minimumValue], MIN(touchPoint.x - distanceFromCenter, [self xForValue:
selectedMaximumValue - minimumRange])), _minThumb.center.y);
        selectedMinimumValue = [self valueForX: _minThumb.center.x];
```

```
    }
    if (_maxThumbOn) {
        _maxThumb.center = CGPointMake(MIN([self xForValue:
maximumValue], MAX(touchPoint.x - distanceFromCenter, [self xForValue:
selectedMinimumValue + minimumRange])), _maxThumb.center.y);
        selectedMaximumValue = [self valueForX: _maxThumb.center.x];
    }
    [self updateTrackHighlight];
    [self setNeedsLayout];

    [self sendActionsForControlEvents: UIControlEventValueChanged];
    return YES;
}

- (BOOL) beginTrackingWithTouch: (UITouch * ) touch withEvent:
(UIEvent * ) event {
    CGPoint touchPoint = [touch locationInView: self];

    if (CGRectContainsPoint(_minThumb.frame, touchPoint)) {
        _minThumbOn = true;
        distanceFromCenter = touchPoint.x - _minThumb.center.x;
    } else if (CGRectContainsPoint(_maxThumb.frame, touchPoint)) {
        _maxThumbOn = true;
        distanceFromCenter = touchPoint.x - _maxThumb.center.x;

    }
    return YES;
}

- (void) endTrackingWithTouch: (UITouch * ) touch withEvent: (UIEvent
* ) event {
    _minThumbOn = false;
```

```
    _maxThumbOn = false;
}

- (void) updateTrackHighlight {
    _track.frame = CGRectMake(
        _minThumb.center.x,
        _track.center.y - (_track.frame.size.height / 2),
        _maxThumb.center.x - _minThumb.center.x,
        _track.frame.size.height
    );

    [self.rangeSliderDelegate updatedMinValue:
self.selectedMinimumValue andMaxValue: self.selectedMaximumValue];
}

/*
 // Only override drawRect: if you perform custom drawing.
 // An empty implementation adversely affects performance during
animation.
 - (void)drawRect:(CGRect)rect
 {
 // Drawing code
 }
 */

#
pragma mark Custom Widget Protocols Methods


- (id) initWithEventDelegate: (id) eventDelegate withKonyEnvironment:
(id) env {
    self = [self init];
```

```
    self.rangeSliderDelegate = eventDelegate;
    self.konyEnviroment = env;

    [self setDefaultValuesFromModel];
    return self;
}

// View related methods
- (UIView * ) getWidgetView {
    return self;
}

- (CGSize) getPreferredSizeForGivenSize: (CGSize) givenSize {
    CGSize preferredSize = givenSize;
    preferredSize.height = 44;
    return preferredSize;
}

- (void) setWidgetViewFrame: (CGRect) frame {
    [self updateWithFrame: frame];
}


- (void) modelUpdatedForProperty: (NSString * ) propertyName
withOldValue: (id) oValue newValue: (id) nValue {

    if ([propertyName isEqualToString: @"maximumValue"]) {

        self.maximumValue = [nValue floatValue];

    } else if ([propertyName isEqualToString: @"minimumValue"]) {

        self.minimumValue = [nValue floatValue];
```

```
    } else if ([propertyName isEqualToString: @"minThumbImageName"]) {

        self.minThumbImageName = nValue;

        [_minThumb setImage: [UIImage imageNamed:
self.minThumbImageName]];

    } else if ([propertyName isEqualToString: @"maxThumbImageName"]) {

        self.maxThumbImageName = nValue;

        [_maxThumb setImage: [UIImage imageNamed:
self.maxThumbImageName]];

    } else if ([propertyName isEqualToString:
@"trackBackgroundImageName"]) {

        self.trackBackgroundImageName = nValue;

        [_trackBackground setImage: [UIImage imageNamed:
self.trackBackgroundImageName]];

    } else if ([propertyName isEqualToString:
@"trackHighlightImageName"]) {

        self.trackHighlightImageName = nValue;
        [_track setImage: [UIImage imageNamed:
self.trackHighlightImageName]];
    }
}
```

```
// Layout related methods
- (void) willBeginLayout {
    NSLog(@"in %s", __PRETTY_FUNCTION__);
}

- (void) didEndLayout {
    NSLog(@"in %s", __PRETTY_FUNCTION__);
}

// Widget lifecycle events
- (void) didCreateWidget {
    NSLog(@"in %s", __PRETTY_FUNCTION__);
}

- (void) willDestroyWidget {

    NSLog(@"in %s", __PRETTY_FUNCTION__);
}


- (void) setDefaultValuesFromModel {

    self.minimumRange = [
        [self.konyEnviroment.model valueForKey: @"minimumRange"]
floatValue
    ];

    self.maximumValue = [
        [self.konyEnviroment.model valueForKey: @"maximumValue"]
floatValue
    ];
```

```
    self.minimumValue = [
        [self.konyEnviroment.model valueForKey: @"minimumValue"]
floatValue
    ];

    self.selectedMaximumValue = [
        [self.konyEnviroment.model valueForKey:
@"selectedMaximumValue"] floatValue
    ];

    self.selectedMinimumValue = [
        [self.konyEnviroment.model valueForKey:
@"selectedMinimumValue"] floatValue
    ];

    self.minThumbImageName = [self.konyEnviroment.model valueForKey:
@"minThumbImageName"];

    self.maxThumbImageName = [self.konyEnviroment.model valueForKey:
@"maxThumbImageName"];

    self.trackBackgroundImageName = [self.konyEnviroment.model
valueForKey: @"trackBackgroundImageName"];

    self.trackHighlightImageName = [self.konyEnviroment.model
valueForKey: @"trackHighlightImageName"];
}

- (void) dealloc {

    self.minThumbImageName = nil;
    self.maxThumbImageName = nil;
    self.trackBackgroundImageName = nil;
```

```
    self.trackHighlightImageName = nil;

    [super dealloc];
}
@end
```

Event Delegate Protocol

The Event delegate protocol consists of information about all the events to be invoked for your custom widget. The methods in this protocol are useful for raising events for the widget in the Kony platform. This ensures that a mapping exists between the methods in this *CustomWidgetEventDelegate* protocol and the events defined in Kony Visualizer. Kony Visualizer automatically generates a class that confirms to the *CustomWidgetEventDelegate* protocol, passing an instance of it in the initialization method. Thus, when an event needs to be raised, the methods in protocol *CustomWidgetEventDelegate* can be invoked on the saved instance object.

Sample Event Delegate Implementation

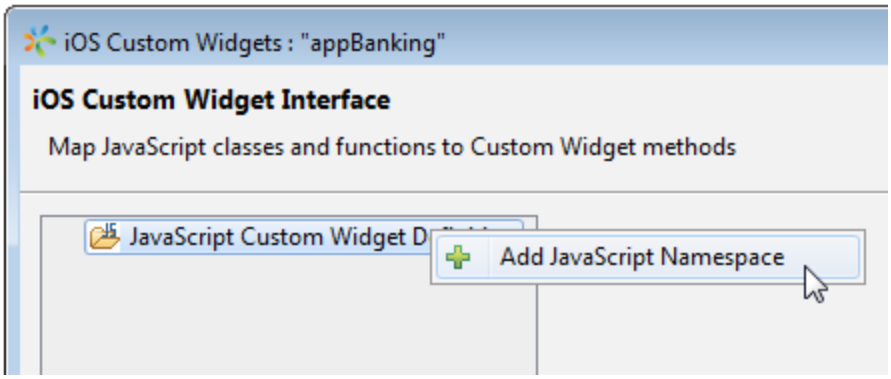
The following code snippet provides a sample Event Delegate implementation for a Range Slider.

```
//
//  RangeSliderEventDelegate.h
//  CustomRangeSlider
//  Copyright (c) 2013 Kony. All rights reserved.
//

#import <Foundation/Foundation.h>

@protocol RangeSliderEventDelegate <NSObject>

-(void)updatedMinValue:(CGFloat)minValue andMaxValue:(CGFloat)
maxValue;

@end
```

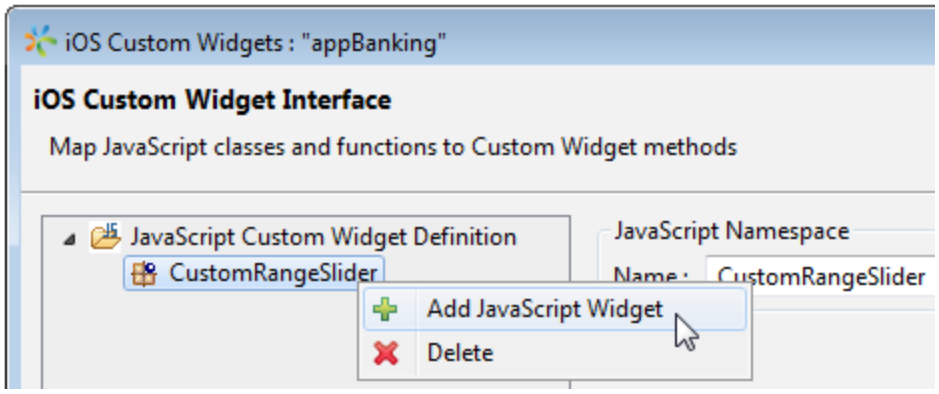
Import a Custom Widget for iOS

To import files for a third-party widget, execute the following:

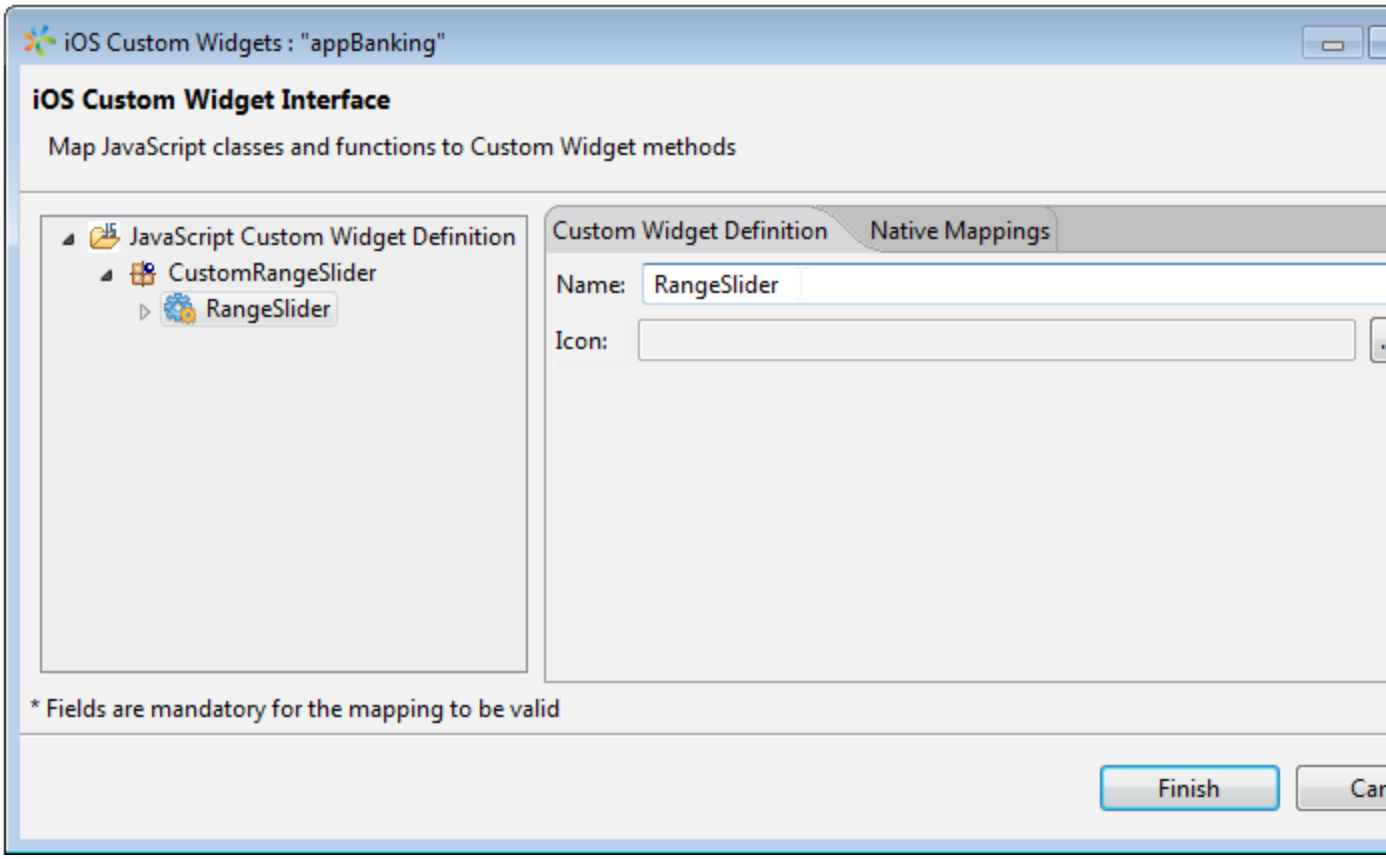
1. On the **Edit** menu of Kony Visualizer, point to **Integrate Third Party**, then **Manage Custom Widgets**, and then click **iOS**.
2. In the left pane, right-click **JavaScript Custom Widget Definition** and click **Add JavaScript Namespace**.



3. Under JavaScript Namespace, in the Name field, enter a name. The name you enter should be easily correspondable with the name of the widget you're importing, must be unique, and cannot be any of the reserved namespace names of the Kony system. For example, widget, window, segui, and so forth are reserved namespace names and should not be used for custom and third-partylibraries.
4. Click **Finish**. If you're asked if you want to proceed with generating code for the custom widget, click **Yes**.
5. To add the widget libraries to the namespace you just created, on the Project menu of Kony Visualizer, point to **Integrate Third Party**, then **Manage Custom Widgets**, and then click **iOS**
6. The namespace you created appears in the left pane. Right-click the namespace and select **Add JavaScript Widget**.



7. Specify a name and an icon for the custom widget. The icon that you specify for the custom widget will accompany the widget in the Kony Visualizer user interface in the list of available custom widgets. The supported icon format is *.png*.



8. You now import the custom widget's library. Click the **Native Mappings** tab, click the browse icon, and then click **Import** to browse to and select the zip file that consists of the protocols, event delegates, files, and images (if any) required for the custom widget.

9. On the Native Mappings tab, enter any additional frameworks required in the frameworks field. Kony provides some default frameworks required for the custom widget. You can add a framework to the list by clicking, and you can delete a framework by selecting a row, and then clicking. You can change the order of the frameworks you add using the up and down arrows. You can also insert a framework before or after a defined framework by clicking the or icons respectively. You can enable or disable frameworks by setting their values as true or false. Click **Finish**.

10. In the Widget Class text box, specify the widget class name, which should be the same as the one widget class declaration in the Custom Widget header file. For example *@interface FancyButtonWidget* in the header implies that the Widget Class name in IDE should be *FancyButtonWidget*.

11. Enter the name of the Event Delegate Protocol (if any). The Event Delegate Protocol name should be the same as the one declared in the Event delegate. For example, *@protocol FEventDelegate* in the protocol implies that the Event protocol name in IDE should be *FEventDelegate*.

12. Now that you have the library imported, you can add any additional properties that you want the custom widget to have in addition to the default properties that Kony Visualizer provides. To do so, in the left pane of the Custom Widget Interface dialog box, click the arrow ▶ of the widget you created to reveal its properties, and then click **Widget Properties**. Click, and then specify the JavaScript Type, the IDE Data Type, the Default Value for the property, indicate whether or not the property is Writable, and specify any Event Parameters (if applicable). Click **Finish**.

13. Click **Populate events and methods** to populate all the events and methods from the protocol files in the **Widget Properties** tab. Description of the **Widget Properties** tab is provided in the step below.

Delegate methods added in the Event delegate Protocol[RangeSliderEventDelegate], will be mapped with **property name** in a specific format.

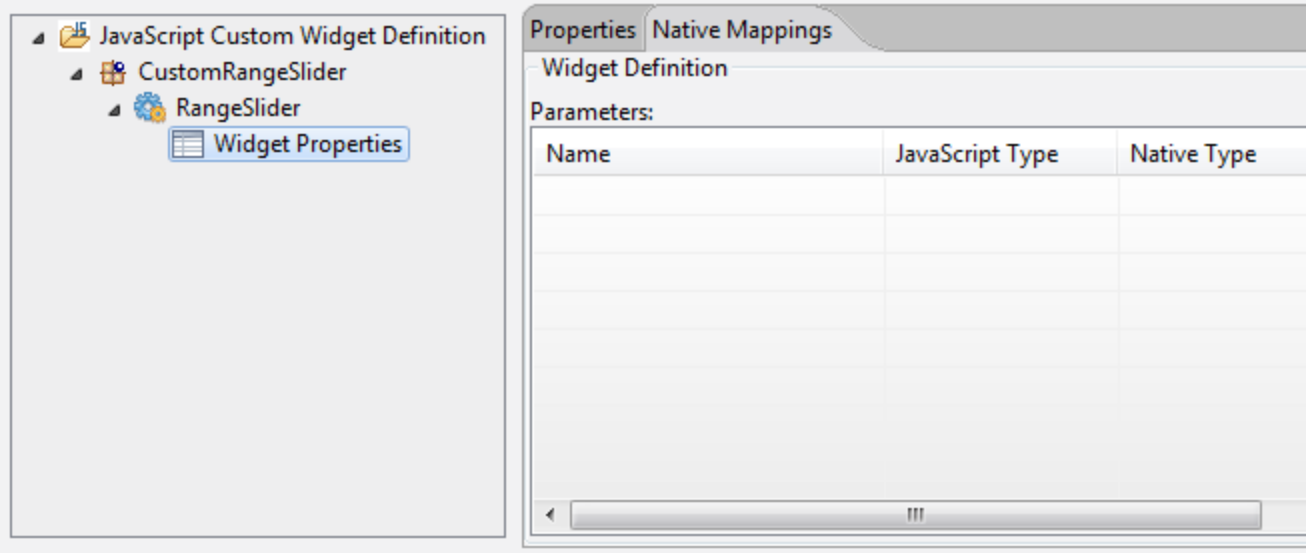
method_Name+**label_Name_For_First_Parameter+label_Name_For_Second_Parameter** ..etc.

The first letter of the parameter names must be in upper case. For example:

```
-(void)updatedMinValue:(CGFloat)minValue andMaxValue:(CGFloat)
maxValue;
```

Property name for the above delegate method is: `updatedMinValueAndMaxValue`.

14. If you want to define additional properties for a widget, do the following:
    - Expand the Custom Widget in the left pane. An entry for Widget Properties appears.



    - In the right pane, under the **Properties** tab, specify the name of the property, its corresponding JavaScript datatype, the IDE datatype, the default value of the property, and specify whether the given property is **Writable**, that is, it can be dynamically set. The datatypes for the widget properties are as follows: string, boolean, number, object, array, and function.

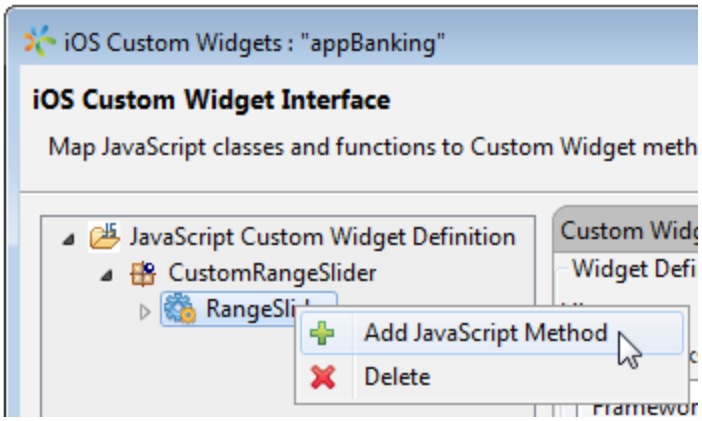
- You can add properties to the list when you click . You can delete a properties when you click a row and select . You can change the order of the properties defined using the up or down arrow. You can also insert a properties before or after a defined properties by clicking the or icons respectively.

- Under the Native Mappings tab, map each of the property to its corresponding datatype in the Native platform. To define an event in the **Widgets Properties** tab, you must specify datatype as **Function**. The Kony Visualizer automatically assigns this datatype as **Callback** in the **Native Mapping** tab.
  The following table explains the datatype mapping for iOS:

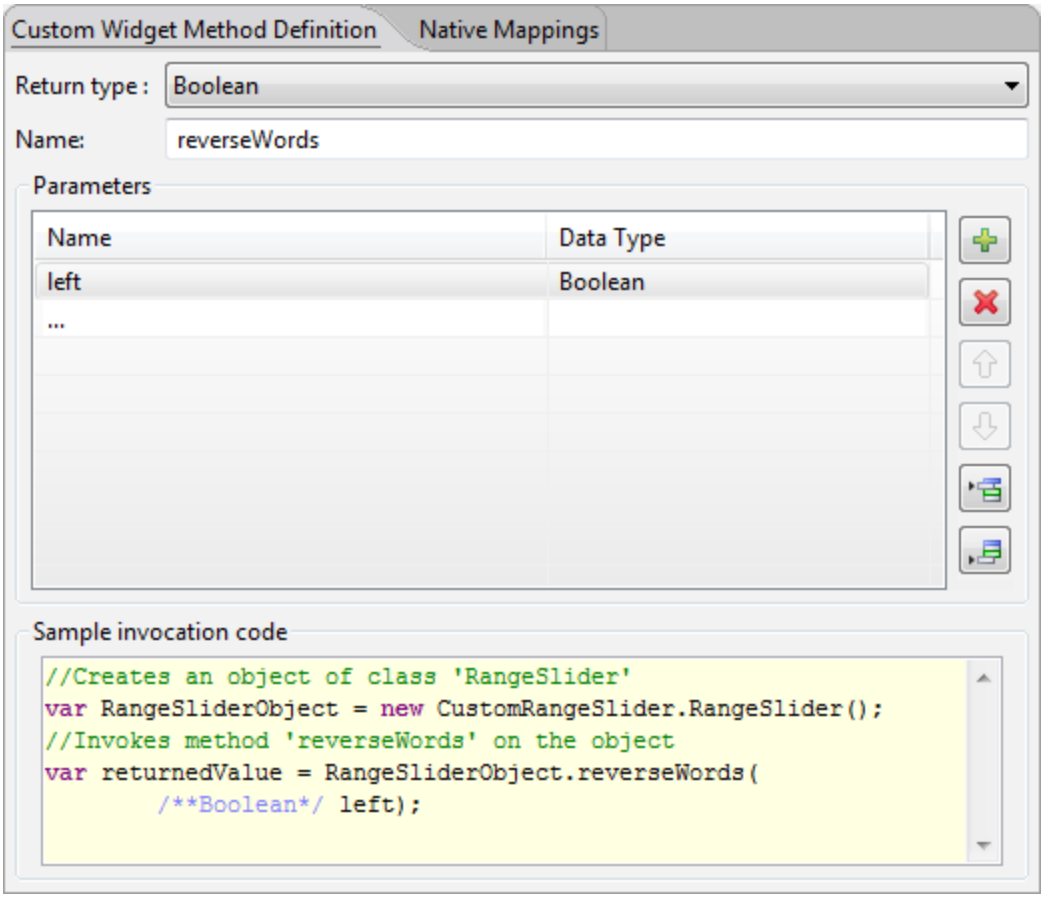
JavaScript Type	Objective -C Type
Number	int, long,float, double, NSNumber (Depending on the metadata provided)
String	NSString
Boolean	BOOL(primitive)
Array	NSArray
Object	NSDictionary, id
Function	Callback
void	void

15. To define a method associated with a widget, do the following:

    - In left pane, right-click the defined widget, and select **Add JavaScript Method**. The widget method, is added in the left pane



- In the right pane, in the **Custom Widget Method Definition** tab, you can define the parameters and corresponding datatypes of a particular method. You can also set a return type of a method.



- Under the **Native Mappings** tab, map each of the parameter to its corresponding datatype in the Native platform. JavaScript data types are represented as Object counter parts in the Native SDK as against primitive types.
  The following table explains the datatype mapping for iOS:

JavaScript Type	Objective -C Type
Number	int, long,float, double, NSNumber (Depending on the metadata provided)
String	NSString
Boolean	BOOL(primitive)
Array	NSArray
Object	NSDictionary, id
Function	Callback
void	void

- Based on the datatypes specified for the parameter and the return type, all the fields in the **Native Mappings** tab are populated. You can change the datatype for a parameter if the JavaScript type is set as Number or Object. Enter the name for the instance method (Defined in the widget protocol file and declared in the corresponding header file) to which you want to map your native method.

16. Click Finish to import the Custom Widget. A confirmation dialog appears.
17. Click **Finish** to import the Custom Widget. A confirmation dialog appears. Click **Yes**.

    The custom widget is now available at the bottom of the Widget tab, located on the Kony Library pane, and you can drag it into a container like any other widget, and view its properties on the Properties pane.

18. Click **Finish** to import the Custom Widget. A confirmation dialog appears. Click **Yes**.

    The custom widget is now available at the bottom of the Widget tab, located on the Kony Library pane, and you can drag it into a container like any other widget, and view its properties on the Properties pane.

    The Preview feature is not available for custom widgets. After you build the application you can view the library files which are built in the *....webapps/<appname>/platform/jslib/thirdparty/widgets* folder.

    > ***Important:*** If you edit the properties of a defined custom widget after you drag and drop it on a form, ensure that you delete the added instances of the custom widget on the form and add it on the form again. Since the custom widgets added on the form before the edit will not reflect the newly changed properties.

    > ***Important:*** Currently custom widget cannot be placed inside a Segment

Importing Custom Widgets for Android

You can import and create your own custom widgets for the Android platform.Kony platform has defined a contract that any custom widget should implement. This contract is referred to as a widget wrapper. The main purpose of the widget wrapper is to enable Kony platform to initialize the widget when it is placed in a form that is currently being rendered. The wrapper notifies the custom widget when there is a change in its state. The change in state can be a programmatic change to one of the properties exposed by it or a user driven event raised on the custom widget.

All changes in state are notified to the custom widget through the widget wrapper contract. If you want to use a widget developed by using any 3rd party library or framework, you should expose the widget to the Kony Android platform by implementing a Java abstract class defined by the Kony Android platform. This java class implementation should be done in a separate java project which should be exported as a library jar file. These class files in the library jar are imported using the Importing Custom Widgets procedure.

Android Wrapper

The contract definition is as follows:

public abstract android.view.View onCreateView (android.content.Context context)

This mandatory method is called by the platform whenever a new instance of custom widget view is required. The subclasses should not hold the reference to the created View and the context argument in any local instance variables. If reference is held, it should be nullified in onDestroyView callback, otherwise would lead to Out of Memory issues.

public void onPropertySet (View widgetInstance, Object name, Object value)

This method is called by the platform whenever a value is assigned to custom widget property by the application. The widgetInstance argument can be null if the custom widget View component is not yet created.

public void Object onPropertyGet (View widgetInstance, Object key)

This method is invoked by the platform when a property is read by the application. The widgetInstance argument can be null if the custom widget view component is not yet created. If this method is not overridden, or if overridden and do not handle the property key i.e., return null, then platform will pick the value of the property from the model. Subclasses can use the setModelProperty to update the model.

public final void setPropertyToModel(String name, Object value)

This method adds or updates the model for a given property with a given value.

public final Object getPropertyFromModel(String name)

This method returns the value for a given property name

public abstract void onDestroyView(View widgetInstance)

This method is called by the platform just before View instance is freed by the platform.

public void onOrientationChanged(int orientation)

This is a callback invoked by platform whenever an orientation is changed for a Form whose orientation mode is set to BOTH. The value of orientation argument is as that of android platform orientation constants.

> ***Important:*** The abstract class for these contracts are available in the following JAR files at:
>
> - Location in the **Workspace** folder: **\temp\AppName\build\luaandroid\dist\AppName\libs\konywidgets.jar**
> - Location in the **Android-SDK** folder: **\android-sdk\platforms\android-18\android.jar**

Android wrapper Java class for custom widget implementation should be done in a separate java project that should be exported as a library JAR file. Then this library JAR is imported using the Importing Custom Widgets procedure.

Creating a Custom Widget Wrapper Java Project

The java project implementing custom widget require methods/interfaces from Kony Android platform to be able to compile the project. These methods are available in the `konywidgets.jar` file. The project must include this JAR file in its project properties as a dependent library.

To get `konywidgets.jar` file, do the following:

1. Open the project in Kony Visualizer and build the project for Android.
2. Navigate to the location `<workspace>/temp/<app>/build/luaandroid/dist/<app>/libs` directory. It contains a `konywidgets.jar` file.

> ***Note:*** The `konywidgets.jar` file can either be:
>
> - Copied to the `libs` directory in the Java project that is created for developing the custom widget. Or
> - Added as dependent library in the Java project's properties by giving the path to this file.

Java application has to be compiled and exported as a JAR file. It will be used during custom widget importing in Kony Visualizer.

Sample wrapper.class File for Android

```
package com.example.bar;

//The following APIs are part of android.jar file

import android.content.Context;
import android.util.Log;
import android.view.View;

//The following APIs are part of konywidgets.jar file

import com.example.bar.RangeSeekBar.OnRangeSeekBarChangeListener;
import com.konylabs.api.ui.KonyCustomWidget;
import com.konylabs.vm.Function;

//The class name specified must be the same as the name of the custom
widget in Kony Visualizer
public class CustomRange extends KonyCustomWidget {

    Function jsCallback;
    RangeSeekBar < Integer > rangeBar;
    private static String TAG = "CustomRange";
```

```
    //This is a mandatory method
    public View onCreateView(Context context) {
        int low = ((Double) getPropertyFromModel("low")).intValue();
        int high = ((Double) getPropertyFromModel("high")).intValue();
        rangeBar = new RangeSeekBar < Integer > (low, high, context);
        rangeBar.setOnRangeSeekBarChangeListener(new
OnRangeSeekBarChangeListener < Integer > () {@
            Override
            public void onRangeSeekBarValuesChanged(RangeSeekBar <? >
bar, Integer minValue, Integer maxValue) {
                // handle changed range values
                try {
                    jsCallback.execute(new Object[] {
                        minValue, maxValue
                    });
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                Log.i("FullscreenActivity", "User selected new range
values: MIN=" + minValue + ", MAX=" + maxValue);
            }
        });
        jsCallback = (Function) getPropertyFromModel("slideCheck");
        return rangeBar;
    }
    //Destroys the widget instance when the form goes out of view
    public void onDestroyView(View widgetInstance) {
        // TODO Auto-generated method stub
        //setPropertyToModel("text",b.gettext());

        widgetInstance = null;
    }
```

```
    public void printLog() {
        Log.d(TAG, "**********CUSTOM
WIDGET****************************************");
    }


    public void onPropertySet(View widgetInstance, Object key, Object
value) {
        Log.d(TAG,
"**********onPropertySe
t****************************************");
        String k = ((String) key).intern();
        if (k == "low")
            ((RangeSeekBar < Number > )
widgetInstance).setSelectedMinValue(((Double) value).intValue());
        else if (k == "high")
            ((RangeSeekBar < Number > )
widgetInstance).setSelectedMaxValue(((Double) value).intValue());
    }

    public Object onPropertyGet(View widgetInstance, Object key) {
        String k = ((String) key).intern();
        Log.d(TAG,
"**********onPropertyGe
t****************************************");
        if (k == "low")
            ((RangeSeekBar < Number > )
widgetInstance).getSelectedMinValue();
        else if (k == "high")
            ((RangeSeekBar < Number > )
widgetInstance).getSelectedMaxValue();
        return null;
```

```
    }

    public void onOrientationChanged(int orientation) {}
}
```

Limitations

1. As JavaScript is dynamic typed language, all it understands is a number which can hold any float number. So, all number type arguments are mapped to Double always. It is the responsibility of the custom widget implementer to convert the double type to the respective types. If not, you would see class cast exception.
2. You cannot pass any kind of java object to JavaScript. It should be either Vector (for JS array) or Hashtable (for JS Object).

Importing Custom Widgets for Android

To import files for a third-party widget, execute the following:

1. On the **Edit** menu of Kony Visualizer, point to **Integrate Third Party**, then **Manage Custom Widgets**, and then click **Android**.
2. In the left pane, right-click **JavaScript Custom Widget Definition** and click **Add JavaScript Namespace**.
3. Under JavaScript Namespace, in the Name field, enter a name. The name you enter should be easily correspondable with the name of the widget you're importing, must be unique, and cannot be any of the reserved namespace names of the Kony system. For example, widget, window, segui, and so forth are reserved namespace names and should not be used for third-party libraries.
4. Click **Finish**. If you're asked if you want to proceed with generating code for the custom widget, click **Yes**.

5. To add the widget libraries to the namespace you just created, on the Project menu of Kony Visualizer, point to **Integrate Third Party**, then **Manage Custom Widgets**, and then click **Android**.

6. The namespace you created appears in the left pane. Right-click the namespace and select **Add JavaScript Widget**.

7. Specify a name and an icon for the custom widget. The icon that you specify for the custom widget will accompany the widget in the Kony Visualizer user interface in the list of available custom widgets. The supported icon format is *.png*.

8. You now import the custom widget's library. Click the **Native Mappings** tab, click the Manage Libraries icon  , and then click **Import** to browse to and select the `.jar` file that consists of the protocols, event delegates, files, and images (if any) required for the custom widget.

9. Enter the **Package** and **Widget Class** names in the appropriate fields. The names should be the same as the package name and class name in the wrapper file.

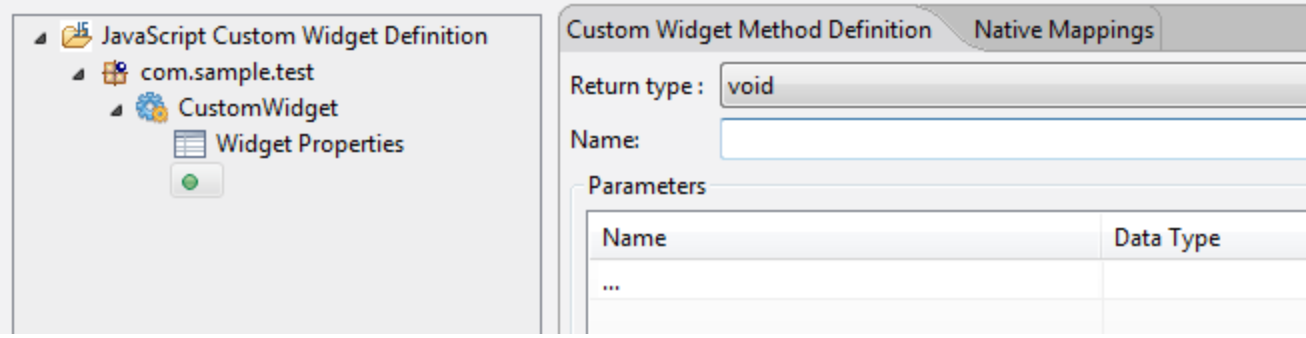
10. Now that you have the library imported, you can add any additional properties that you want the custom widget to have in addition to the default properties that Kony Visualizer provides. To do so, in the left pane of the Custom Widget Interface dialog box, click the arrow ▶ of the widget you created to reveal its properties, and then click **Widget Properties**. Click  , and then specify the JavaScript Type, the IDE Data Type, the Default Value for the property, indicate whether or not the property is Writable, and specify any Event Parameters (if applicable). Click **Finish**.

11. On the Native Mappings tab, map each of the property to its corresponding datatype in the Native platform. To define an event in the Widgets Properties tab, you must specify datatype as **Function**. Kony Visualizer automatically assigns this datatype as **Callback** in the Native Mapping tab.

The following table explains the datatype mapping for Android:

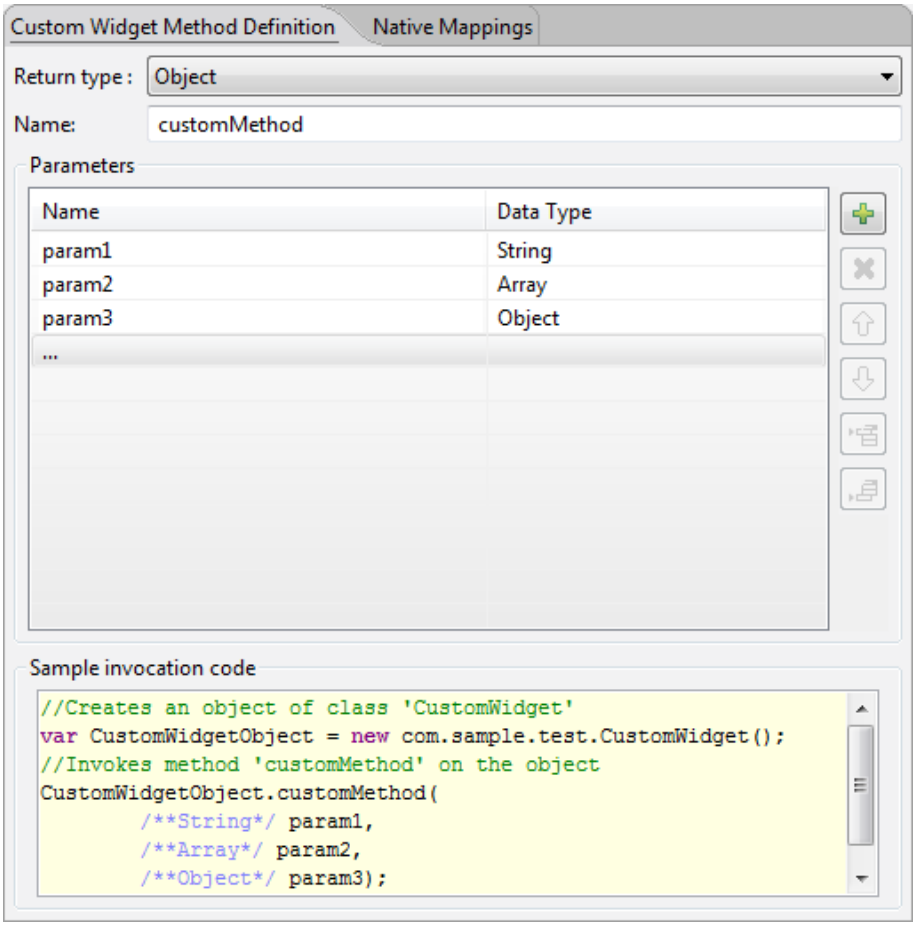
JavaScript Type	Native Type
Number	int, long,float, double , java.lang.Integer, java.lang.Long, java.lang.Double, java.lang.Float(Depending on the metadata provided)
String	java.lang.String
Boolean	java.lang.Boolean, boolean
Array	java.util.Vector
Object	java.lang.Object, java.lang.Hashtable
Function	com.kony.vm.Function

12. To define a method associated with a widget, do the following:
    - In left pane, right-click the defined widget, and select **Add JavaScript Method**. The widget method, is added in the left pane



    - In the right pane, in the **Custom Widget Method Definition** tab, you can define the parameters and corresponding datatypes of a particular method. You can also set a

return type of a method.

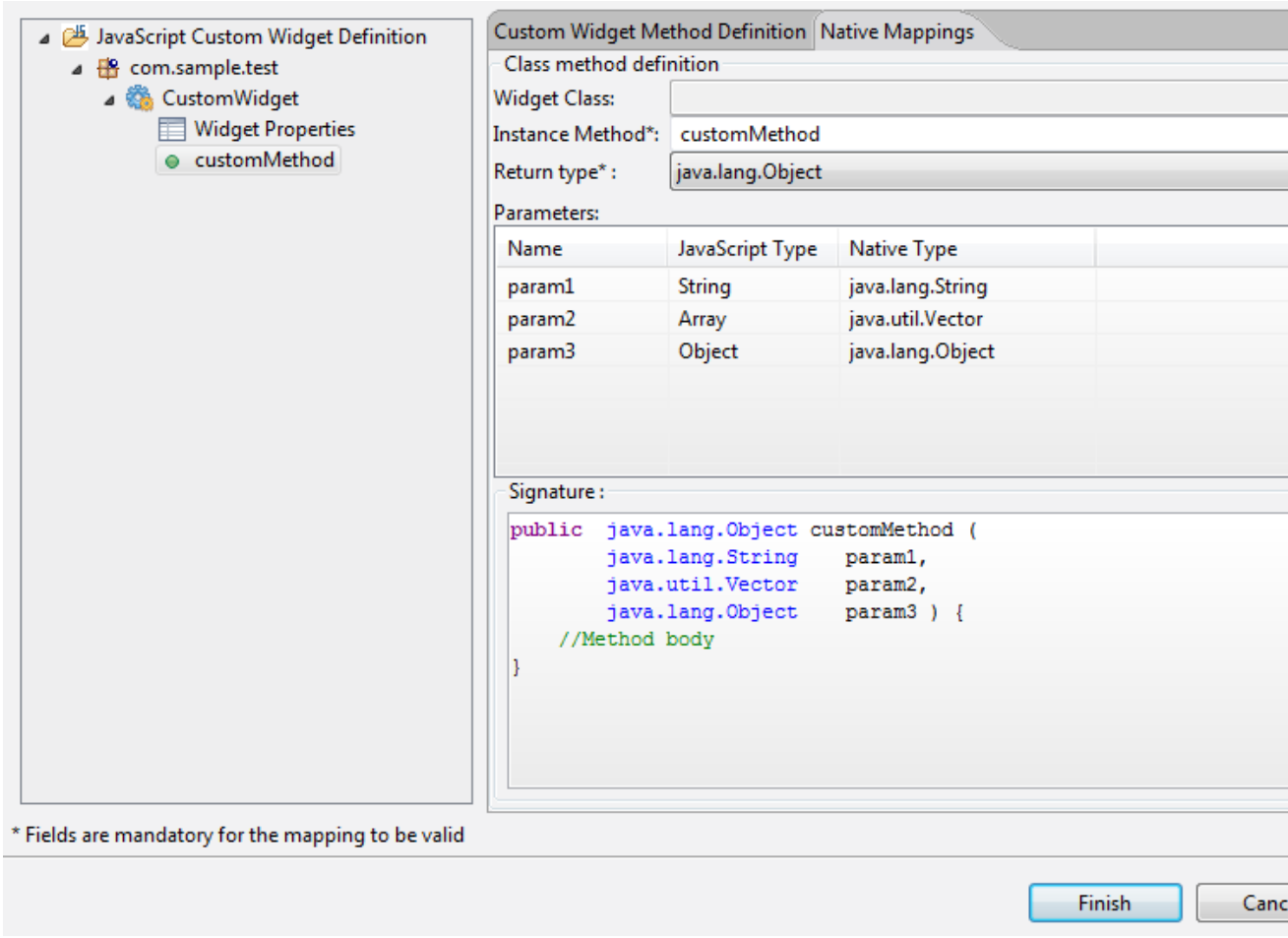


- Under the Native Mappings tab, map each of the parameter to its corresponding datatype in the Native platform. JavaScript data types are represented as Object counter parts in the Native SDK as against primitive types. For example, *java.lang.Double* is used instead of *double* and *java.lang.Boolean* instead of *boolean*.

The following table explains the datatype mapping for Android:

JavaScript Type	Native Type
Number	int, long,float, double , java.lang.Integer, java.lang.Long, java.lang.Double, java.lang.Float (Depending on the metadata provided)
String	java.lang.String
Boolean	java.lang.Boolean, boolean
Array	java.util.Vector
Object	java.lang.Object, java.lang.Hashtable
Function	com.kony.vm.Function

- Based on the datatypes specified for the parameter and the return type, all the fields in the **Native Mappings** tab are populated. You can change the datatype for a parameter if the JavaScript type is set as Number or Object. Enter a name for the instance method in which the parameters are passed. If any property or parameter to the method is of type Function, then this function/method/callback can be executed from the Java class with the help of a wrapper method provided by the platform.




```
public void printLog(String name, Function callbackJS) {
    try {
        callbackJS.execute(new Object[] {
            infotable
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
If the callback / method has time taking tasks then it should use
the method "executeAsync” instead of "
execute”. And it is always advised not to perform time taking
```

```
tasks in these callbacks.Also use "executeAsync” as much as
possible.try {                  callbackJS.executeAsync(new Object[]
{infotable});}catch(Exception e) {e.printStackTrace();}}
```

13. Click **Finish** to import the Custom Widget. A confirmation dialog appears. Click **Yes**.

    The custom widget is now available at the bottom of the Widget tab, located on the Kony Library pane, and you can drag it into a container like any other widget, and view its properties on the Properties pane.

    The Preview feature is not available for custom widgets. After you build the application you can view the library files which are built in the *....webapps/<appname>/platform/jslib/thirdparty/widgets* folder.

> ***Important:*** If you edit the properties of a defined custom widget after you drag and drop it on a form, ensure that you delete the added instances of the custom widget on the form and add it on the form again, since the custom widgets added on the form before the edit will not reflect the newly changed properties.

Important Considerations

1. All resources related to custom widget must reside in this location: `<workspace>/<app>/resources/mobile/native/android`.
2. Under resources folder, sub-directories include: assets, drawable, layout, anim, values, xml, raw, and so on.
3. In the custom widget, these resources cannot be accessed using resource IDs directly like R.drawable.xxx.

   These resources can be accessed as follows:

```
int resId = KonyMain.getAppContext().getResources
().getIdentifier(resName, type, KonyMain.getAppContext
().getPackageName());
```

- **Scenario 1:** If the resource name is `logo.png` and it is present in `drawable` (type) directory, then resource ID can be retrieved as follows:

```
int resId = KonyMain.getAppContext().getResources
().getIdentifier("logo.png", "drawable",
KonyMain.getAppContext().getPackageName());
```

- **Scenario 2**: If the xml layout resource `calendar_view.xml` is present in `layout` directory, then it can be retrieved as follows:

```
KonyMain.getAppContext().getResources().getLayout
(KonyMain.getAppContext().getResources().getIdentifier
("calendar_view", "layout", KonyMain.getAppContext
().getPackageName()))
```

4. Any UI update made to custom widget must be posted as a runnable to UI Thread as follows:

```
KKonyMain.getAppContext().getResources().getLayout
(KonyMain.getAppContext().getResources().getIdentifier
("calendar_view", "layout", KonyMain.getAppContext
().getPackageName()))((Activity) KonyMain.getActivtiyContext
()).runOnUIThread(new Runnable() {
    public void run() {
        //Do necessary Custom widget UI updates
    }
});
```

5. To receive touch events for the widget (If custom widget is placed inside a scrollable container), it needs to override below function and the widget or view must be changed to

ViewGroup.

```
public boolean onInterceptTouchEvent(MotionEvent ev) {
    int action = ev.getAction();
    switch (action) {
        case MotionEvent.ACTION_DOWN:
            this.getParent
().requestDisallowInterceptTouchEvent(true);
            break;
    }
    return super.onInterceptTouchEvent(ev);
}
```

Custom Widgets for SPA and Desktop Web

A custom widget are a group of widgets created to perform a specific task, functionality, slot or signal.

In Kony Visualizer, you can use the following types of custom widgets.

- Custom widgets created using third party web libraries and frameworks.
- Custom widgets created in Kony Visualizer and wrapped as components. These components can be re-used in other applications. For more information about how to create components and use them, refer the Working with Components.

This document describes the process of using custom widgets using third party web libraries and frameworks in SPA and Desktop web channels and how to import them as a component.

This section provides information about the following processes of custom widgets in SPA and Desktop web applications.

- Defining new custom widgets in Kony Visualizer
- Import the Custom Widget into Kony Visualizer
- Edit Custom Widget and Wrapper in Kony Visualizer

- Export a Custom Widget in Kony Visualizer
- Delete a Custom Widget in Kony Visualizer

Defining New Custom Widgets in Kony Visualizer

You can define custom widgets in Kony Visualizer using a third party library or framework.

This process involves the following three tasks:

- Copy the Assets to the Workspace
- Write and Import the Contract Definition File

> ***Note:*** For SPA applications, to view the added custom web widget in the Default Library, you must open the form in the web view of the canvas. To open the form in web view, from the top of the canvas(near the button BVR), select the platform as web. For example, for an iOS SPA application, from the drop-down, select **iOS Mobile: Web**.
> In Desktop Web application, the Default Library opens in web view by default.

Copy the Assets to the Workspace

Ensure that all the custom widget assets(.css and .js files) are in a folder in your **workspace**[1] before you begin importing a custom widget.

For example, the resource location path is `<workspace folder>\CustomWidgetTest\resources\thirdpartywidgets`

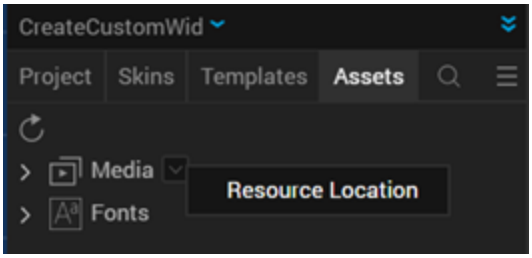
Let us consider an example where you want to import a custom widget (named **NewGrid**) that is built on a **jQuery** library. Ensure that you copy the library files, .css, .js, and the contract definition files into your workspace.

To copy the custom widget assets to your workspace:

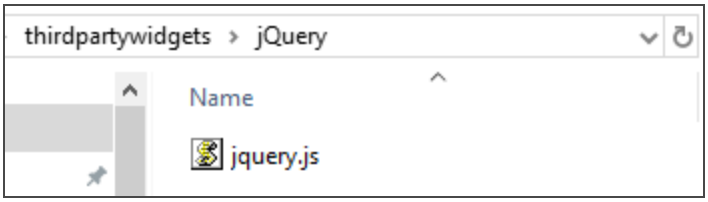
---

[1]A repository for the applications and any other projects you create (such as a patch or new feature). It resides on your computer as a folder, the default path being C:\Workspace.

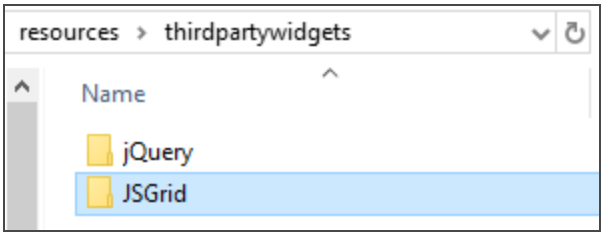
1. In Kony Visualizer, create a new project.
2. Create and design a form for the Responsive Web/ Desktop channel.
3. From the Project Explorer, go to **Assets**.
4. Right-click on **Media** and click **Resource Location**.



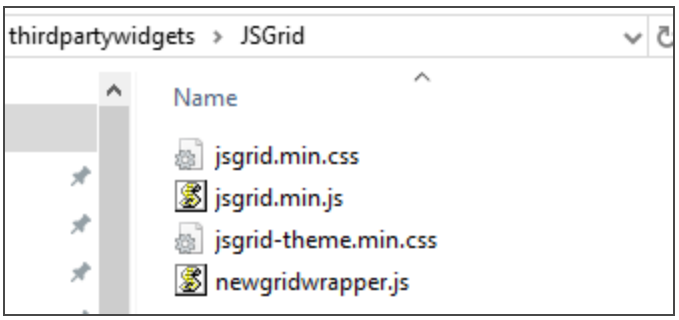
5. Create a new folder and name it **thirdpartywidgets**, if it doesn't exist already.
6. In the **thirdpartywidgets** folder, create a new directory to store the library file on which the custom widget is developed. In this example, **jQuery**.
7. Add your .js file to the **jQuery** folder. In this example, the **jQuery.js** file.



8. Create a new directory to store the .css and .js files, in this example, **JSGrid**.



9. Add your .css and .js files to the newly created directory. The files must include the contract definition file. In this example, the **wrapper.js** file.



To learn more about the contract definition file, refer Write and Import the Contract Definition File. Once you copy the custom widget assets to your workspace, you must import the custom widget into your Kony Visualizer project.

> ***Note:*** Multiple widgets can be created using the same library file.

Write and Import the Contract Definition File

All changes in state are notified to the custom widget through the widget wrapper contract (*modelChange*). The developer of the custom widget and its wrapper can make use of the information provided in the *modelChange* call and make changes to the custom widget state accordingly.

If you want to use a widget developed by using any third party library or framework, you should expose the widget to the Kony SPA platform by implementing an interface defined in JavaScript. The contract definition is as follows:

```
/*Using this function you initialize the widget. A parentNode is  a
container where you place the custom widget. It is the placeholder div
that respects the layout of a Kony Widget and the custom widget is
placed in the parentNode.The widgetModel parameter is a hashmap
containing key - value pairs of the properties and its corresponding
values.*/

function initializeWidget(parentNode, widgetModel)

/*Using the modelChange function any changes to the widget property is
mapped both in Kony widget model and the third party widget property.
The widgetModel is a hash map containing key-value pairs of the
```

```
properties and its corresponding values. propertyChanged is the key of
the property being changed. propertyValue is the value assigned to the
key.*/

function modelChange(widgetModel, propertyChanged, propertyValue)
```

Sample wrapper *.js file for SPA

Here is the contract definition file for the **NewGrid** widget that you want to import into your project.

```
/*Ensure the name of the class is the same as that of the widget name
specified through IDE*/
NewGrid = {
    /*Initialize widget is invoked by SPA for widget initialization
using the constructor*/
    initializeWidget: function(parentElement, widgetModel) {
        parentElement.innerHTML = '<div id="jsGrid"
style="height:100%; width:100%;"><div id="div1"></div></div>';
        var clients = [{
            "Name": "Otto Clay",
            "Age": 25,
            "Country": 1,
            "Address": "Ap #897-1459 Quam Avenue",
            "Married": false
        }, {
            "Name": "Connor Johnston",
            "Age": 45,
            "Country": 2,
            "Address": "Ap #370-4647 Dis Av.",
            "Married": true
        }, {
            "Name": "Lacey Hess",
            "Age": 29,
            "Country": 3,
```

```
            "Address": "Ap #365-8835 Integer St.",
            "Married": false
        }, {
            "Name": "Timothy Henson",
            "Age": 56,
            "Country": 1,
            "Address": "911-5143 Luctus Ave",
            "Married": true
        }, {
            "Name": "Ramona Benton",
            "Age": 32,
            "Country": 3,
            "Address": "Ap #614-689 Vehicula Street",
            "Married": false
        }];

        var countries = [{
                Name: "",
                Id: 0
            }, {
                Name: "United States",
                Id: 1
            }, {
                Name: "Canada",
                Id: 2
            }, {
                Name: "United Kingdom",
                Id: 3
            }


        ];
```

```
        $("#jsGrid").jsGrid({
            height: "auto",
            width: "100%",
            inserting: true,
            editing: true,
            sorting: ((widgetModel.sortable === 'true' ||
widgetModel.sortable === true) ? true : false),
            paging: true,
            data: clients,
            fields: [{
                name: "Name",
                type: "text",
                width: 150,
                validate: "required"
            }, {
                name: "Age",
                type: "number",
                width: 50
            }, {
                name: "Address",
                type: "text",
                width: 200
            }, {
                name: "Country",
                type: "select",
                items: countries,
                valueField: "Id",
                textField: "Name"
            }, {
                name: "Married",
                type: "checkbox",
                title: "Is Married",
                sorting: false
```

```
            }, {
                type: "control"
            }]
        });
    },

    modelChange: function(widgetModel, propertyChanged, propertyValue)
{
        if (propertyChanged === 'data') {
            $("#jsGrid").jsGrid({
                data: propertyValue
            });
        } else if (propertyChanged === 'sortable') {
            $("#jsGrid").jsGrid({
                sorting: ((propertyValue === 'true' || propertyValue
=== true) ? true : false)
            });
        }
    }
};
```

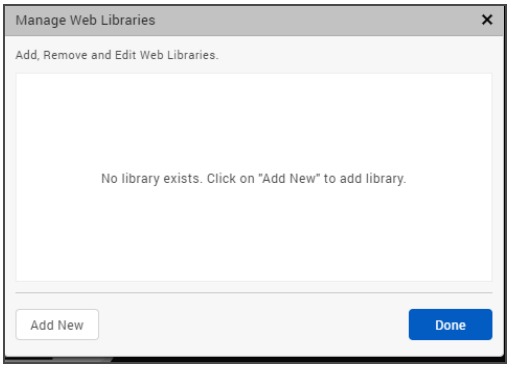
Important: If you specify an absolute value for a property in the library files, the widgets placed next to it or after it might be distorted. We recommend not to provide an absolute value.

Important: Third-party container custom widgets such as tab panes are not supported.

Import the Custom Widget into Kony Visualizer

Here are the steps for importing a third-party custom widgets or web library in SPA and Desktop Web application using Kony Visualizer.

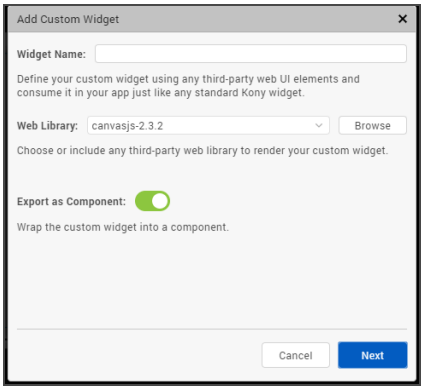
1. In Kony Visualizer, open the application to which you want to add the custom web widget.
2. From the **Edit** menu of the Visualizer, select **Manage Web Libraries**.
   The **Manage Web Libraries** window appears.



3. To add a new custom widget, click **Add New**.
   The **Add Custom Widget** window appears.
4. In the **Add Custom Widget** window, provide the following information:
   - **Widget Name**: The name used to display the custom widget in your application.
   - **Widget Library**: Click **Browse** to navigate to the location of the folder containing the custom widget/ library.

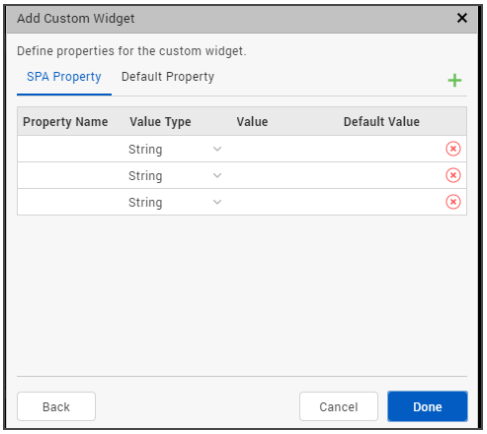
     > ***Note:*** To reuse a third-party library already added to your application, select the required library from the **Widget Library** drop-down list.

   - **Export as Component**: Toggle the switch to **On** to add the custom widget as a component to your application.



When you Toggle the switch to **Off**, the custom widget is not added as a component. The custom widget is added as a new widget to the Default Library of Kony Visualizer.

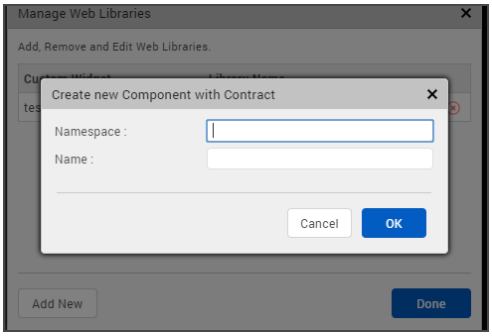
5. Click **Next**.
6. In the new window that appears customize the properties of the custom widget as per your requirement, and then click **Done**.



The **Create New Component with Contract** window appears.

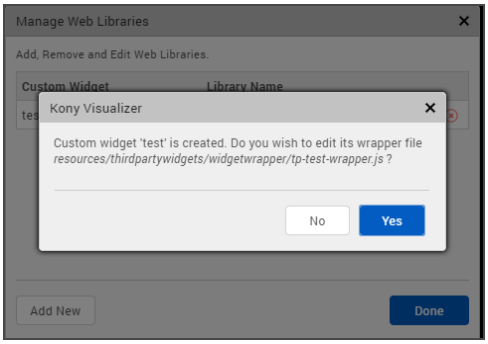
7. Provide the following details in the Create New Component with Contract window:
   - **Namespace**: Provide the name of the namespace of the component

- **Name**: Provide the name of the component that will contain the custom widget.



8. Click **OK**.

9. The custom widget is added to your application with a default wrapper.
   Click **Yes**, to edit the custom widget wrapper.js file.
   Click **No**, if not required.



10. Click **Done**.
    The new custom widget has been added to the **Templates** explorer, under Components.
    The custom widget is also be added under Default Library on the bottom-left of your canvas.

Edit Custom Widget and Wrapper in Kony Visualizer

After adding the Custom widgets, you can edit the custom widget and wrapper.js file in Kony Visualizer.

You can edit all the properties of the custom widget except for the name of the custom widget.

Follow these steps to edit the custom widget and widget wrapper.js file.

1. From the **Edit** menu of the Visualizer, select **Manage Web Libraries**.
2. In the **Manage Web Libraries** window, select the  icon to edit the custom widget.
   The **Edit Custom Widget** window appears.
3. Click **Next**.
4. In the new window, customize the properties of the custom widget as required.
5. In the same window, under **Files**, for the custom widget wrapper, select  icon to edit the wrapper.js file.
   The custom widget wrapper file opens in the JS code editor.

Export a Custom Widget in Kony Visualizer

Follow these steps to edit the custom widget and widgetwrapper.js file.

1. From the **Edit** menu of the Visualizer, select **Manage Web Libraries**.
2. In the **Manage Web Libraries** window, select the  icon to export the custom widget.
   A file explorer window appears.
3. Select a folder from your system.
   Click **Select Folder**.

The custom widget is now downloaded as a zip file into the specified folder.

Delete a Custom Widget in Kony Visualizer

Follow these steps to delete the custom widget from your applicaation.

1. From the **Edit** menu of the Visualizer, select **Manage Web Libraries**.
2. In the **Manage Web Libraries** window, select the  icon to delete the custom widget.
   A Visualizer dialog appears.

3. Click **Yes**.
   The custom widget is now deleted from your application.

Organizing and Moving Application Elements

As you add forms, modules, and other items to your application, you may want to organize related application elements into groups, or copy and move them. You can create application groups for related forms and other elements, then drag and drop forms and elements from one application group to another, or copy individual application element to the Clipboard library.

You can then move application elements to an application group or collection, or export them. You can also save an application group to a collection, or export and import application groups between projects.

In addition to including forms, modules, and related elements in your project, you can make a variety of related resources such as test scripts, specifications, and HTML files available in your project by adding them to the **Other Resources** folder in Project Explorer.

The following topics provide additional information about using application groups, the Clipboard library, and the **Other Resources** folder:

Create an Application Group

Add an Application Group to a Collection

Copy an Application Element to the Clipboard Library

Export and Import an Application Group

Adding Other Resources to Your Project

Create an Application Group

You can create a new application group for mobile, tablet, desktop or watch channels. Kony Visualizer automatically creates an a group with the same name for the form's associated modules and controllers.

To create an application group in a Kony Reference Architecture project:

1. In Project Explorer, expand the **Mobile**, **Tablet**, **Desktop**, or **Watch** node where you want to add the application group.
2. Right-click the **Forms** node or an existing application group, and select **New Group**. Kony Visualizer creates a new application group within the **Forms** node or application group, and adds a corresponding group to the **Controllers** node.

   For mobile, tablet, and desktop channels, Kony Visualizer also adds a corresponding group to the **require** folder of the **Modules** node. For the watch channel, Kony Visualizer adds a corresponding group to the **Watch** folder of the **Modules** node.

   You can nest application groups and include a variety of elements within the group, including forms and components. You can also move elements between groups by dragging and dropping them from one group to another.

To create an application group in a Free Form JavaScript project:

1. In Project Explorer, expand the **Mobile**, **Tablet**, **Desktop**, or **Watch** node where you want to add the application group.
2. Right-click the **Forms** node or an existing application group, and select **New Group**. Kony Visualizer creates a new application group within the **Forms** node or application group, and adds a corresponding group to the **Modules** node. For the watch channel, Kony Visualizer adds the corresponding group to the **Watch** folder of the **Modules** node.

You can nest application groups and include a variety of elements within the group, including forms and components. You can also move elements between groups by dragging and dropping them from one group to another.

Add an Application Group to a Collection

Once you have created an application group, you can make it available to all of your application projects by adding it to a collection.

To add an application group to a collection:

1. In Project Explorer, right-click the application group that you want to add to a collection, and hover over **Add to Collection**.
2. Hover over the library that contains the collection, and click the collection where you want to save the component. If the library or collection does not exist, you can create the library or collection.

   If any elements of the application included bundled services, Kony Visualizer prompts you to select and package the services. When you use the application group in an application, Kony Visualizer adds the services to the application.

Copy an Application Element to the Clipboard Library

You can make an individual application element available to other application projects by copying it to the built-in Clipboard library. You can copy a single form, multiple forms, components, code modules, and masters created in earlier versions of Kony Visualizer. When you copy forms, any associated templates or controller modules are copied with them.

Once you have copied an application element to the Clipboard library, you can insert it into an application group, move it to a collection, or export it to your computer, network, or the cloud.

> ***Note:*** There are some restrictions on what can be copied to the Clipboard library, depending on the type of application project. When you copy an application element to the Clipboard library from a Kony Reference Architecture project, you cannot insert into or move it to a Free Form JavaScript project. Also, when you copy a form from a Free Form JavaScript project to the Clipboard library, named actions and popups are not included. For more information on types of Kony custom application projects, see Types of Projects.

To copy an application element to the Clipboard library:

1. In Project Explorer, right-click the form, component, template, or code module, and then select **Copy to Clipboard Library**. Kony Visualizer displays the **Create New Library Component** dialog box.
2. In the **Create New Library Component** dialog box, specify a name and description for the application element, and then click **OK**. Kony Visualizer adds the application element to the Clipboard library in Library Explorer.

To insert an application element in the Clipboard library into an application group:

1. In Library Explorer, select the My Libraries tab and navigate to the Clipboard library.
2. For a form, application group, or component, right-click the application element and then hover over **Insert Into**. For a code module, right-click the application element and then hover over **Merge Into**.
3. Select the application group where you want the application element to be inserted. Kony Visualizer inserts the application element into the application group.

To move an application element in the Clipboard library to a collection:

1. In Library Explorer, select the My Libraries tab and navigate to the Clipboard library.
2. Right-click the application element, and then hover over **Move To**.

3. Select the library and collection where you want the application element to be moved. Kony Visualizer moves the application element to the collection.

To export an application element in the Clipboard library to your computer, network, or the cloud:

1. In Library Explorer, select the My Libraries tab and navigate to the Clipboard library.
2. Right-click the application element, and then hover over **Export**. Kony Visualizer displays the **Save As** dialog box.
3. Navigate to the location where you want to export the application element, and then click **Save**. Kony Visualizer exports the application group to that location.

Export and Import an Application Group

You can share application groups between projects without adding them to a collection by exporting and importing them. For example, if you have created an application group in your project and want to use it in a different project, you can export it to your computer, network, or the cloud, and then import it into another project.

To export an application group:

1. In Project Explorer, right-click the application group, and then select **Export**. Kony Visualizer displays the **Save As** dialog box.
2. Navigate to the location where you want to export the component, and then click **Save**. Kony Visualizer exports the application group to that location.

The exported application group includes all forms and corresponding modules associated with the application group, including any nested folders.

To import an application group:

1. In Project Explorer, expand the **Mobile**, **Tablet**, **Desktop**, or **Watch** node where you want to add the application group.

2. Right-click the **Forms** node or an existing application group, and select **Import**. Kony Visualizer displays the **Import Forms** dialog box.
3. Click **Browse** to navigate to the location of the application group, select the application group, and then click **Import**. Kony Visualizer adds the application group and its associated application items to your project.

The imported application group includes all forms and corresponding modules associated with the application group, including any nested folders.

Adding Other Resources to Your Project

In addition to including forms, modules, and related elements in your project, you can add a variety of related resources such as test scripts, specifications, and HTML files. These resources are listed in the **Other Resources** folder in Project Explorer.

To add resource files to your project:

1. In Project Explorer, right-click the **Other Resources** folder and select **Resource Location**. Kony Visualizer opens the **otherresources** folder on your computer.
2. Drag and drop the files you want to add to the project to the **otherresources** folder.
3. To view the list of resource files in Kony Visualizer, right-click the **Other Resources** folder and select **Refresh**.

To view or edit a resource file in Kony Visualizer:

1. In Project Explorer, open the **Other Resources** folder.
2. Select the file name, or right-click the file name and select **Open**. Kony Visualizer displays the file in an editable window for the following file types:

- Text (.txt)
- Gherkin (.feature)
- JSON (.json)
- XML (.xml)
- Java (.java)
- Python (.py, .pw)
- HTML (.html, .htm)
- Javascript (.js)
- CSS (.css)
- Swift (.swift)
- Plist (.plist)

Kony Visualizer opens files with other file types using the default application associated with the file type.

Add Custom CSS Code to an SPA App

You can customize the look and feel of certain widgets in an SPA app by adding your own CSS code. This code replaces certain properties that you would otherwise set on the **Skin** tab of a widget, including background, border, font, and shadows.

In SPA and Desktop web channels, on some widgets, you have to use selectors in the custom CSS and for other widgets, you don't have to use selectors.

You must write the custom CSS along with selectors such as `.labelskin1 { background-color: red; }` for the following widgets:

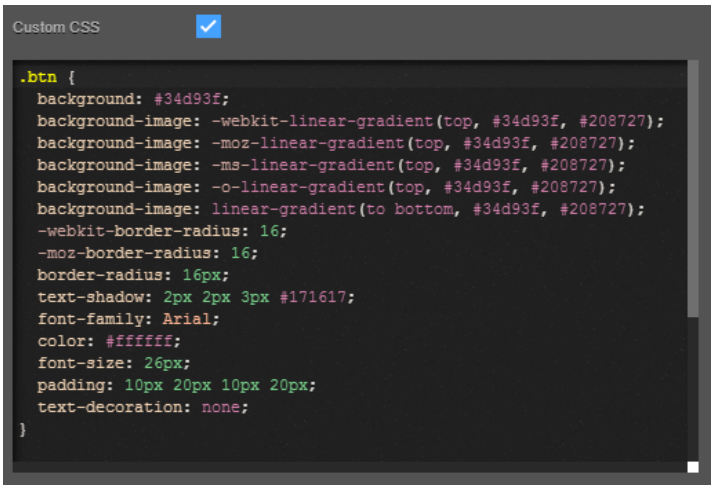
- Label
- Segment
- Tabpane
- Menu container
- Switch
- Datagrid

For all remaining widgets, you can write custom CSS without selectors for example, `background-color:red;`.

To add custom CSS code to an SPA app, do the following:

1. On the **Project** tab of the Project Explorer, expand the Desktop channel, and then navigate to and select the widget that you want to customize with CSS code.
2. On the **Skin** tab of the Properties pane, check the **Custom CSS** checkbox. Doing so opens a small CSS code editor.
3. Type or paste the CSS code that you want. The CSS code editor offers syntax

recommendations as you type.



Add Local HTML Content

With Kony Visualizer, you can leverage your existing web apps and other HTML content, incorporating it right into your project. Accessed through the Browser widget, such content is packaged locally within the completed app so that it is always available. You can also integrate it with Kony Fabric to make the most of Kony's powerful services integration.

The local HTML content that you add to your project can be used for a traditional web app, or embedded within a native app. If you want, you can use the local HTML content across all platforms and channels.

For more information, refer to the following topics:

Add Existing HTML Content to a Project

Create New HTML Content for a Project

Open a Resource's Folder

Add HTML Content to an App Screen

Preview HTML Content

Disable Live Rendering

Add Existing HTML Content to a Project

If you have existing web content, you can integrate it into your Kony Visualizer project.

To add existing HTML content to a project, do the following:

1. Using your computer's folder browser, navigate to the files and folders you want to add to your Kony Visualizer project.
2. In Kony Visualizer, on the Project Explorer, on the **Project** tab, expand the **Web** category. A folder displays called **Localfiles**.
3. Click the context menu arrow of the **Localfiles** folder, and then click **Resource Location**. The folder opens in a new window of your computer's folder browser.
4. Switch to the folder browser window that contains the files and folders you want to add to your Kony Visualizer project, select them, and then copy them.
5. Switch to the folder browser window for the **Localfiles** folder, and then paste the files you copied.
6. To see the added files reflected in Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer), click **Refresh**.

Alternately, you can import HTML content by clicking the context menu arrow of the **Localfiles** folder, clicking **Import File(s)**, navigating to the files and folders you want to import, selecting them, and then clicking **Open**.

Create New HTML Content for a Project

You can create different kinds of web-related files to your local HTML content, and then edit them within Kony Visualizer. The Intellisense capabilities of the Code Editor simplify the creation of code by recommending completed code based on the incomplete code you have typed.

To create new HTML content for a project, do the following:

1. On the Project Explorer, on the **Project** tab, expand the **Web** category. A folder displays called **Localfiles**.
2. Click the context menu arrow of the **Localfiles** folder, and then click any of the following options:
   New Folder
   New JS File
   New HTML File
   New CSS File
3. Enter a name for the new folder or file, and then click **OK**.

   In the case of a new file, it opens in the Code Editor, where you can begin coding it.

Open a Resource's Folder

All the web-based resources of your project reside in folders in your workspace. With Kony Visualizer, you can open the folder location of any resource from the Project Explorer.

To open a resource's folder, do the following.

- In the Project Explorer, click the context menu arrow of any resource, and then click **Resource Location**.

  The folder where that resource is located opens.

Add HTML Content to an App Screen

Once you have imported or added local HTML content to your project, you make it accessible within your app by associating it with a browser widget.

To add HTML content to an app screen, do the following:

1. Display on the Visualizer Canvas the browser widget that you want you want to associate your HTML content with.

2. On the Project Explorer, on the **Project** tab, expand the **Web** category, and then click **Localfiles**.
3. Locate the HTML file you want to associate with the browser widget, and then drag it onto the browser widget. The widget's Master Data property is updated to associate with the HTML file you selected.
4. On the Project Explorer, click the **Project** tab.
5. Locate and open the container widget in the form you want to add the HTML content to.
6. On the Library Explorer, on the **Widget** tab, under Advanced Widgets, click and drag a Browser widget onto the container widget.
7. In the Properties Editor, click the **Browser** tab, and then click the Master Data **Edit** button. The Master Data dialog box displays.

   > ***Note:*** You can also open the Master Data dialog box by double-clicking the browser widget.

8. Select **Local File**, click **Browse**, navigate to and select the HTML file you want, and then click **OK**. The path to the file is represented by a relative path to the HTML content file you want, where the **Localfile** folder is the root. For example, if the file you want is called Home.htm and is located in a folder called Source, the relative path would be as follows:
   `\Source\Home.htm`
9. Click **OK**.

Alternately, you can add HTML content to an app screen by doing the following:

1. On the Project Explorer, click the **Project** tab.
2. Locate and open the container widget in the form you want to add the HTML content to.
3. On the Library Explorer, on the **Widget** tab, under Advanced Widgets, click and drag a Browser widget onto the container widget.

4. In the Properties Editor, click the **Browser** tab, and then click the Master Data **Edit** button. The Master Data dialog box displays.

   > ***Note:*** You can also open the Master Data dialog box by double-click the browser widget.

5. Select **Local File**, click **Browse**, navigate to and select the HTML file you want, and then click **OK**. The path to the file is represented by a relative path to the HTML content file you want, where the **Localfile** folder is the root. For example, if the file you want is called Home.htm and is located in a folder called Source, the relative path would be as follows:
   `\Source\Home.htm`

6. Click **OK**.

Preview HTML Content

You can preview HTML content within Kony Visualizer, or using an external browser. Within Kony Visualizer, you can also use the side-by-side functionality to preview the HTML in one pane while viewing and editing the HTML code in the other pane.

To preview HTML content, do the following:

1. On the Project Explorer, on the **Project** tab, expand the **Web** category, expand the **Localfiles** folder, and then locate the HTML file you want to preview.
2. Click the file's context menu arrow, and then do one of the following:
   - Click **Open Preview**. The file displays within the Visualizer Canvas.
   - Click **Launch in external browser**. The file opens in a new instance of your computer's default web browser.

To preview HTML content side by side, do the following:

1. Enable the HTML preview feature. To do so, do the following, depending on whether you're using Kony Visualizer or Kony Visualizer Classic:

- **Kony Visualizer.** On the **Edit** menu, click **Preferences**. From the left pane of the dialog box, click **General**, and then set Enable HTML Preview to **Yes**. Click **Apply**.
- **Kony Visualizer Classic.** On the **Window** menu, click **Preferences**. From the left pane of the dialog box, click **Kony Visualizer**, and then set Enable HTML Preview to **Yes**. Click **Apply**, and then click **OK**.

2. On the Project Explorer, on the **Project** tab, expand the **Web** category, expand the **Localfiles** folder, and then click the HTML file you want, opening it as a tab in the Code Editor.
3. On the **Project** tab, click the HTML file's context menu arrow, and then click **Open Preview**. The file previews on the Visualizer Canvas, and a second pane opens displaying the HTML code. Changes you make to the HTML code and then save are reflected in the preview.

If you prefer not to enable the HTML Preview setting but still want to preview the HTML and view the HTML code side by side, you can manually preview the content.

To manually preview HTML content side by side, do the following:

1. On the Project Explorer, on the **Project** tab, expand the **Web** category, expand the **Localfiles** folder, and then click the HTML file you want, opening it as a tab in the Code Editor.
2. On the **Project** tab, click the HTML file's context menu arrow, and then click **Open Preview**. The file previews on the Visualizer Canvas.
3. On the **Window** menu, point to **Arrange**, and then click **Side By Side**. A new pane opens to the right of the existing pane.
4. From the pane on the left, drag the tab of the HTML file you opened onto the right pane. When the edges of the right pane glow blue, drop the file onto the pane.

   The preview of the HTML file appears on the left pane while the code of the HTML file appears on the right pane. Changes you make to the HTML code and then save are reflected in the preview.

Disable Live Rendering

Once you have associated a local HTML file with a browser widget, you may not want the contents of the HTML file to be perpetually rendering on the Visualizer Canvas, especially if the HTML content is complex and requires an inordinate amount of your computer's resources. In such a case, you can disable the browser widget's live rendering functionality.

To disable live rendering, do the following:

1. Right-click the browser widget for which you want to disable live rendering, and then click **Disable Preview**. The browser widget takes on a gray background, and only displays the name of the HTML file associated with it.
2. To re-enable live rendering, click the context menu arrow, and then click **Enable Preview**.

Add a Local Database to an App

With Kony Visualizer, you can bundle a SQLite database into an app so that the data in the database is always available locally on whatever device the app is installed on. This capability can be used to store something as simple as a collection of web site favorites to more complex data, such as a product catalog.

> ***Notes:***
> - Any file type can be added to the folder where the database resides.
> - You must ensure that the size of the database, in addition to all the other assets of your app, do not exceed the app size limit for a given platform's application store.

To add a SQLite database to an app, do the following:

1. Using your computer's folder browser, navigate to the files and folders you want to add to your Kony Visualizer project.

2. In Kony Visualizer, on the Project Explorer, on the **Assets** tab, expand the channel that you want to add the SQLite database to, whether Common (to be available to all channels), Desktop, Mobile, Tablet, or Watch.
3. Expand the environment that you want the SQLite database to be accessible to, whether Common (to be available to all environments), Native, or Web.
4. If you selected the Native environment, expand the platform that you want the SQLite database to be accessible to, whether Common (to be available to all platforms), Android, iOS, or Windows.

   Once the node is fully expanded, a **raw** folder is visible.
5. Click the context menu arrow of the **raw** folder, and then click **Resource Location**. The folder opens in a new window of your computer's folder browser.
6. Switch to the folder browser window that contains the files and folders you want to add to your Kony Visualizer project, select them, and then copy them.

   > ***Important:*** If you are copying the files to a **raw** folder for the Android platform, be aware that Android only supports file names that contain letters, numbers, periods (.), and underscores (_). The presence of other characters in a file name invalidate that file for inclusion in the compiled Android app.
7. Switch to the folder browser window for the **raw** folder, and then paste the files you copied.
8. To see the added files reflected in Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer), click **Refresh**.

Folder Paths for a SQLite Database

The folder path where you store a SQLite database in a Kony Visualizer project varies, depending on the channel, environment, and platform. Any file type can be added to these folders.

> ***Note:*** Files in the platform-specific folders whose file names are identical to files in the Common folder take precedence and override the Common folder versions of those files.

Common (located under the Mobile channel)
<WorkspaceName>\<ProjectName>\resources\mobile\common\raw

Common (located under the Tablet channel)
<WorkspaceName>\<ProjectName>\resources\tablet\common\raw

Android Mobile
<WorkspaceName>\<ProjectName>\resources\mobile\native\android\raw

Android Tablet
<WorkspaceName>\<ProjectName>\resources\tablet\native\androidtab\raw

iOS Mobile
<WorkspaceName>\<ProjectName>\resources\mobile\native\iphone\raw

iOS Tablet
<WorkspaceName>\<ProjectName>\resources\tablet\native\ipad\raw

Create Cordova Applications

Kony Visualizer supports the creation of Apache Cordova apps. Cordova is an open-source development framework for mobile applications that rely on JavaScript, HTML5, and CSS3 rather than the APIs that are specific to a given platform, such as iOS and Android. The layout of a Cordova app is rendered using web views rather than any platform's native UI framework, and yet unlike web apps, they're capable of accessing native APIs and are bundled as apps for publication. For these reasons, Cordova applications are hybrid in nature, being neither native to a particular platform, nor entirely web-based, and are capable of mixing native and web-based code snippets.

Kony Visualizer supports internationalization in Cordova applications.

> ***Important:***
> - For Android: maximum Cordova version supported is 7.1.0

- For Android: Adding Gradle to the global scope is a pre-requisite for building Cordova applications. For more information, refer to Gradle in Android Platform Guide.
- For Windows: You must integrate Gradle home into your machine by setting the PATH environment variable.
- For Mac: You can integrate Gradle into your Mac using any of the following procedures and verify the command "ls -l /usr/local/bin/gradle" points to the installed Gradle.
    a. Create a symbolic link to "/usr/local/bin/gradle" to the actual Gradle installation.
    b. Install Gradle using the command "brew install gradle".

Creating a Cordova app involves the following tasks:

- Enable Cordova
- Import or Create Cordova Assets
- Add a Cordova Browser Widget to a Form
- Add or Import Cordova Plugins
- Default Cordova Plugins for [[[Undefined variable MyVariables.KonyQuantumAppViewer]]]
- Best Practices for Creating Cordova Applications
- Manually Customize the Cordova-Generated Android Project
- Errors and Workarounds

Enable Cordova

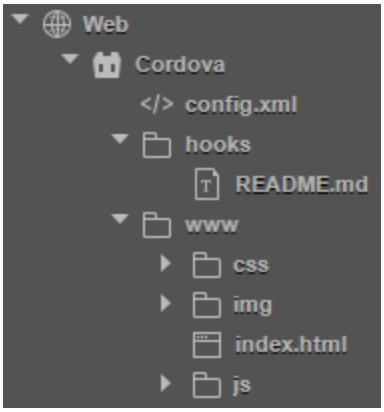
To enable Cordova, do the following:

1. On the **File** menu (the **Project** menu in [[[Undefined variable MyVariables.Kony QuantumViz]]]), click **Settings**.
2. On the Application tab, check the **Enable Cordova** checkbox.
3. By default, Kony Visualizer uses the version of Cordova that you have installed, as indicated by the checkbox **Use globally installed Cordova version** being checked. However, if you want to use a different version of the Cordova command line interface (CLI), clear the **Use globally installed Cordova version** checkbox, and then from the **Cordova CLI** drop-down list, select the version you want. After you close the dialog box, Kony Visualizer installs the version of Cordova that you selected. Kony Visualizer supports Cordova version 4.2 and up.

   > *Important:* Changing the Cordova CLI version may affect your application code.

4. Click **Finish** (**Apply** in [[[Undefined variable MyVariables.Kony QuantumViz]]]).

   Kony Visualizer creates a Cordova folder structure accessible on the Project tab of the Project Explorer, under the **Web** section.



Import or Create Cordova Assets

With Kony Visualizer, you can import existing Cordova content or create new content.

Import Cordova Assets

You can import an entire existing Cordova project into Kony Visualizer, or import one or more assets into individual Cordova folders.

To import an entire existing Cordova project into Kony Visualizer, do the following:

1. Using your computer's folder browser, navigate to the Cordova project that you want to import, and compress it into an archive (.zip) file.
2. In Kony Visualizer, on the **File** menu (the **Project** menu in [[[Undefined variable MyVariables.Kony QuantumViz]]]), click **Import Cordova Project**.
3. Navigate to the archive file that contains the Cordova project that you want to import, click it, and then click **Open**.

   Kony Visualizer imports the Cordova assets into the Cordova file structure within your Kony Visualizer project.

To import one or more assets into individual Cordova folders, do the following:

1. On the **Project** tab of the Project Explorer, expand the **Web** section, and then expand the **Cordova** folder.
2. Navigate to any of the folders within the **Cordova** folder, click that folder's context menu arrow, and then click **Import File(s)**. This command is also available from the context menu of the **Cordova** folder itself.
3. Navigate to the file or files you want to import, select them, and then click **Open**.

   The file or files you selected are imported into that folder.

Create Cordova Assets

To create Cordova assets, do the following:

1. On the **Project** tab of the Project Explorer, expand the **Web** section, and then expand the **Cordova** folder.
2. Navigate to the folder within the Cordova folder structure where you want to add the asset, click that folder's context menu arrow, hover over **New**, and then select one of the following options:
   Folder
   JS File
   HTML File
   CSS File

   In the case of a folder, a new folder is added. In the case of a file, a new file is opened in the code editor. You can rename the file or folder by clicking its context menu arrow in the Project Explorer, clicking **Rename**, and then entering the name you want.

Add a Cordova Browser Widget to a Form

To make Cordova content accessible to the user of your app, you add a Cordova browser widget to a form. For information about its properties, see Cordova Browser.

To add a Cordova browser widget to a form, do the following:

1. Open the form that you want to add the Cordova browser widget to.
2. From the **Widget** tab of the Library Explorer, under the Advanced Widgets section, drag a CordovaBrowser widget onto the form. The Cordova browser widget becomes the widget of focus on the Visualizer Canvas.
3. Associate an HTML file—typically Index.html for Cordova content—with the Cordova browser widget. To do so, on the **Project** tab of the Project Explorer, in the Cordova folder structure, locate the HTML file you want to associate with the browser widget—typically Index.html for Cordova content—and then drag it onto the Cordova browser widget. The widget's Master Data property is updated to associate with the HTML file you selected.

Add or Import Cordova Plugins

A Cordova plugin is a code module that extends Cordova's functionality beyond what is available in a purely web-based app, giving your Cordova context an interface to a device's native components and capabilities, such as the battery status or camera.

Cordova plugins can either be added or imported. When you add a plugin in Kony Visualizer, you are in fact fetching it from the Apache Cordova web site. Kony Visualizer searches the Cordova plugin catalog and lists the available plugins so that you can add the ones you want. In contrast, when you import a Cordova plugin into a Kony Visualizer project, you browse your computer for a plugin that you have already downloaded.

By default, a number of Cordova plugins are supported in [[[Undefined variable MyVariables.KonyQuantumAppViewer]]]. For more information, see Default Cordova Plugins for [[[Undefined variable MyVariables.KonyQuantumAppViewer]]].

To add Cordova plugins, do the following:

1. On the **Edit** menu, click **Manage Cordova Plugins**. The Manage Cordova Plugins dialog box displays, and lists any plugins that you have already added to the project.
2. Click **Add**. Kony Visualizer populates the dialog box with the Cordova plugins available on the Cordova Plugins page of the Apache Cordova web site.
3. In the Search text box, type keywords that help narrow the scope of the plugins listed.
4. Click the name of the plugin you want, and then click **Add**. The plugin is added to the list of plugins associated with your project.
5. Code your Cordova context to access the plugin. For more information, see *CordovaBrowser Widget* in the Kony Visualizer Widget Programmer's Guide.

To import Cordova plugins, do the following:

1. On the **Edit** menu, click **Manage Cordova Plugins**. The Manage Cordova Plugins dialog box displays, and lists any plugins that you have already added to the project.

2. Click **Import**.
3. Navigate to the location of the plugin, select it, and then click **OK**. The plugin is added to the list of plugins associated with your project.
4. Code your Cordova context to access the plugin. For more information, see *CordovaBrowser Widget* in the Kony Visualizer Widget Programmer's Guide.

Default Cordova Plugins for [[[Undefined variable MyVariables.KonyQuantumAppViewer]]]

Kony Visualizer by default supports the following plugins for [[[Undefined variable MyVariables.KonyQuantumAppViewer]]]:

Battery	Device	Network Information
BlueTooth LE	Device Motion	Notification
Camera	Device Orientation	Push
Capture	File	Splashscreen
CodePush	File Transfer	StatusBar
Console	Geolocation	Vibration
Contacts	Globalization	Whitelist
Cordova SQLite storage	InAppBrowser	
CrossWalk WebView engine	Media	

Best Practices for Creating Cordova Applications

To ensure the best performance and security, it is recommended that you adhere to the following best practices.

Invoking Kony Fabric Services from Cordova

With Kony Visualizer, a Cordova application can invoke Kony Fabric services. And if your application requires it, you can invoke Kony Fabric services from both a Cordova context and a native context.

If your Kony Visualizer project includes both native forms and Cordova forms, it is best to invoke Kony Fabric services from the native context, and then pass the required data to the Cordova browser widget.

Prevent the Execution of Unverified JavaScript Functions

To prevent the execution of any unverified JavaScript functions on a device, any functions that are invoked for use in the Browser or Cordova Browser widgets should exist in the underlying JavaScript files of your project.

Manually Customize the Cordova-Generated Android Project

While integrating your Cordova-generated Android project, certain gradle dependency conflicts or `android.support` to `androidx` conversion issues can arise. From Kony Visualizer V8 SP4 Fixpack 47, you can manually customize this Cordova project and then resolve these issues.

You can bundle this customized Cordova project by specifying the `cordovabuildmode` property as `incremental` in the `androidbuild.properties` file. For more information about the `cordovabuildmode` property, click here.

To manually customize your Cordova-generated Android project, follow these steps:

1. You must build your kony project at least once to get a temporary **cordova** folder, where the actual Cordova project is copied and built.
2. After the kony project is built once (which may fail due to unresolved `android.support` references or due to some other build conflicts), you must copy all the contents in the `<konyproject>\cordovatemp\platforms\android` folder to the `<konyproject>/cordovaprebuilt` folder.
   The Cordova project (which is, **<konyproject>/cordovaprebuilt**) is now available for customization.

> ***Note:*** If you face any `android.support` to `androidx`conflicts, you can manually change those references. Alternatively, with Android Studio 3.2 and later, you can migrate an existing project to AndroidX by selecting **Refactor** > **Migrate to AndroidX** from the menu bar. This action converts all `android.support` references to `androidx`.

3. To pick the customization from the new Cordova project, you must specify the value of the `cordovabuildmode` property as `incremental` in the `androidbuild.properties` file, which is located at: **<konyproject>/androidbuild.properties**.

> ***Note:*** If you do not specify the `cordovabuildmode` property, the `<konyproject>/cordovaprebuilt` folder will be ignored.

> ***Note:*** If you specify the `cordovabuildmode` property, the `<konyproject>/cordovaprebuilt` folder must be available; otherwise, the build will fail.

Errors and Workarounds

While developing Cordova supported applications in Kony Visualizer, you might see the following errors due to memory-related issues.

Execution failed for task ':mergeDebugResources'

Error: java.util.concurrent.ExecutionException: java.io.IOException: The pipe is being closed

Follow the steps below to fix the problem:

1. Navigate to the home directory of the user.
   You can navigate to the User home directory by running the command (Windows + R) `%userprofile%` on Windows machines and `~/` on Macs.

2. In the directory, create a new file with the name **gradle.properties**.

3. Copy the code below and paste it in the gradle.properties file.

```
org.gradle.jvmargs=-Xms512M -Xmx4g -XX:MaxPermSize=2048m -
XX:MaxMetaspaceSize=1g -Dkotlin.daemon.jvm.options="-Xmx1g"
```

4. Save and close the gradle.properties file.
5. Run the command `gradle -- stop` on your machine. This action stops the Gradle process and unlocks any locked files by Gradle.

Incorporate Amazon AWS Services

Applies to *Kony Visualizer Classic*.

With Kony Visualizer, you can download and import the Amazon AWS JS SDK into your project to integrate AWS cloud services into your app, which are available for the iOS, Android, and Windows platforms.

Amazon offers separate AWS JS SDKs for Node.js and browser-based implementation. Kony Visualizer uses the browser SDK.

After importing the Amazon AWS JS SDK into your project, you access the AWS APIs, enabling your app to interface with the following AWS services:

- **Amazon Cognito Identity.** For adding user sign-up and sign-in to your apps, including the capability to authenticate users through Facebook, Twitter, or Amazon, whether via SAML or another identity system.
- **Amazon Cognito Sync.** For synchronizing app-related user data across mobile devices and the web without the need for your own backend.

- **Amazon DynamoDB.** For storing, updating, and querying JSON documents.
- **Amazon Kinesis.** For loading and analyzing streaming data on AWS, and creating custom streaming apps for data.
- **Amazon AWS Lambda.** For administrating the running of code for just about any application or backend service without the overhead of having to provision and manage servers yourself.
- **Amazon Mobile Analytics.** For gathering app usage and revenue data, including measuring new versus returning users, tracking user retention, and monitoring custom in-app behavior events so that you can make data-driven business decisions.
- **Amazon Simple Storage Service (S3).** For backing up and archiving data.

Tasks Involved in Integrating AWS Cloud Services into Your App

Incorporating Amazon AWS services into your project involves the following tasks:

Enable AWS JS SDK Functionality

Download the AWS JS SDK

Import the AWS JS SDK into Your Project

Integrate your App with AWS Services

Enable AWS JS SDK Functionality

AWS JS SDK functionality is enabled in the Project Settings dialog box.

To enable AWS JS SDK functionality, do the following:

1. On the **File** menu, click **Settings**.
2. On the **Application** tab, check the **Enable AWS SDK** checkbox.
3. Click **Finish**.

Download the AWS JS SDK

Amazon offers separate AWS JS SDKs for Node.js and browser-based implementation. Kony Visualizer uses the browser SDK. You can download either the default build of the SDK, or build your own customized SDK to include services not available in the default SDK. For more information about building a custom SDK, see Building the SDK for Browsers in Amazon's AWS SDK for JavaScript Developer Guide.

To download the AWS JS SDK, do the following:

1. On the **Edit** menu, point to **Manage AWS SDK**, and then click **Download**. Doing so opens a browser that navigates to the Amazon AWS SDK for JavaScript in the Browser page.
2. In the Download portion of the page, click one of the following buttons:
    - **Download the default build.** Clicking this button leads to a version of the SDK that includes all the AWS services needed to support the cross-origin resource sharing (CORS) standard for accessing web resources on different domains. Next, under the Download section for the latest release of the SDK, click **browser.zip**, and then save the file to your computer.
    - **Customize your build.** Clicking this button leads to a version of the SDK that gives you the option of selecting additional services, such as `AWS.CloudSearch` and `AWS.AppStream`. Initially, this custom SDK includes all the services of the default SDK, but you can deselect any or all of the default services to be part of your custom SDK. Next, under the **Build Configuration** list of services, ensure the **Minified** option is selected, and then click **Build**. Your custom SDK is bundled as a .js file which, after a minute or so, you are prompted to save to your computer. For more information, see Building the SDK for Browsers in Amazon's AWS SDK for JavaScript Developer Guide.
3. Using your computer's file browser, navigate to the folder where the AWS JS SDK was saved. If you downloaded the default SDK as a .zip file, select it and extract its contents, which consist of two .js files: a minified and developer version of the SDK. For optimal app performance, the minified version is recommended.

Import the AWS JS SDK into Your Project

Once you have downloaded the Amazon AWS JS SDK, you import it into your product so that you can invoke and build its services into your app.

> ***Note:*** Importing the AWS JS SDK adds it only to your current project; it does not globally import the SDK so that it's available to all projects.

To import the AWS JS SDK into your project, do the following:

1. On the **Edit** menu, point to **Manage AWS SDK**, and then click **Import**.
2. Navigate to the folder where you downloaded the SDK.
3. If you downloaded and extracted the default version of the SDK, you will see two .js files listed: a minified version and a developer version. For better app performance, select the minified version (`aws-sdk.min.js`). If you downloaded a custom version, its file name uses the following naming convention, where `n` represents numbers: `aws-sdk-n.nn.n.min.js`.
4. Click **Open**. Kony Visualizer imports the AWS JS SDK into your project in the following folder:

   `<WorkSpaceName>\<ProjectName>\cloudssdks`

Integrate your App with AWS Services

Once you have imported the Amazon AWS JS SDK into your project, you can write code that accesses the AWS cloud services you want to use in your app. An example follows the upcoming procedure. The intellisense capabilities of the Kony Visualizer code editor assist you in using the correct AWS JS SDK syntax.

For more information on how to structure your code, see Working with Services in the SDK for JavaScript in Amazon's AWS SDK for JavaScript Developer Guide. To access Amazon's AWS API reference, see AWS SDK for JavaScript. For code examples, see SDK for JavaScript Code Examples.

> ***Important:*** Your code needs to reference the applicable AppKey and AppSecret.

To integrate AWS Services into your app, do the following:

1. In the Project Explorer, on the **Project** tab, hover over **Modules**, click its context menu arrow, and then click **New JS module**. A new file is opened in the code editor. You can rename the file by clicking its context menu arrow in the Project Explorer, clicking **Rename**, and then entering the name you want.
2. Write the code necessary to access the AWS APIs and invoke the AWS services you want to use. Ensure that your code makes reference to the applicable AppKey and AppSecret.

AWS Services Integration Example

The following code sample illustrates how you can use the Lambda API and invoke a Lambda function on AWS. For more code examples, see SDK for JavaScript Code Examples in Amazon's AWS SDK for JavaScript Developer Guide.

```
var awsDemo = (function(AWS, config) {
        AWS.config.update({
            accessKeyId: config.accessKeyId,
                secretAccessKey: config.secretAccessKey,
                httpOptions: {
                timeout: 1800000,
                      xhrAsync: true
            }
        });
        AWS.config.region = cpnfig.region;
        var dynamodb = new AWS.DynamoDB({
            apiVersion: '2012-08-10'
        });
        var cognitoidentity = new AWS.CognitoIdentity({
            apiVersion: '2014-06-30'
        });
        var cognitoidentityserviceprovider = new
AWS.CognitoIdentityServiceProvider();
        var cognitosync = new AWS.CognitoSync();
```

```
        var lambda = new AWS.Lambda();
        var bucket = new AWS.S3({
            apiVersion: '2006-03-01',
                                    params: {
                Bucket: config.Bucket
            }
        });

        function validateUser(userDetails, callback) {
            var params = {
                FunctionName: 'validateUser',
                     InvocationType: 'RequestResponse',
                     LogType: 'None',
                     Payload: JSON.stringify(userDetails)
            };
            lambda.invoke(params, function(err, data) {
                    if (err) kony.print("error in validation " + err);
//an error occurred     else {       var playLoad = JSON.parse
(data.Payload);       callback &amp;&amp; callback(playLoad);     }
    });  }});
```

Set App Lifecycle Events

The arc of activity from the time an app is launched and enters into a device's memory to when it is exited and released from memory is referred to as the app's lifecycle. Particular phases characterize the lifecycle of an app, such as the following (which are explained in greater detail later): Pre-initialization, Post-initialization, App Service, Deeplink, On View State Change, and On Search.

For any of these phases, you can design your app to launch action sequences, as your app requires. App lifecycle action sequences are supported for all four available app channels, namely Mobile, Tablet, Desktop, and Watch.

To set app lifecycle events, do the following:

1. In Kony Visualizer, open the project for which you want to set action sequences for the app's lifecycle.
2. From the **Project** tab of the **Project Explorer**, click either **Mobile**, **Tablet**, **Desktop**, or **Watch**. The Visualizer Canvas displays text indicating that you can define app life cycle events from the Properties Editor.
3. On the **App Events** tab of the Properties Editor, click the **Edit** button of the app lifecycle phase for which you want to create action sequences. Doing so opens the Action Editor. All four available app channels consist of the following app lifecycle phases:

    - **Pre Appinit.** Pre-initialization, which refers to that part of the app lifecycle after the app is launched but prior to it loading.
    - **Post Appinit.** Post-initialization, which refers to that part of the app lifecycle after the app has loaded, but prior to the displaying of data.
    - **App Service.** Refers to the launching of background services that the app relies upon.
    - **Deeplink.** Refers to creating services links in your app that enable third-party native and browser applications to connect to your app, providing a unified and seamless experience.

    Additionally, the Tablet channel includes the following two app lifecycle phases:

    - **On View State Change.** Refers to any time the state of the view changes between portrait and landscape, or between normal view and full screen.
    - **On Search.** Refers to any time a search is initiated.
4. Using the Action Editor, create the action sequences you want. For more information, see Add Actions.

Using Native Function APIs and Widgets

With Kony Visualizer, you can take advantage of native functionality for iOS and Android.

Click any of the following topics for more information.

Enable Native Function APIs

Add or Import Native Function APIs

Using the Native UI Container

Enable Native Function APIs

Applies to *Kony Visualizer Classic*.

With Kony Visualizer, you can use native function APIs for iOS and Android in your apps.

To enable native function APIs, do the following:

1. On the **File** menu, click **Settings**. Doing so opens the Project Settings dialog box.
2. Click the **Native** tab, and then do any of the following, depending on the platforms you're enabling native function APIs for:
    - Click the **iPhone/iPad/Watch** sub-tab, and then check the **Enable Native Function APIs** check box.
    - Click the **Android** sub-tab, and then in the Miscellaneous section, check the **Enable Native Function APIs** check box.

      > ***Important:*** If Kony Visualizer does not find any Android native function APIs in the project, functionality to support them is disabled, even if you enable it in the Project Settings dialog box. However, once you add or import Android native function APIs, Kony Visualizer automatically enables support for them. For information on adding or importing Android native function APIs, see Add or Import Native Function APIs.
3. Click **Finish**.

Add or Import Native Function APIs

In Kony Visualizer, you can add native function APIs for use with native UI widgets. These native UI widgets are platform-specific widgets to either iOS or Android, and can be used in an application by placing them in a native UI container widget called a NativeContainer. A native UI container can only be used for native UI widgets; it cannot contain any Kony cross-platform widgets. For more information, see Using the Native UI Container.

To add native function APIs, do the following:

1. On the **Edit** menu, click **Manage Native Function API(s)**. The Native Function Interface dialog box displays, and lists any native APIs you have already added or imported into the project.
2. Click the platform you want to download an API for, either iOS or Android.
3. Click **Add**. Kony Visualizer retrieves a list of native APIs available for the platform you chose.
4. In the Search text box, type keywords that help narrow the scope of the APIs listed.
5. From the list, select an API you want, and then click **Add**.
6. Repeat steps 3 through 5 until you have added all the native APIs you want.
7. When you are finished adding native APIs, close the dialog box by clicking the **X** in the upper right corner.

To import native function APIs, do the following:

1. On the **Edit** menu, click **Manage Native Function API**. The Native Function Interface dialog box displays, and lists any native APIs you have already added or imported into the project.
2. Click the platform you want to download an API for, either iOS or Android.
3. Click **Import**.
4. Navigate to the location of the native API, select it, and then click **OK**. The API is added to the list of native APIs associated with your project.

5. Repeat steps 3 and 4 until you have imported all the native APIs you want.
6. When you are finished importing native APIs, close the dialog box by clicking the **X** in the upper right corner.

Using the Native UI Container

Applies to *Kony Visualizer Classic*.

In Kony Visualizer, you can use native UI widgets in addition to the existing Kony cross-platform widgets. These native UI widgets are platform-specific widgets to either iOS or Android, and can be used in an application by placing them in a native UI container widget called a NativeContainer. A native UI container can only be used for native UI widgets; it cannot contain any Kony cross-platform widgets.

You can place a NativeContainer widget in a FlexContainer, FlexScrollContainer, component, or master. However, you cannot add it to the Tab and TabPane container widgets, nor can you add it to templates. Even a component or master containing a NativeContainer widget cannot be placed in a template.

Once it's added, to be able to see a NativeContainer widget listed in the widget library, you have to select that platform's canvas on the Visualizer Canvas. For example, the iOS NativeContainer will not display in the widget library if the platform of choice on the Visualizer Canvas is *Android: Native*. You'd need to select *Apple-iOS: Native*.

To add a NativeContainer or native UI widget to an app, you first must download it and add it to Kony Visualizer. The following procedure describes how to do so, along with how to activate the NativeContainer and native UI widget once it's downloaded. For more information about SSMs and NativeContainers, see the *Kony Visualizer API Developer's Guide*.

To learn how to use this widget programmatically, refer Kony Visualizer Widget guide.

Download and Activate the Assets for the Native Container

To download and activate the native function UI API assets for NativeContainer widgets, do the following:

1. Download the Native Function UI API you want from the Kony install site. They are located at the bottom of the page under the heading, *Other Downloads*.
2. On the **Project** menu of Kony Visualizer, click **Add JS Modules**. The **Add JS Modules** dialog box opens.
3. In the **Add JS Module** dialog box, click **Browse**, and then navigate and select an native function UI API zip file that you downloaded.
4. Click **Finish**. A message box appears indicating that the API was successfully added to your app. Click **OK**.
5. Repeat steps 2 through 4 for each API that you want to add.
6. Click the **Project Settings** icon. The **Project Settings** window opens with the **Self Sufficient Module** tab added. Click the **Self Sufficient Module** tab. The tab opens with the name of the native function API that you imported, displayed with a check box under the platform to which it is applicable.
7. Select the check boxes of the native function APIs that you want to use as widgets with the native UI container, and then click **Finish**.
8. On the Visualizer Canvas, select the platform of the SSM that you just added to Kony Visualizer.
9. In the widget library, scroll to the bottom. You will see a new category of widget called *NativeContainer*, and listed under it is the widget itself. You can now drag the native UI container onto your app and add the imported native UI widgets to it.

Add or Import Native Function APIs for the Native Container

To add native function APIs for NativeContainer widgets, do the following:

1. On the **Edit** menu, click **Add Native Function API**. The Native Function Interface dialog box displays, and lists any native APIs you have already added or imported into the project.
2. Click the platform you want to download an API for, either iOS or Android.
3. Click **Add**. Kony Visualizer retrieves a list of native APIs available for the platform you chose.

4. In the Search text box, type keywords that help narrow the scope of the APIs listed.
5. From the list, select an API you want, and then click **Add**.
6. Repeat steps 3 through 5 until you have added all the native APIs you want.
7. When you are finished adding native APIs, close the dialog box by clicking the **X** in the upper right corner.
8. On the Visualizer Canvas, select the platform that corresponds to the NativeContainer widget you want to add.
9. In the widget library, scroll to the bottom. You will see a new category of widget called *NativeContainer*, and listed under it is the widget itself. You can now drag the native UI container onto your app and associate the added native APIs to it.

To import native function APIs for NativeContainer widgets, do the following:

1. On the **Edit** menu, click **Add Native Function API**. The Native Function Interface dialog box displays, and lists any native APIs you have already added or imported into the project.
2. Click the platform you want to download an API for, either iOS or Android.
3. Click **Import**.
4. Navigate to the location of the native API, select it, and then click **OK**. The API is added to the list of native APIs associated with your project.
5. Repeat steps 3 and 4 until you have imported all the native APIs you want.
6. When you are finished importing native APIs, close the dialog box by clicking the **X** in the upper right corner.
7. On the Visualizer Canvas, select the platform that corresponds to the NativeContainer widget you want to add.

8. In the widget library, scroll to the bottom. You will see a new category of widget called *NativeContainer*, and listed under it is the widget itself. You can now drag the native UI container onto your app and associate the imported native APIs to it.

Add Actions by using Action Editor

After you have added widgets to a form, you can make the form and form elements functional by adding actions to them. To do so, select the form or widget with the action sequence, click the Action tab on the Properties pane, and then click the Edit button of the action you want. The action sequence opens in the Action Editor window.

The Action Editor window displays the Action ID of the Form and the Name of the action for which you are creating an action sequence. There are three types of views in the Action Editor by using which you can create and view Action Sequences. The three types of views are as follows:

- Diagram View
- Design View
- Code View

The Action Editor opens in the Diagram view, by default.

> ***Note:*** You can rename an action by editing the Action ID field.

Search for an Action in Action Editor

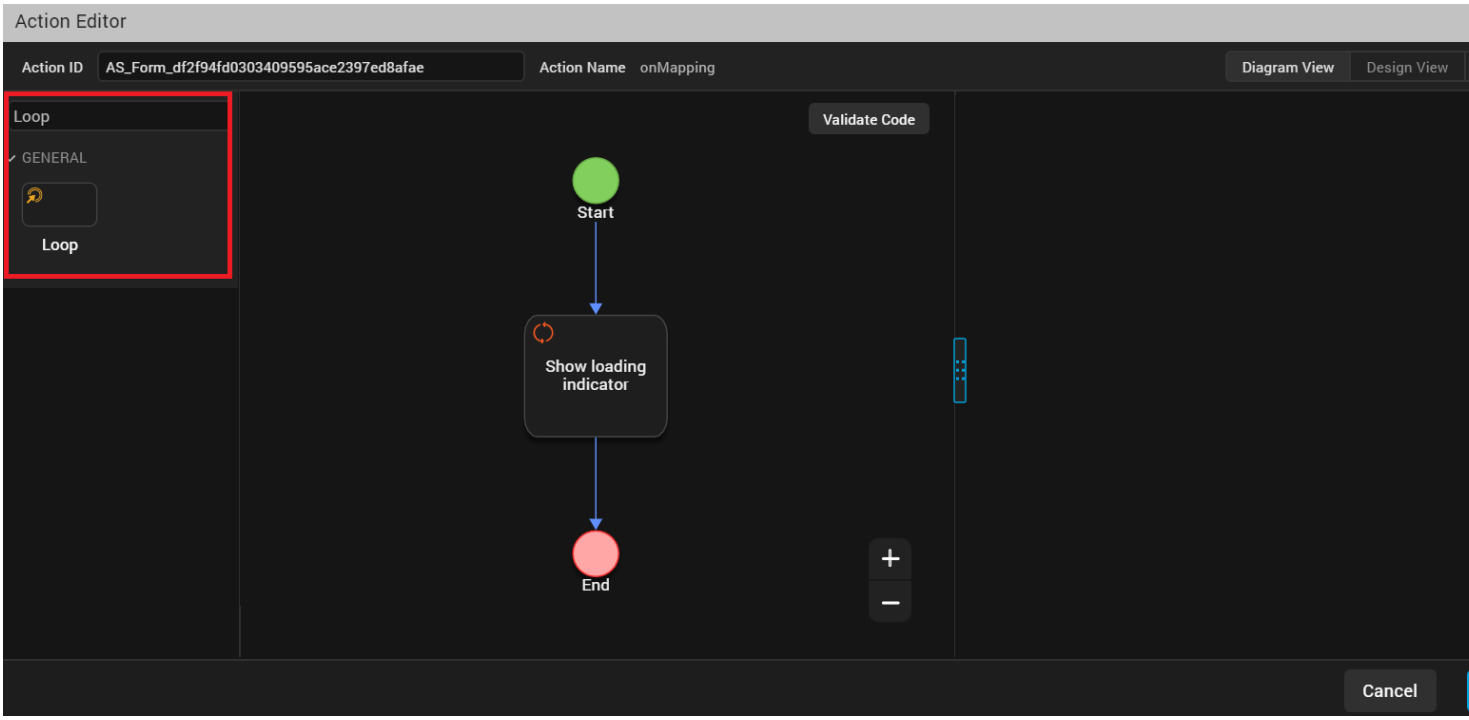
You can search for an action from the left pane of the Action Editor, as shown in the following images. The type of search that is available in the Action Editor is called as *Fuzzy Search*. So when you start typing in the Search box, filtered results that match the search query begin to appear.

In addition, you can search for actions by typing relevant tags in the Search box. For example, if you type *Code* in the Search box, the **Add Snippet** action appears as the search result. Currently, the search by tags functionality is limited to only these two tags: *Code* and *Conditional*.

Type the appropriate action tag name or start to type a specific action in the Search box, above the **General** action category. Only the actions that match the search query are displayed.

> ***Note:*** This feature is applicable in Diagram View and Design View, but not in Code View.

Here, when you type *Loop* in the Search box, the **Loop** action is displayed.



Here, when you type the *Conditional* tag, all the actions belonging to the **Condition** category are displayed.

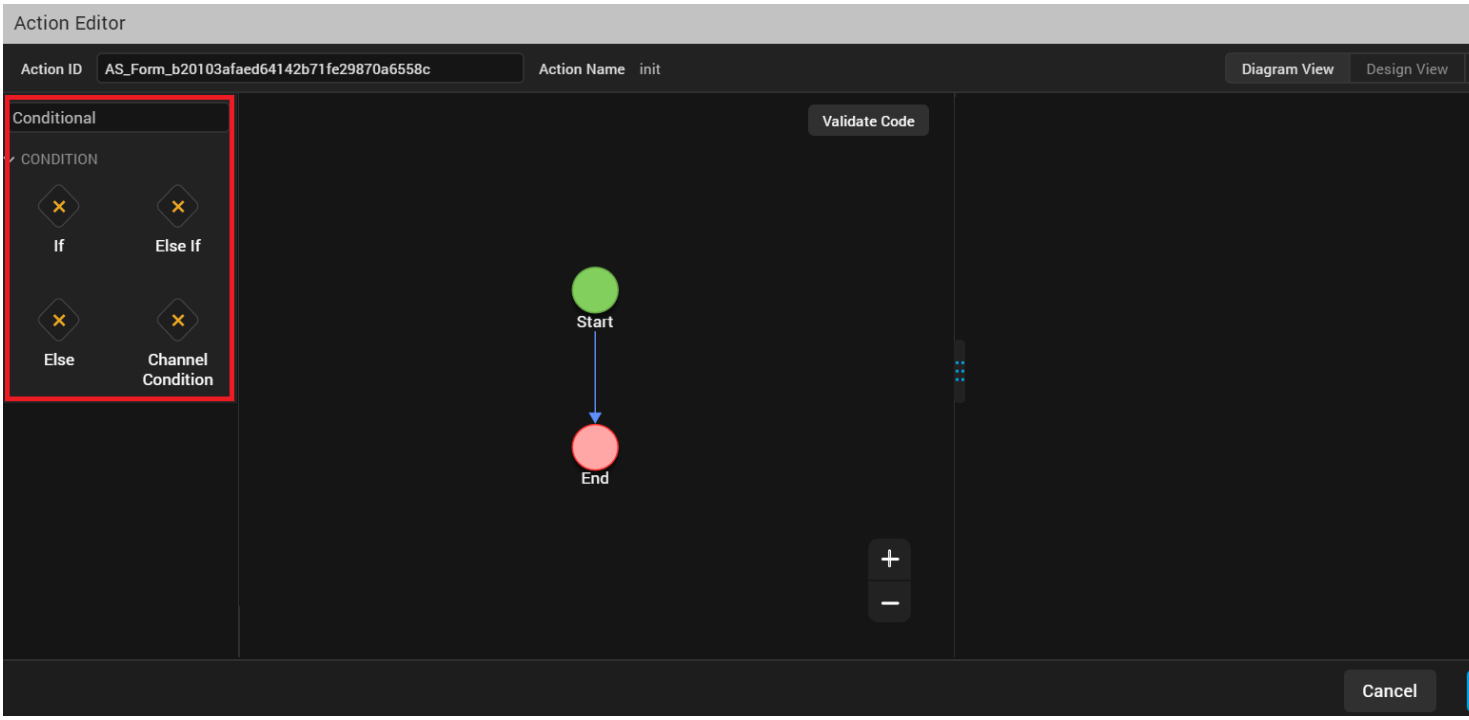
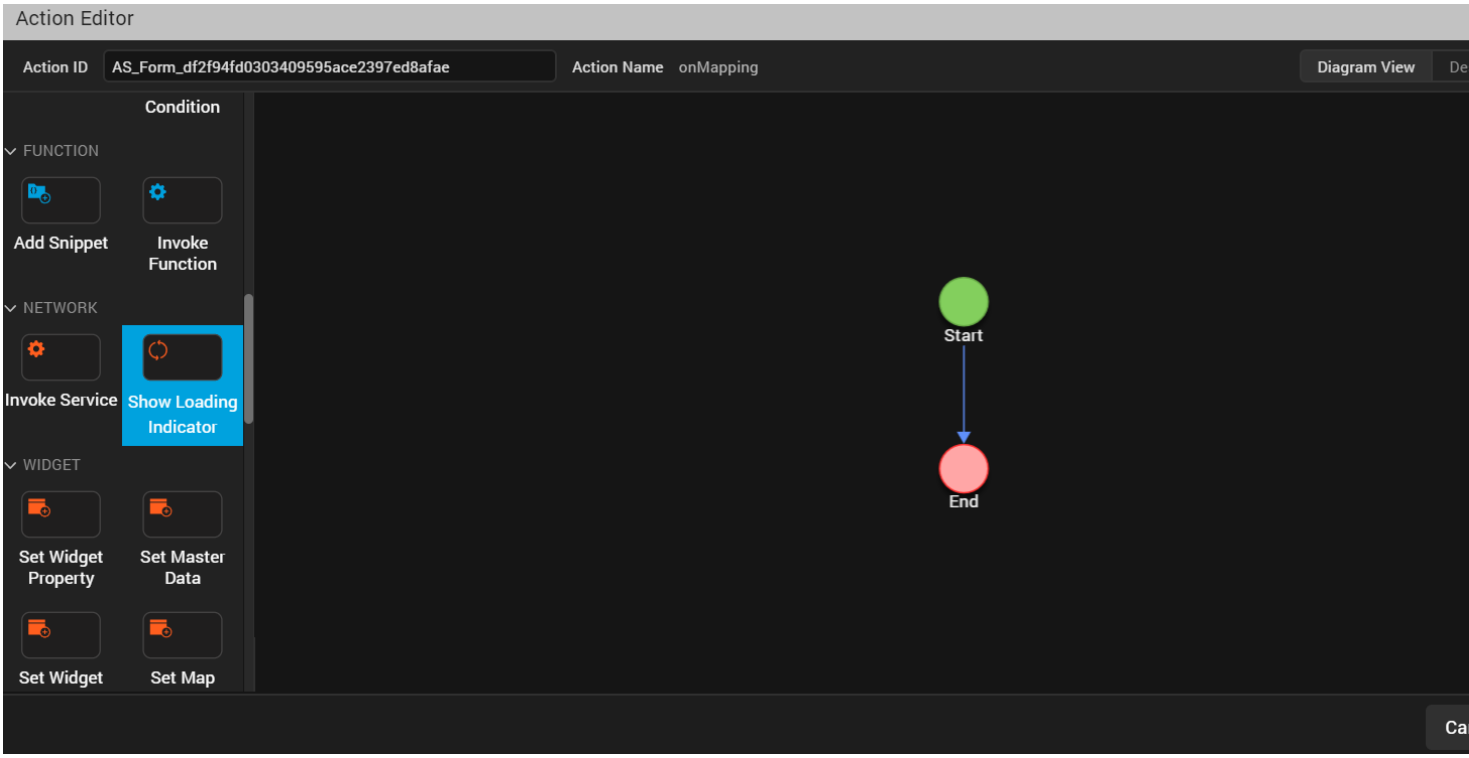


Diagram View

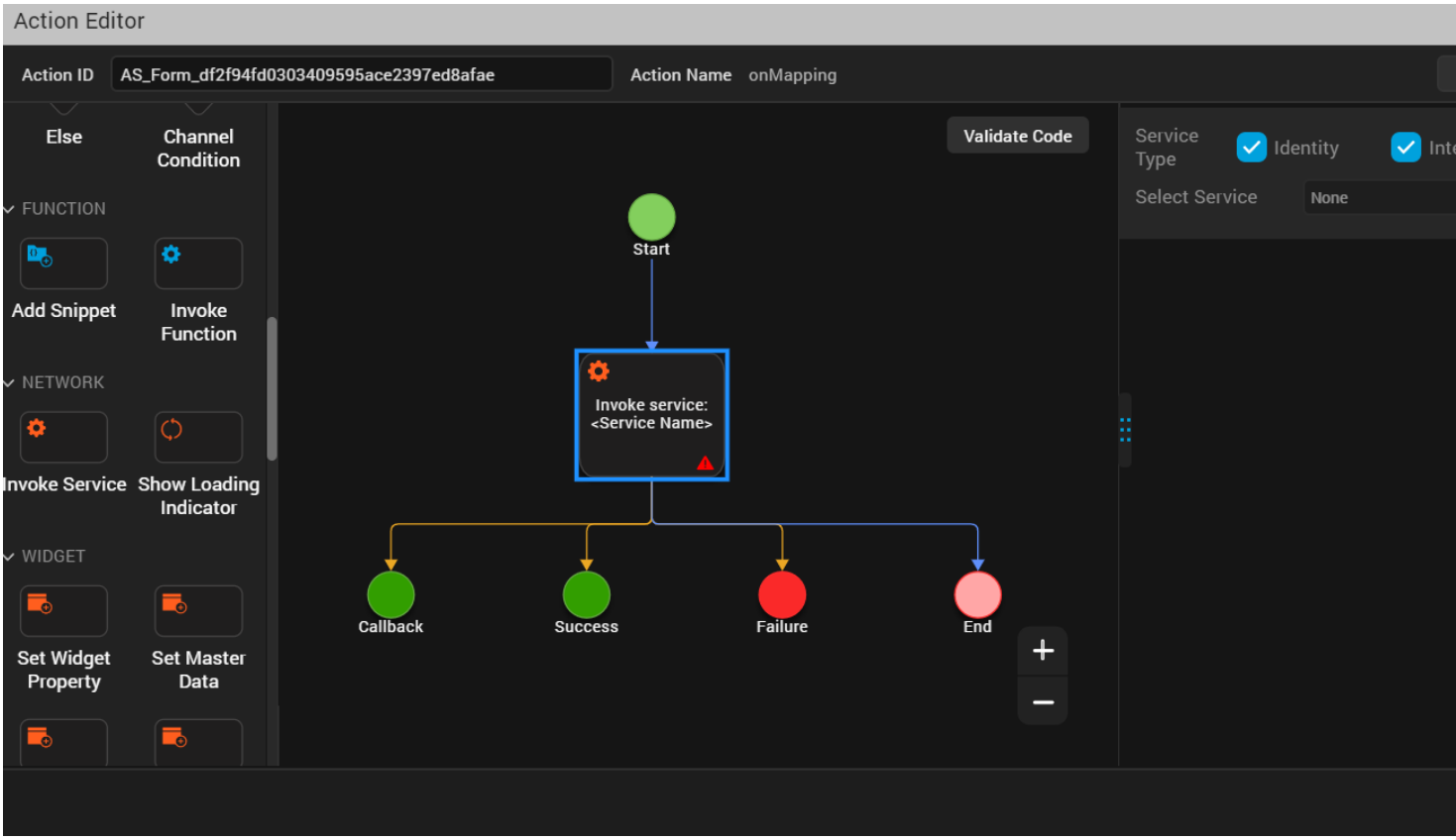
Using the Diagram View, you can quickly create and view the flow of an Action Sequence in a graphical representation. In the Diagram View, the Action Editor window is divided into three parts. The left pane contains a list of all the different actions that you can assign to a widget along with the icons associated with the action. The column in the middle contains a canvas that displays a flow diagram of the action sequence that you create. The right pane displays the properties and parameters that you can configure for the action that you select.



The Action Editor Canvas contains two nodes, by default; **Start** and **End**. To add an action in the Diagram View, from the left pane, either click the icon of the action you want to add in the action sequence or drag the icon of the action you want to add and drop it on the canvas. The action is added to the action sequence in between the Start and Stop nodes on the action editor canvas. After an action is added, the properties pane appears. For more information about the various actions available in Kony Visualizer, refer Categories of Actions.

> ***Note:*** The Properties Editor always displays the properties of the most recently selected widget. If you switch between the Diagram View and the Design View of an action sequence, regardless of what widget it is associated with, the Properties Editor will continue to display the properties of the most recently selected widget.

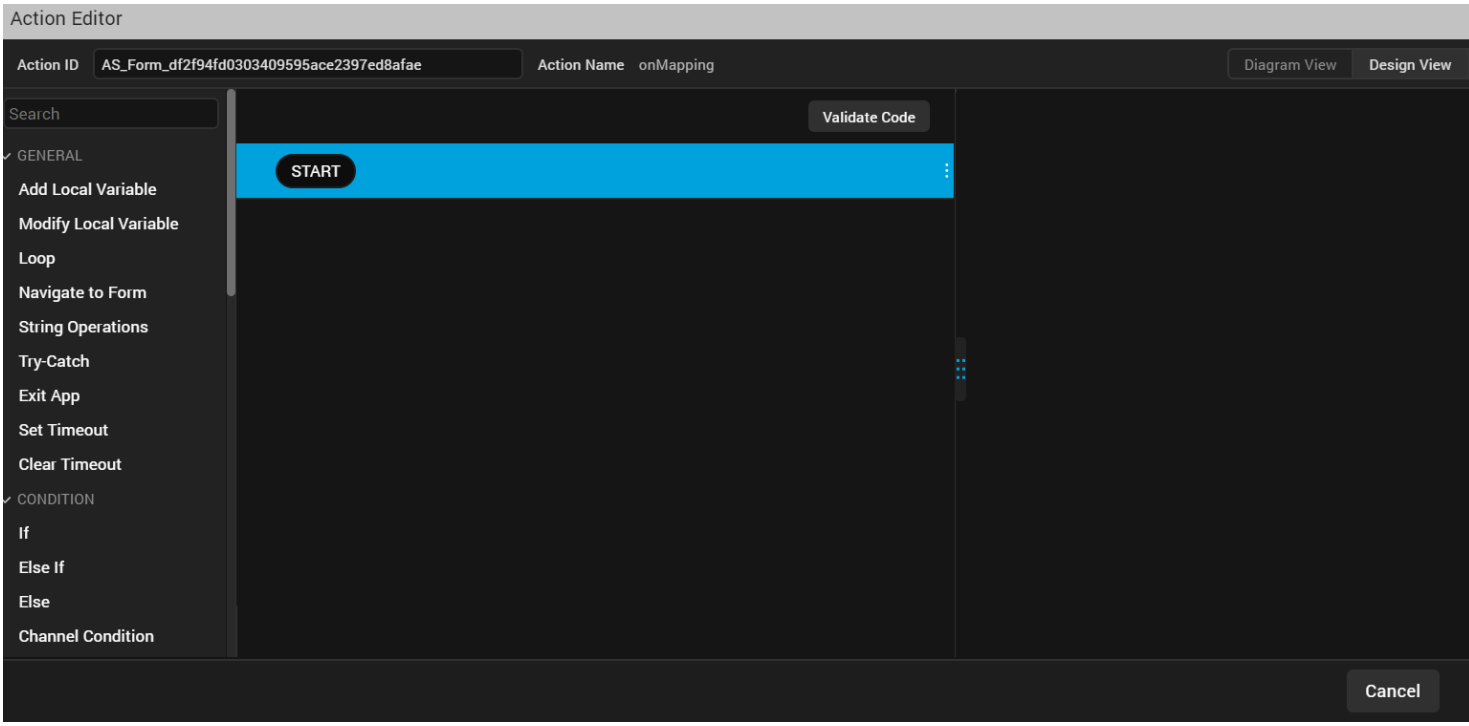
In Diagram View, the main flow and the nested flow are displayed in different colors. The main flow is displayed in blue color and the nested flow is displayed in orange color. This feature helps you to visually differentiate between the main flow and the nested flow in graphical view of the added actions, and helps you to easily understand the flow.



In Diagram View, you can click the **+** icon to zoom in to the graphical representation of the added actions and click the **-** icon to zoom out.

Design View

Using the Design View, you can create and view the flow of an Action Sequence. In the Design View, the Action Editor window is divided into three parts. The left pane contains a list of all the different actions that you can associate with a widget. The column in the middle contains a canvas that displays the action sequence that you create. The right pane displays the properties and parameters that you can configure for the action that you select.

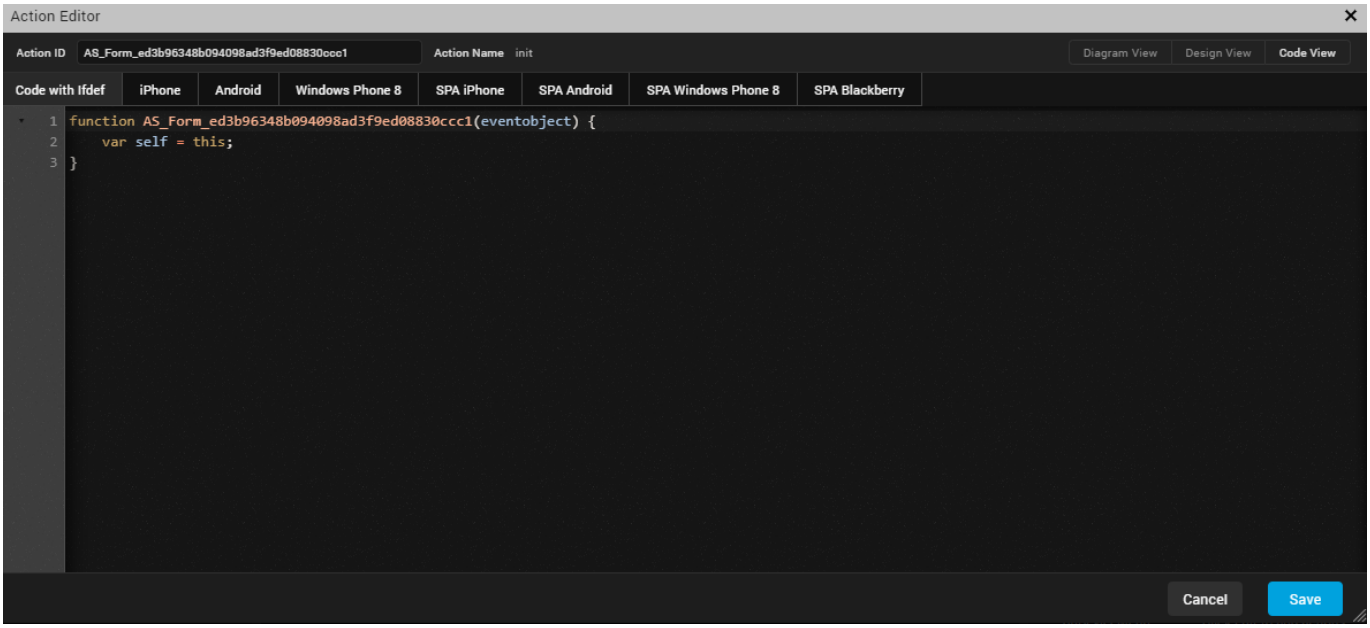


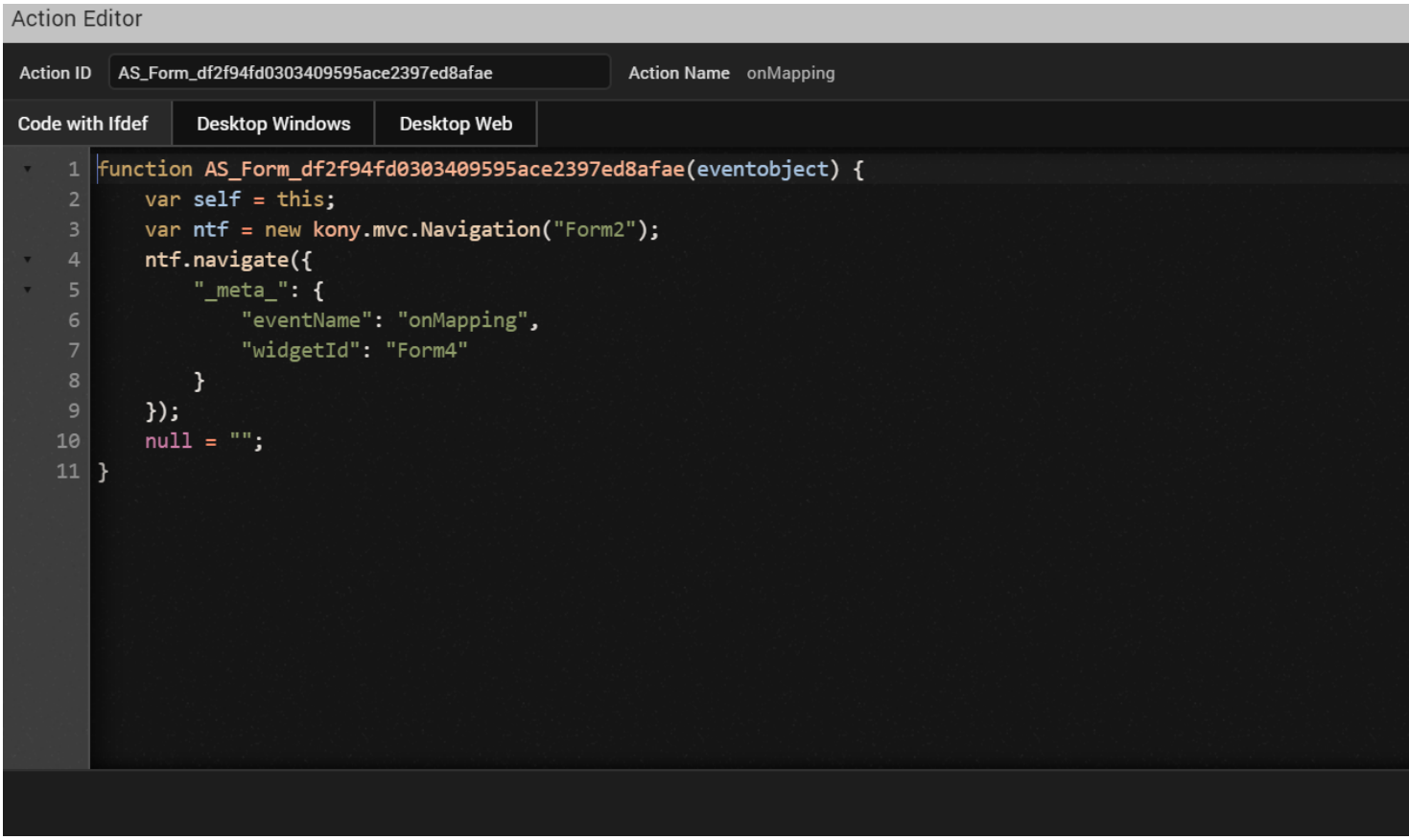
In Design View, the Action Editor Canvas contains a Start action, by default. To add an action in the action sequence, from the left pane, you can either click the action you want to add or drag an action and drop it on the canvas. The selected action is added to the action sequence after the Start action on the Action Editor canvas. After an action is added, the properties pane appears. For more information about the different actions available in Kony Visualizer, refer Categories of Actions.

> ***Note:*** The Properties Editor always displays the properties of the most recently selected widget. If you switch between the Diagram View and the Design View of an action sequence, regardless of what widget it is associated with, the Properties Editor will continue to display the properties of the most recently selected widget, and those properties will not be currently editable.

Code View

Once you have configured an action, Kony Visualizer generates a code snippet that depicts the action sequence that you can reuse in a JavaScript module.





Depending on the channel of the selected form, in the Code View window, you can view the code of the action sequence for the following platforms and channels:

- Code with Ifdef
- iPhone
- Android
- SPA iPhone
- SPA Android
- Desktop Windows
- Desktop Web

Categories of Actions

In Kony Visualizer, there are seven categories of actions that you can add to a widget. Click a category for instructions on how to add its actions to an action sequence.

General Actions: For adding navigation from one form to another, add and modify variables, add loop actions, perform String operations, execute a try-catch block, or closing an app.

Conditions Actions: For running an action based on certain conditions.

Network Actions: For invoking a loading indicator, an Object service, an Identity service, an Orchestration Service, or an Integration service.

Widgets Actions: For setting widget properties, updating master data, replacing a widget skin, or setting a map location.

Client Actions: For sending an SMS message, an email message, displaying an alert, opening a webpage, placing a phone call, set and get local storage, display, anchor, or dismiss a pop-up.

Functions Actions: For writing code snippets or reusing an existing action.

Animation Actions: For moving, scaling, rotating, or transforming a widget, or changing its background color.

Mapping Actions: For mapping (i.e. associating) the properties of one widget with another.

For the above actions, when you right-click on them in an action sequence, the following options are available:

- **Copy**: For copying an action.
- **Paste**: For pasting a copied action. Available when you copy an action
- **Delete**: For deleting an action.
- **Move Down**: For moving down an action in the sequence. Available when more than one action is defined.
- **Move Up**: For moving up an action in the sequence. Available when more than one action is defined.
- **Indent In:** For moving an action inside a condition action. Available when an action is below a condition action.
- **Indent Out:** For moving an action outside a condition action. Available when an action is below a condition action.

This topic also contains information related to the following subjects:

Important Considerations

Add Watch Actions

Validate the Code of an Action

Generate Code from an Action

Copy an Action Sequence

Disable an Action

List All Action Sequences in the Actions Node

Important Considerations

The following are limitations you should be aware of relative to actions:

- Animation actions are available only for flex widgets.
- The Action Editor does not check whether you have assigned a valid action or not. If you assign an invalid action, the app might not function as expected during functional preview.
- You cannot run the actions on the Visualizer Canvas.
- The Properties Editor always displays the properties of the most recently selected widget. If you switch to an action sequence or code, regardless of what widget it is associated with, the Properties Editor will continue to display the properties of the most recently selected widget, and those properties will not be currently editable.

> ***Important:*** If your project was originally created using Kony Visualizer 2.5 that was then imported into Kony Studio 6.5, and you import it into the latest version of Kony Visualizer, the action sequences are not imported.

Add Watch Actions

In Kony Visualizer, you add action sequences for Apple Watch as functions written in the Swift programming language; watchOS does not support JavaScript. For more information, see Swift on the Apple Developer site.

You can access all the Swift code snippets in your project from the Watch folder, located under Modules on the Project pane of the Project Explorer.

> ***Important:*** For projects with Watch action sequences written in JavaScript, created in Kony Visualizer 7.0, to continue working on such a project in Kony Visualizer 7.0.3 or later, those Watch action sequences will have to be rewritten in Swift. When you attempt to open such a project in

Kony Visualizer 7.0.3, a dialog box appears stating that continuing will cause all Watch action sequences with JavaScript to be dropped, and they will need to be rewritten. You can choose to continue, which opens the project and drops those Watch action sequences with JavaScript, or cancel, which doesn't open the project and leaves it unchanged.

To add a Swift code snippet, do the following:

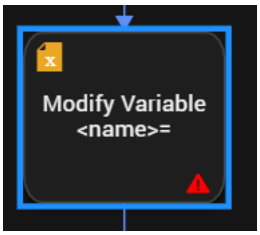
1. From the Project tab of the Project Explorer, navigate to the Watch channel, locate the form or widget you want to apply the action to, click its context menu arrow, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. Click **Add Swift Snippet** from the Functions section. The action is added to the action sequence and is the current action of focus.
3. In the bottom pane of the Action Editor, in the code editor, enter the Swift code that you want to add, or paste it in from another source by pressing **Ctrl+V**.
4. Save the action by pressing **Ctrl+S**.

Validate the Code of an Action

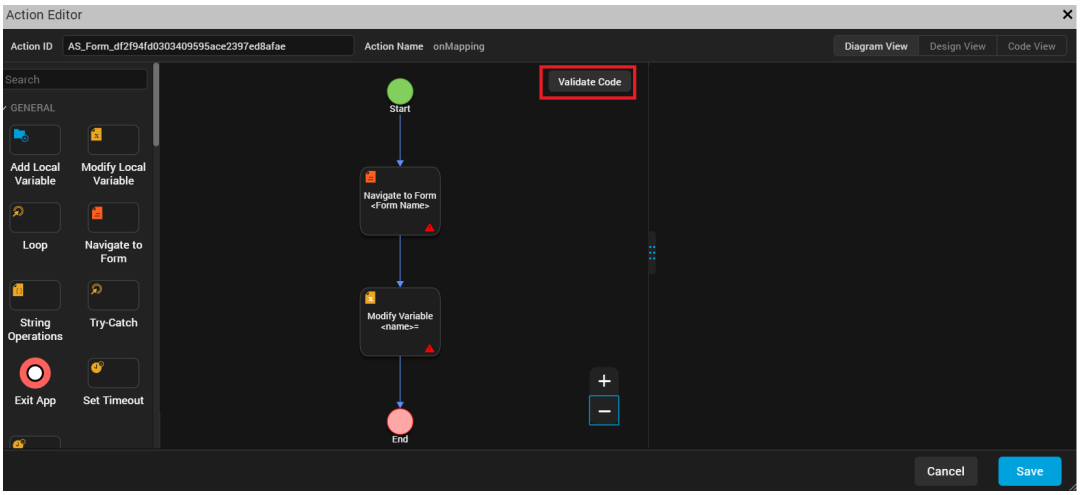
After you have configured an action, you can verify to ensure that the code of the action sequence is valid.

To validate the code of an action sequence, do the following:

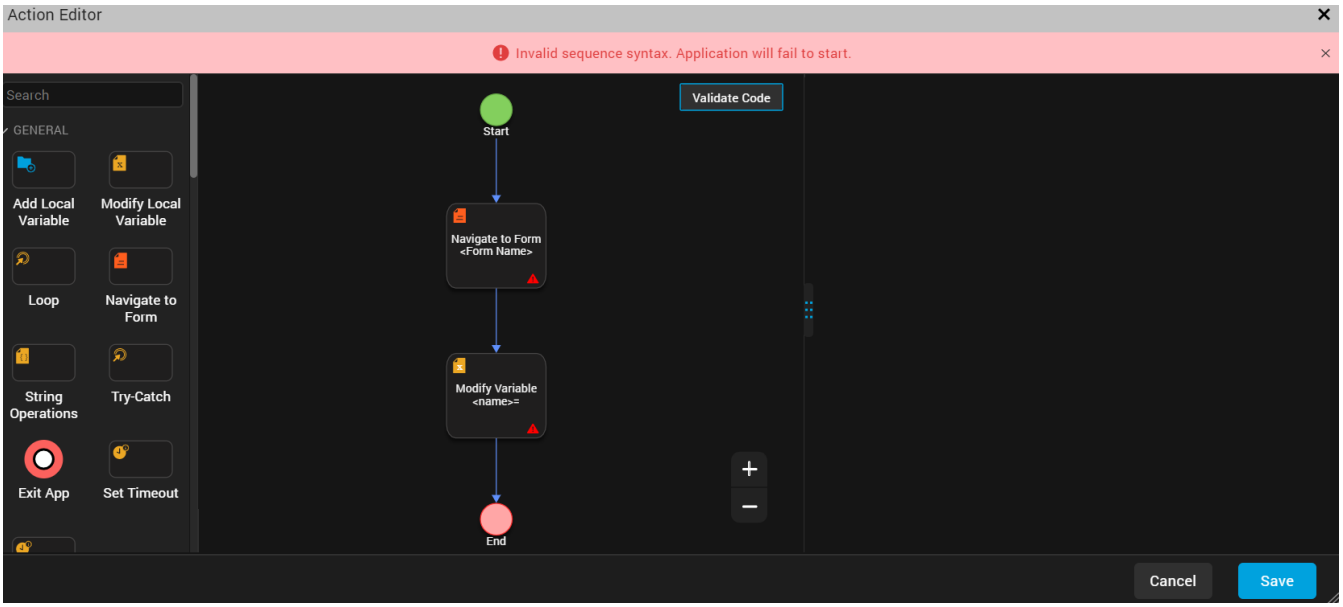
1. Open the action sequence that you want to validate. To do so, select the widget with the action sequence, click the Action tab on the Properties pane, and then click the **Edit** button of the action you want. The action sequence opens on the Visualizer Canvas.
2. In Design View and Diagram View, Kony Visualizer points out statements in the action sequence that are in error or are incomplete with a red information icon. Correct these statements.



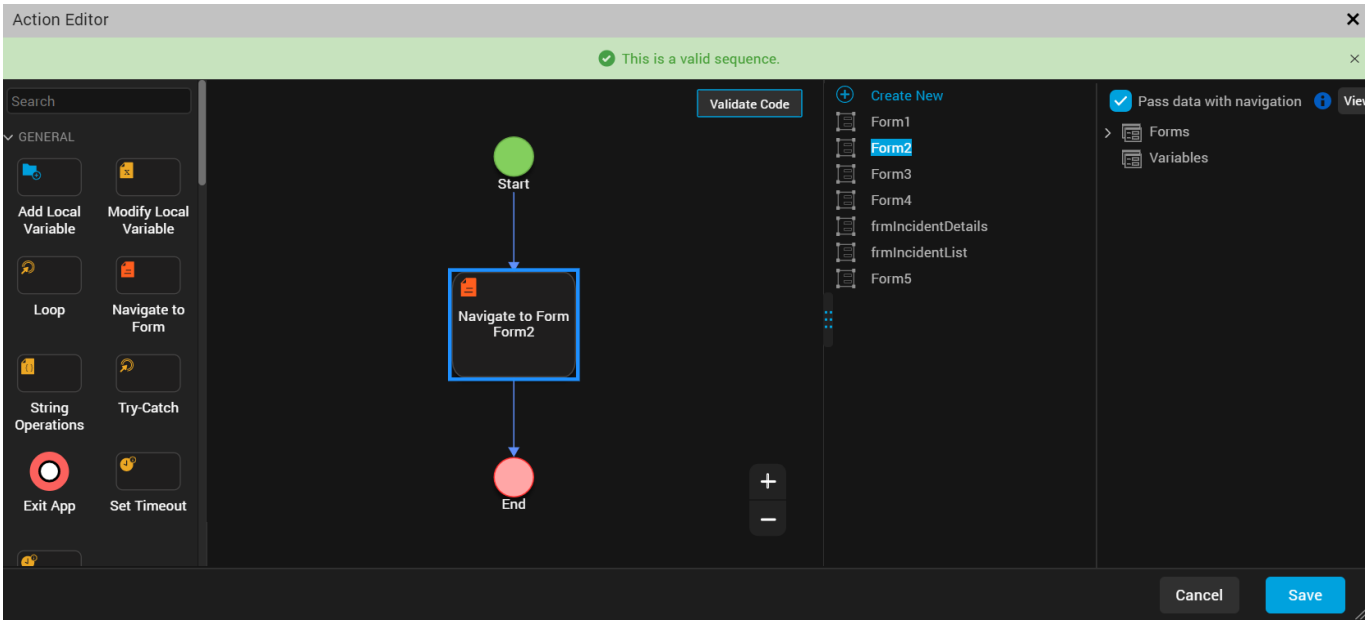
3. To validate the action sequence, in the Action Editor window, for Design View/Diagram View click **Validate Code**.



4. Kony Visualizer evaluates the action sequence. If the code is not valid, Kony Visualizer displays an appropriate error message. Follow the feedback of the validator and correct the sequence.



5. After you correctly revise the sequence, Kony Visualizer verifies the validity of the action sequence and displays an appropriate success message.



Copy an Action Sequence

You can copy an entire action sequence and apply it to another widget.

To copy an action sequence, do the following:

1. Open the action sequence you want to copy.
2. Right-click the top node name of the action sequence, and then click **Copy**.
3. Navigate to the widget you want to paste the action sequence to, right-click it, and then select the action that you want to paste to.
4. Right-click the top node name of the action sequence, and then click **Paste**.

Reuse an Action Sequence

To reuse the code snippet for the action sequence specific to a platform, do the following:

1. From the Code View window, select the tab for the platform and channel that you want.
2. Click in the pane to set the focus, and then select the code snippet by pressing **Ctrl+A**.
3. Copy the snippet by pressing **Ctrl+C**.

Delete an Action

You can delete unwanted actions from an action sequence.

To delete an action in an action sequence, do the following:

1. Open the action sequence that has the action or actions you want to disable.
2. Right-click the action you want to delete, and click **Delete**. The action is removed from the action sequence.

Disable an Action

You can disable an action to help you isolate trouble areas in your testing of action sequences.

To disable an action in an action sequence, do the following:

1. Open the action sequence that has the action or actions you want to disable.
2. Right-click the action you want to disable, and click **Disable**. The name of the action fades, indicating that it is disabled.

List All Action Sequences in the Actions Node

By default, the only action sequences that display in the Actions node of the Project tab in the Project Explorer are those actions that you create from within the Action node. In contrast, action sequences that you add to widgets do not appear in the Actions node. If you want, you can have all action sequences display in the Actions node.

To list all action sequences in the Actions node, do the following:

- Do one of the following, depending on the edition of Kony Visualizer you are using:
  - **Kony Visualizer.** On the **Edit** menu, click **Preferences**. Click **General** in the left pane, select **Show All Actions**, and then click **Apply**.
  - **Kony Visualizer Classic.** On the **Window** menu, click **Preferences**. Click **General** in the left pane, select **Show All Actions**, click **Apply**, and then click **OK**.

    > ***Important:*** You must click **Apply** before clicking **OK**. If you click **OK** without clicking **Apply**, your selection is not saved, and so your changes are not applied.

Invoke Sublime Text from Kony Visualizer

From V8 SP4, you can invoke the latest version of Sublime Text (a third-party source code editor) from within Kony Visualizer. This feature ensures that developers have the option to use an alternative tool to write and modify code within Visualizer, in addition to Visualizer's in-built code editor. You can also make use of the auto-complete Intellisense feature for Kony UI, API, and SDK functions while working with Sublime Text.

Sublime Text is a proprietary cross-platform source code editor with a Python API. This source code editor natively supports various programming and markup languages. You can use Sublime Text to add functions with plugins, which are typically community-built and maintained under free-software licenses.

Prerequisites

Before you can start using Sublime Text from Visualizer, you must meet the following prerequisites:

- Use Kony Visualizer V8 SP4
- Install the latest version of Sublime Text
- Install Package Control
- Install Ternjs for Sublime Text (to add Intellisense for Kony APIs)

Enable Sublime Text

After you have met the prerequisites mentioned previously, you must enable the feature to invoke Sublime Text from Visualizer.

> ***Note:*** By default, Visualizer will recognize Sublime Text if it is installed at a default location.

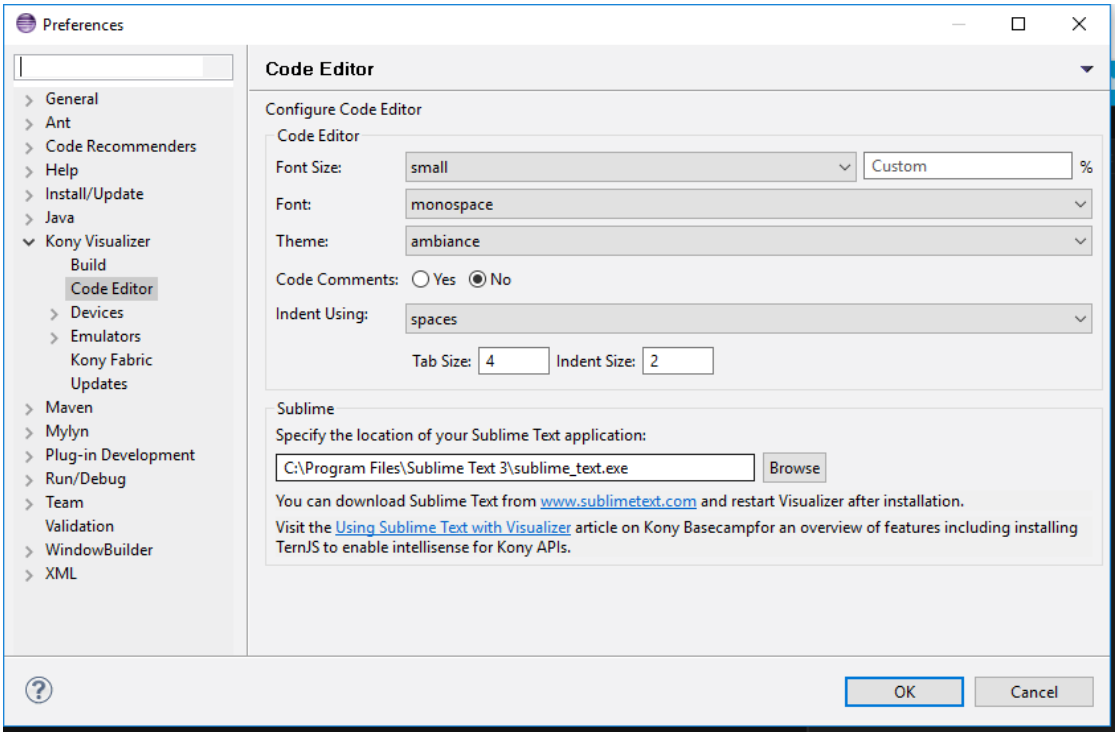
In Kony Visualizer Classic

To enable the feature in Kony Visualizer Classic, follow these steps:

1. In Kony Visualizer Classic, do the following:
   - **For Windows**: On the main menu, click **Window** and then click **Preferences**.
   - **For Mac**: On the main menu, click **Kony Visualizer** and then click **Preferences**.

   The **Preferences** window appears.
2. From the left pane, expand **Kony Visualizer** and click **Code Editor**. The Code Editor section appears.

3. Under the **Sublime** section, click **Browse** and select the absolute file path where you installed Sublime Text in your local system.
   For example, *C:\Program Files\Sublime Text 3\sublime_text.exe*.
   For more information, click the Using Sublime Text with Visualizer link. You will be navigated to the Kony Basecamp article on using Sublime Text.



4. Click **OK**. The Sublime Text feature is enabled.

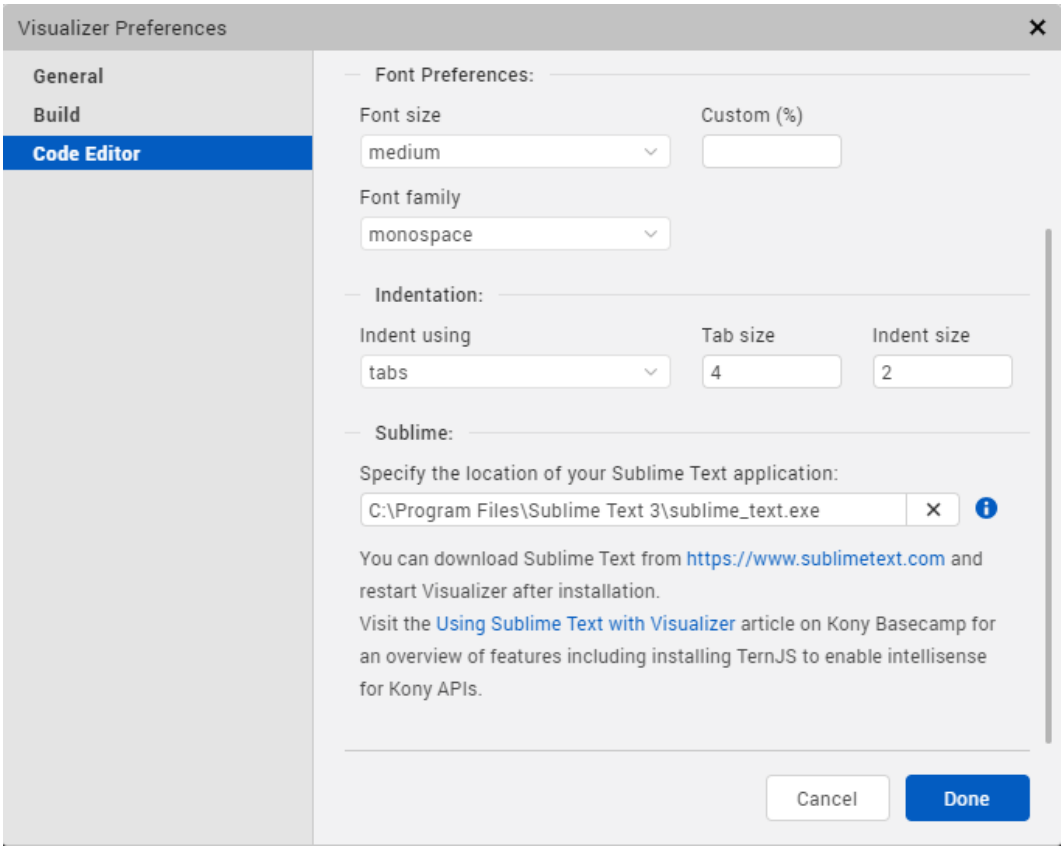
For Kony Visualizer

To enable the feature for Kony Visualizer, follow these steps:

1. In Kony Visualizer, do the following:
   - **For Windows**: On the main menu, click **Edit** and then click **Preferences**.
   - **For Mac**: On the main menu, click **Kony Visualizer** and then click **Preferences**.

The **Visualizer Preferences** window appears.

2. From the left pane, click **Code Editor**. The Code Editor section appears.
3. Under the **Sublime** section, click **Browse** and select the absolute file path where you installed Sublime Text in your local system.
   For example, *C:\Program Files\Sublime Text 3\sublime_text.exe*.
   For more information, click the Using Sublime Text with Visualizer link. You will be navigated to the Kony Basecamp article on using Sublime Text.



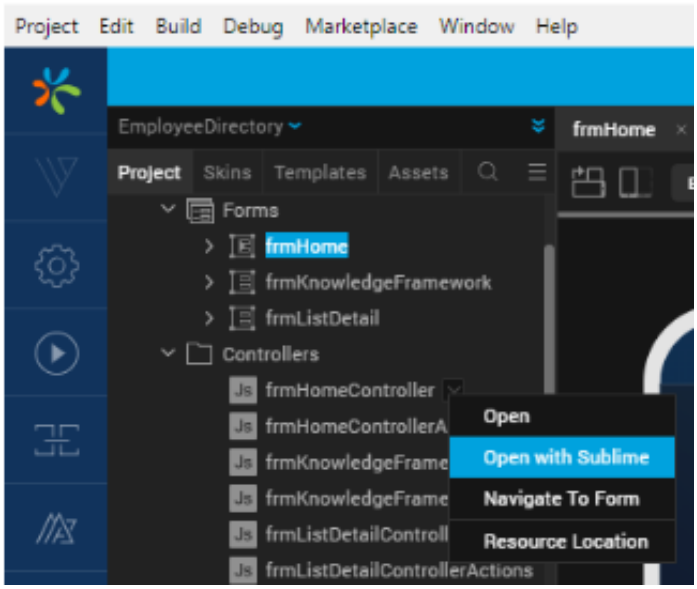
4. Click **Done**. The Sublime Text feature is enabled.

Open a JavaScript File with Sublime Text

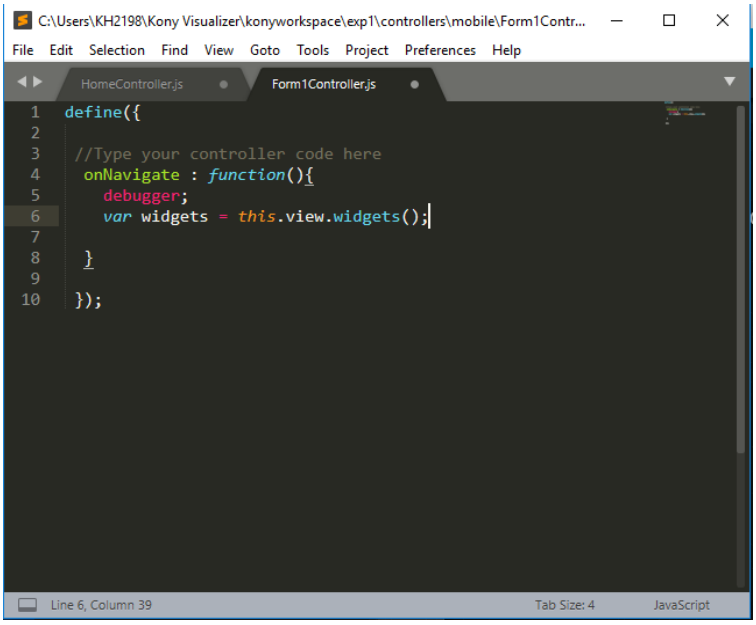
Now that you have enabled the required option, you can open any JavaScript file in Sublime Text from Visualizer.

To do so, follow these steps:

1. In your Kony Visualizer project, in the Project Explorer, right-click a JavaScript file/folder. A context menu appears.



2. Click **Open with Sublime**. The Sublime Text source code editor appears with the code snippet of that particular JavaScript file/folder.



You can make any changes to the code in any JavaScript file/folder by using Sublime Text.

Add Intellisense for Kony APIs

You can add the auto-complete Intellisense feature for Kony UI, API, and SDK functions while working with Sublime Text from Visualizer. You must install the Ternjs for Sublime package to enable this feature. Ternjs is a package for Sublime Text that provides JavaScript auto-fill intelligence.

To install Ternjs, follow these steps:

1. Install the Ternjs for Sublime package from GitHub.
2. Check out the following code into a sub-directory of your Sublime Text's Packages directory.

```
cd /path/to/sublime-text-N/Packages
git clone https://github.com/ternjs/tern_for_sublime.git
```

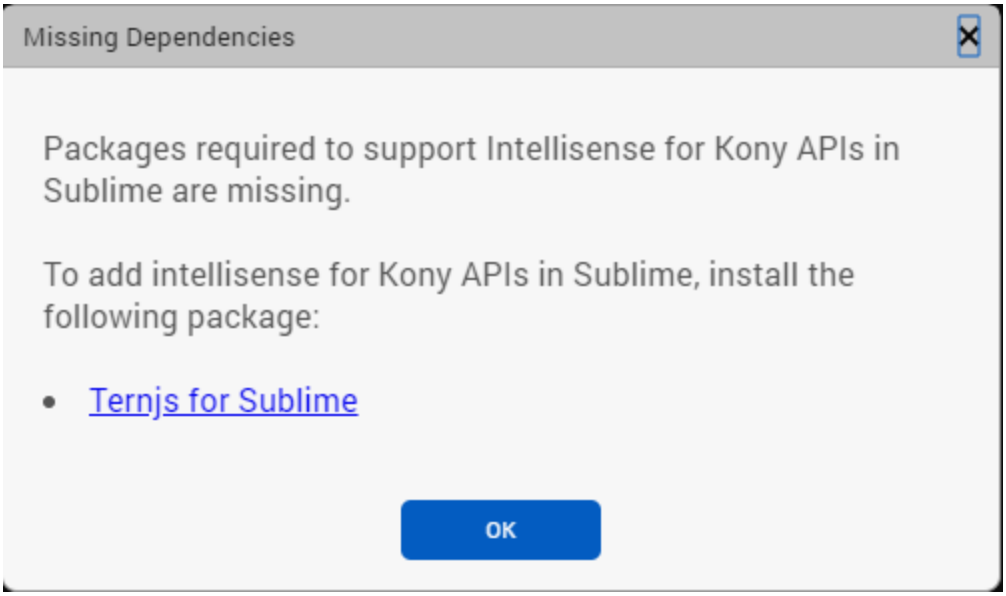
3. Ensure that you have installed node.js and npm (Tern is a JavaScript program), and then install the dependencies of the package.

```
cd tern_for_sublime
npm install
```

> ***Note:*** On OS X, you might also need to install the Fix Mac Path Sublime plugin to help Sublime Text to locate your node binary.

Once you successfully install the Ternjs for Sublime package, the Kony JavaScript auto-complete feature appears as you start typing in the Sublime Text editor.

> ***Note:*** If you have not installed the Ternjs for Sublime package, the **Missing Dependencies** window appears asking you to install that package.



Create a Condition Based on a Form's Orientation

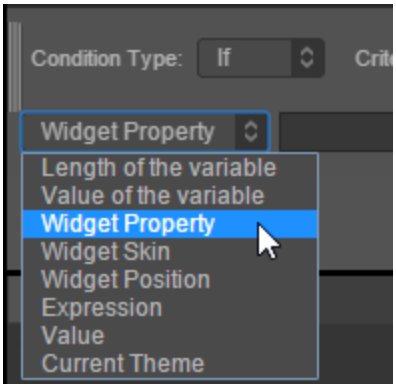
With Kony Visualizer, you can have an action sequence execute a conditional statement based on a form's orientation.

To create a conditional statement based on a form's orientation, do the following:

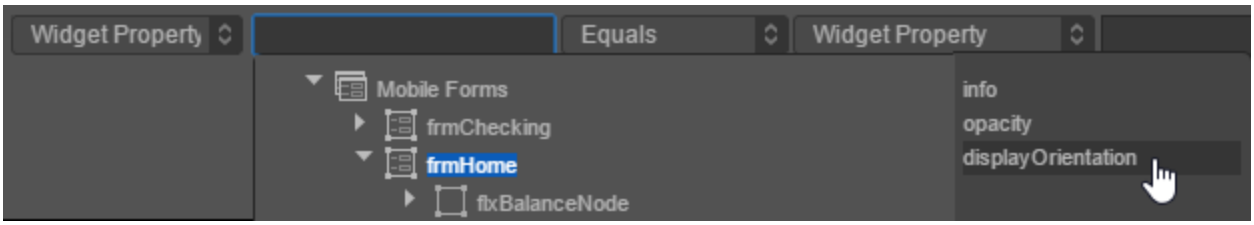
1. From the **Project** tab of the Project Explorer, select the form you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as postShow. Doing so opens the Action Editor and creates an action sequence for you to configure.

2. From the list of actions available along the left column of the Action Editor, click one of the four types of conditions from the Conditions section. The action is added to the action sequence and is the current action of focus.

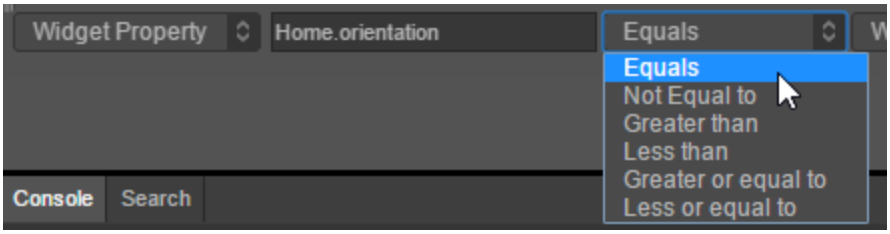
3. In the bottom pane of the Action Editor, from the leftmost drop-down list, set the condition to **Widget Property**.



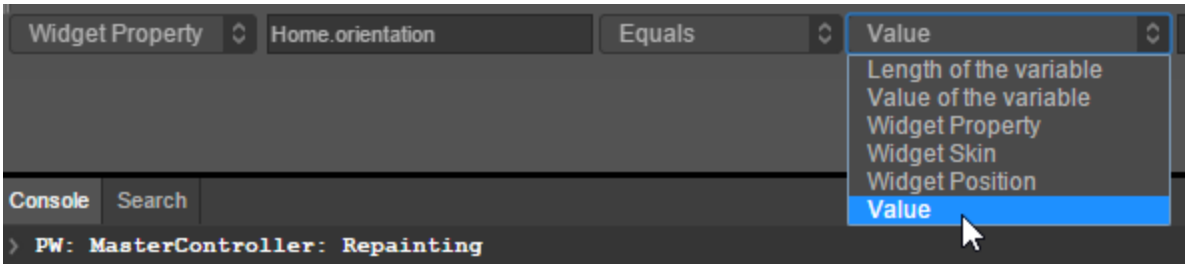
4. Click the next field over, and with the form being the widget of choice, click **displayOrientation**.



5. Set the operator you want, most likely either **Equals** or **Not Equal to**.

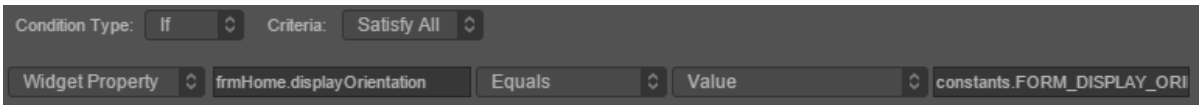


6. Set the second condition to **Value**.



7. In the last field, enter one of the following values, depending on whether the condition is portrait, landscape, or both:

```
constants.FORM_DISPLAY_ORIENTATION_PORTRAIT
constants.FORM_DISPLAY_ORIENTATION_LANDSCAPE
constants.FORM_DISPLAY_ORIENTATION_BOTH
```



8. Save the action sequence by pressing **Ctrl+S**.

General Actions

The general actions that you can use in an action sequence are as follows:

Action	Property
Loop Action	Repeat a set of nested statements for the specified number of times.
Add Local Variable	Adds a variable that is available within a function. Local variable can be of type Constant or Expression and take the following data type: String, Number, Boolean and or Collection.
Modify Variable	Modify an existing variable.

Action	Property
Navigate to Form	Navigates from one form to another.
String Operation	Perform operations (such as combine or split) on strings.
Try-Catch	Marks a block of statements to try, and specifies a response, should an exception be thrown.
Exit App	Closes the application.
Set Timeout	Set a timeout variable.
Clear Timeout	Clear the timeout.

Loop Action

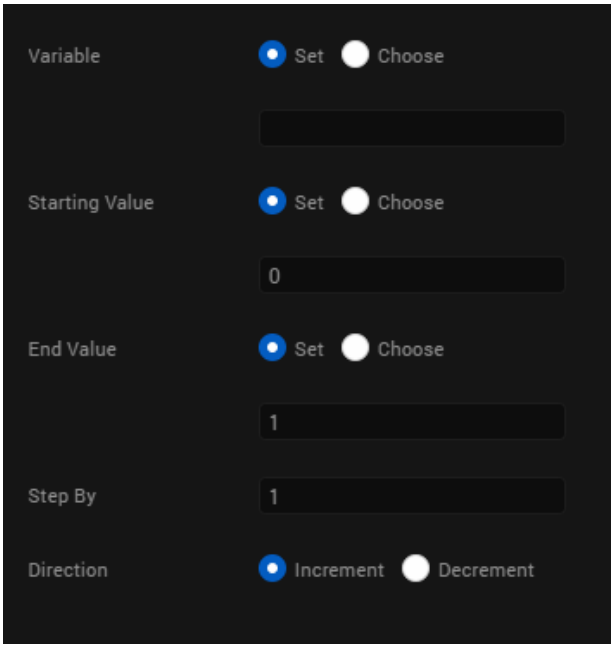
To define an action sequence for executing a loop action, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to. The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart. The Action Editor window opens and an action sequence is created.
3. From the **General** section along the left column of the Action Editor, click **Loop Action**. The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, configure the following
    a. In the **Variable** field, you can either select `set` and enter a name for the variable or select `choose` and select a variable from a list of existing variables.
    b. In the **Starting Value** field, you can either select `set` and enter a starting value for the variable or select `choose` and select a value from the list.

c. In the **End Value** field, you can either select `set` and enter an ending value for the variable or select `choose` and select a value from the list.

d. In the **Step By** field, enter the value for the step.

e. Select the direction of the count, either `Increment` or `Decrement`.



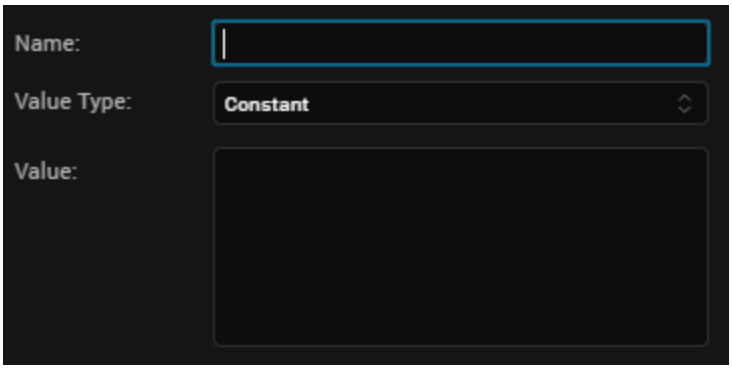
5. Click **Save** to save the action sequence, and then close the Action Editor window.

Add Local Variable

To define an action sequence to add a local variable, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to. The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart. The Action Editor window opens and an action sequence is created.
3. From the **General** section along the left column of the Action Editor, click **Add Local Variable**. The action is added to the action sequence and is the current action of focus.

4. In the properties pane of the Action Editor, in the **Name** text box, enter a name for the variable.
5. From the **Value Type** list, select a value type, either **Constant** or **Expression**.
6. In the **Value** field, enter the value of the local variable.



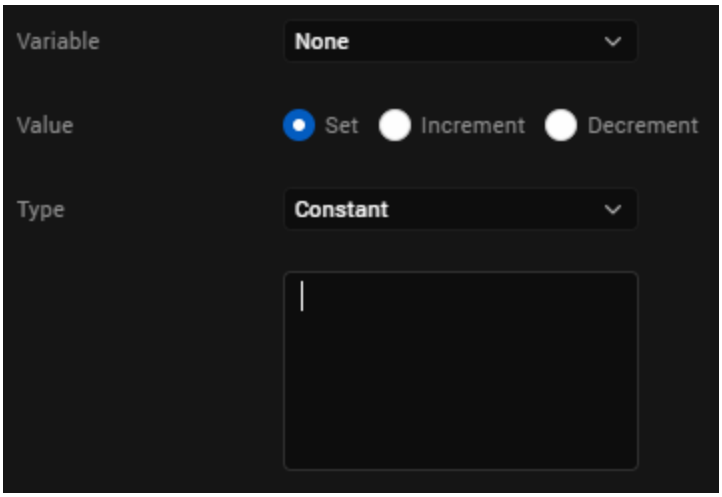
7. Click **Save** to save the action sequence, and then close the Action Editor window.

Modify Variable

To define an action sequence to modify an existing variable, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to.
   The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart.
   The Action Editor window opens and an action sequence is created.
3. From the **General** section along the left column of the Action Editor, click **Modify Variable**.
   The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, in the **Variable** field, select a variable from a list of existing variables.
5. In the **Value** field, you can choose to either `set` a value for the variable or `Increment` or `Decrement` the value of the variable.

If you choose to set a value for the variable, select a value type from the **Type** list, either **Constant** or **Expression**, and then enter the value of the variable in the text box provided below.



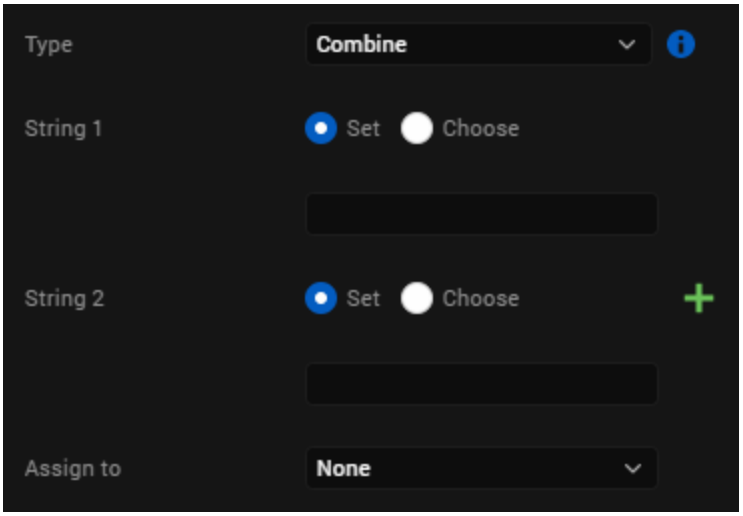
6. Click **Save** to save the action sequence, and then close the Action Editor window.

String Operation

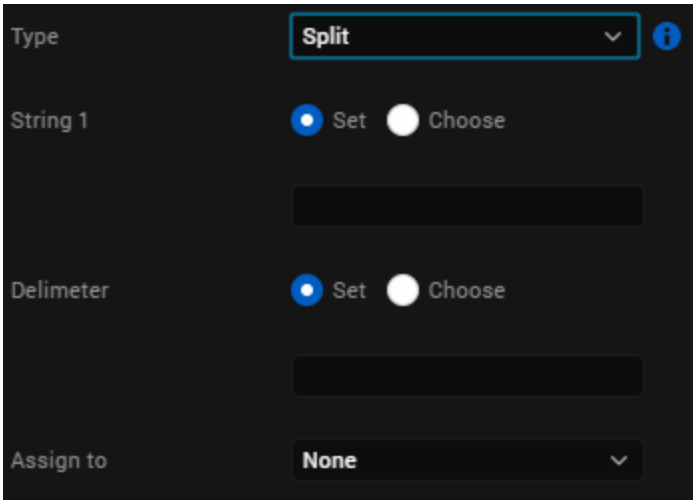
To define an action sequence to execute a String operation, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to. The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart. The Action Editor window opens and an action sequence is created.
3. From the **General** section along the left column of the Action Editor, click **String Operation**. The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, in the **Type** field, select the type of String operation, either **Combine** or **Split**.
5. In the **String 1** field, you can choose to either `set` a value for the string or select `choose` and select a string from a list of existing strings.

6. The second field varies depending on the type of String operation you select.
    - If you select the **Combine** option, in the **String 2** field, you can choose to either `set` a value for the string or select `choose` and select a string from a list of existing strings. To add additional strings, click the green + icon. To delete a string, click its corresponding red x icon.



    - If you select the **Split** option, in the **Delimiter** field, you can choose to either `set` a delimiter for the string or select `choose` and select a string from a list of existing strings.



7. In the **Assign To** list, select the variable to which you want to assign the resulting String.
8. Click **Save** to save the action sequence, and then close the Action Editor window.

Try-Catch

To define an action sequence to execute a try-catch block of code, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to. The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart. The Action Editor window opens and an action sequence is created.
3. From the **General** section along the left column of the Action Editor, click **Try-Catch**. The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, in the code editor, enter the code that you want to execute or paste it in from another source by pressing **Ctrl+V**.
    - The **try** statement allows you to define a block of code to be tested for errors while it is being executed.
    - The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block.
    - The **finally** statement lets you execute code, after try and catch, regardless of the result.
5. 

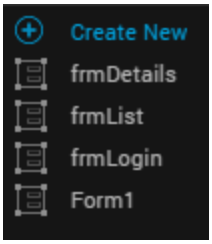
6. Click **Save** to save the action sequence, and then close the Action Editor window.

Navigation

To define an action sequence for navigating to a form, or exit an app, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to. The widget is highlighted on the Visualizer Canvas.

2. Right-click the widget and then select one of the actions for example, onTouchStart.
   The Action Editor window opens and an action sequence is created.
3. From the **General** section along the left column of the Action Editor, click one of the two navigation actions.
   The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, select the **Form** that the navigational action applies to (this step isn't used for the Exit App action).



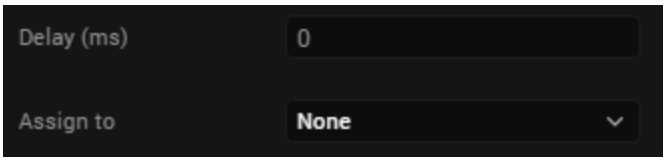
5. Click **Save** to save the action sequence, and then close the Action Editor window.

Set Timeout

To define an action sequence to set a timeout, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to.
   The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart.
   The Action Editor window opens and an action sequence is created.
3. From the **Client** section along the left column of the Action Editor, click **Set Timeout**.
   The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, in the **Delay** field, enter a value for the time in milliseconds (ms) after which the timeout must occur.

5. In the **Assign To** list, select the variable that you want to use to set the timeout.



6. Click **Save** to save the action sequence, and then close the Action Editor window.

Clear Timeout

To define an action sequence to clear a timeout that has already been set, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to.
   The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart.
   The Action Editor window opens and an action sequence is created.
3. From the **Client** section along the left column of the Action Editor, click **ClearTimeout**.
   The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, from the **Timeout Handle** list, select the variable that you want to use to clear the timeout.

5. Click **Save** to save the action sequence, and then close the Action Editor window.

Condition Actions

The behaviors that you can assign are as follows:

Action	Property
If Condition	Specifies an action to be executed when one or more conditions are satisfied.

Action	Property
Else If Condition	Specifies an action to be executed when one or more conditions (in addition to the **If** condition) are satisfied.
Else Condition	Specifies an action to be executed when an **If** or **Else If** condition is not satisfied.
Channel Condition	Specifies for which channels an action is executed.

To add conditional behavior (if condition, or elseif condition, or else condition, or channel condition) to an action sequence, do the following:

1. From the **Project** tab of the Project Explorer, select the widget you want to apply the action to. The widget is highlighted on the Visualizer Canvas.
2. Right-click the widget and then select one of the actions for example, onTouchStart. The Action Editor window opens and an action sequence is created.
3. From the **Condition** section along the left column of the Action Editor, click one of the four types of conditions. The action is added to the action sequence and is the current action of focus.
4. In the properties pane of the Action Editor, set the conditions that you want to be met. To add additional conditions, click the green + icon. To delete a condition, click its corresponding red x icon.
5. Add the action or actions that you want to carry out when the condition is met by locating and clicking the action from the list of actions available along the left column of the Action Editor. To subordinate the action under the conditional statement, right-click the action you just added, and then click **Indent In**.
6. Repeat steps 2 through 4 for any other general behavior that you want to associate with the

action sequence.

7. Click **Save** to save the action sequence, and then close the Action Editor window.

Functions Actions

From the Functions section of the list of actions available along the left column of the Action Editor, you can add any of the following four actions. Click an action for instructions on how to add it to an action sequence.

Action	Description
Add Snippet	Opens the code editor where you can write or paste a code snippet for use within a function.
Add Swift Snippet	Available only for the Watch channel. Opens the code editor where you can write or paste a Swift code snippet for use within a function. To create action sequences for watchOS, you must use Swift; watchOS does not support JavaScript.
Call Actions	Calls the action of your choosing from a list of available actions. To use this feature, you have to have already set up at least one action in the Project Explorer, on the Project tab, under the Actions section.
Invoke Function	Invokes the function of your choosing from a list of available functions.

Add Snippet

To add a code snippet, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action

sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.

2. From the list of actions available along the left column of the Action Editor, click **Add Snippet** from the Functions section. The action is added to the action sequence and is the current action of focus.

3. In the properties pane of the Action Editor, in the code editor, enter the code that you want to add, or paste it in from another source by pressing **Ctrl+V**.

4. Save the action by pressing **Ctrl+S**.

Call Action

To use this feature, you have to have already set up at least one action in the Project Explorer, on the Project tab, under the Actions section.

Once you have configured the Call Action function, you can navigate to and edit the source action that the function is calling.

To call an action, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.

2. From the list of actions available along the left column of the Action Editor, click **Call Action** from the Functions section. The action is added to the action sequence and is the current action of focus.

3. In the bottom pane of the Action Editor, from the Action Type drop-down list box, select the action you want to call.

4. Save the action by pressing **Ctrl+S**.

To edit the source action that the function is calling, do the following:

1. Open the action sequence that has the Call Action function.
2. Right-click the Call Action function, and then click **Edit Source Action**. The action opens in the Action Editor.

Invoke Function

With the Invoke Function action, you choose the function you want to invoke from the Function Name list. To locate in the code the function you're invoking, you use the Navigate to Definition feature.

To invoke a function, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Invoke Function** from the Functions section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, from the **Function Name** list box, select the function you want to invoke.
4. To locate the function you've chosen in the code, click **Navigate to Definition**. Kony Visualizer opens the asset that contains the function to the line number where the function begins.
5. Save the action by pressing **Ctrl+S**.

Widgets Actions

From the Widgets section of the list of actions available along the left column of the Action Editor, you can add any of the following four actions. Click an action for instructions on how to add it to an action sequence.

Action	Property
Set Widget Property	Alters a widget's property. The following are the properties that you can define: • **isVisible**:This property defines whether you wanted a widget to appear in an application. • **text**: This property allows you to update the text that is displayed on a widget. • **src**: This property allows you to update an image to the **Image2** widget.
Set Master Data	Some widgets, such as CheckBoxGroup, ComboBox, DataGrid, RadioButtonGroup, PickerView, and Segment2, need to be configured with an initial set of options or data before they can meet the needs for which you're using them. This initial configuration is referred to as the widget's master data.
Set Widget Skin	Modifies which skin is associated with the widget.
Set Map Location	Assigns map coordinates to a Map widget.

Set Widget Property

To configure an action that sets a particular property for a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.

2. From the list of actions available along the left column of the Action Editor, click **Set Widget Property** from the Widget section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, from the Property drop-down list, select the property that you want to set. When you do so, that property's sub-properties display.
4. Make changes to the sub-properties.
5. For each additional widget property that you want to set for this action sequence, repeat steps 2 through 4.
6. Save the action by pressing **Ctrl+S**.

Set Master Data

Some widgets, such as the CheckBoxGroup, ComboBox, DataGrid, RadioButtonGroup, PickerView, and Segment2, need to be configured with an initial set of options or data before they can meet the needs for which you're using them. For instance, a checkbox group needs to be configured with the correct number of options, their labels, and their initial setting (either checked or not checked). This initial configuration is referred to as the widget's master data.

Setting the master data is applicable only for certain widgets.

To configure an action that sets the master data for a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. This widget needs to be capable of having master data set for it, such as a CheckBoxGroup. Once it’s highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Set Master Data** from the Widget section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, click **Set Master Data**.

4. Configure the master data so that the data set of the widget contains the data you want as a result of the action.
5. When you are finished, click **OK**.
6. Save the action by pressing **Ctrl+S**.

Set Widget Skin

To configure an action that sets the skin of a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Set Widget Skin** from the Widget section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, from the **Skin** list, select whether the skin you select applies to the Skin or the focusSkin.
4. From the **Theme** list, select the theme you want to use.
5. Select the skin you want from the list of available skins. The quantity and names of the skins listed depend on whether or not you have already created skins.
6. Save the action by pressing **Ctrl+S**.

Set Map Location

The Set Map Location action applies only to the Map widget, and gives you the ability to set a map location, label, and description in response to an event.

To configure an action that sets the map location for a Map widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Set Map Location** from the Widget section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, in the **Latitude** text box, enter the latitude coordinate for the location.
4. In the **Longitude** text box, enter the longitude coordinate for the location.
5. In the **Name** text box, enter the name of the location that you want the user to see.
6. In the **Desc** text box, enter the description of the location that you want the user to see.
7. Save the action by pressing **Ctrl+S**.

Client Actions

From the Client section of the list of actions available along the left column of the Action Editor, you can add any of the following four actions. Click an action for instructions on how to add it to an action sequence.

Action	Property
Send SMS	Sends a text message.
Send Email	Sends an email message.
Anchor Popup	Anchors a pop-up to the form from where the pop-up is called.

Action	Property
Dismiss Popup	Closes a pop-up.
Display Popup	Opens a pop-up.
Get Local Storage	Returns a structured clone of the current value associated with the given key. If the given key does not exist in the list associated with the object then this method returns null.
Set Local Storage	Creates a structured clone of the given value, assigning it to the given key.
Open URL	Opens a web page.
Phone Call	Make calls to other numbers without leaving the application.
Show Alert	Displays an alert message to the user. Alerts can be one of three types: • **Confirmation**: A confirmation message with Yes and No options appears on the screen. • **Error**: An error message appears on the screen. • **Info**: An informative message appears on the screen. This message can function as either a warning or a success message. All the alerts are modal in nature. That is, the user cannot proceed with other UI operations until the alert is dismissed.

Send SMS

To configure an action that sends an SMS message, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Send SMS** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, in the **Send SMS to number** text field, enter the phone number to where you want the SMS message sent.
4. In the **SMS Content** text field, enter the text of the message you want sent.
5. Save the action by pressing Ctrl+S.

Send Email

To configure an action that sends an email message, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Send Email** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, enter the appropriate values for the following:

Field	Description

To	The email address of the recipient or recipients (separated by semicolons) that the email message is being sent to.
Cc	The email address of the recipient or recipients that are to receive a "carbon copy" of the email message being sent.
Bcc	The email address of the recipient or recipients that are to receive a "blind carbon copy" of the email message being sent (i.e. the other recipients of the email message are not able to see that the Bcc recipients are included.
Subject	The text that you want to appear in the subject line of the email message.
HTML Formatted Body	Check this check box if the body of your email message includes HTML code, otherwise, the email message is sent using simple text formatting. ***Note:*** If you check the **HTML Formatted Body** check box, you must ensure that the HTML code that you use in the body of your email message is correct. Otherwise, your message may not display correctly.
Body	The text that you want to appear in the body of the email message. This text appears in simple text format unless you check the **HTML Formatted Body** check box and format the body with correct HTML code.

4. Save the action by pressing Ctrl+S.

Pop up behavior (Display, Anchor, Dismiss)

To add behaviors to a Pop-up in an action sequence, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.

2. From the list of actions available along the left column of the Action Editor, click one of the three Pop-up actions from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, select the popup that the navigational action applies to.
4. Save the action by pressing **Ctrl+S**.

Get Local Storage

To define an action sequence to get the local storage, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Get Local Storage** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, in the **Key** field, you can either select `set` and enter a value for the key, or select `choose` and select a key from a list of existing keys.
4. In the **Assign To** list, select the variable to assign a value associated with the key. To do so, you can either select `set` and enter a value for the key, or select `choose` and select a key from a list of existing keys.
5. Save the action by pressing **Ctrl+S**.

Set Local Storage

To define an action sequence to set the local storage, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Set Local Storage** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, in the **Key** field, you can either select `set` and enter a name for the key, or select `choose` and select a key from a list of existing keys.
4. In the Value field, you can either select `set` and enter a value for the key, or select `choose` and select a value from a list of existing keys.
   If you select set, select a data type for the value from the list, either String, Number, or Boolean. Then, enter a value for the key in the text box provided below the type field.
5. Save the action by pressing **Ctrl+S**.

Open URL

To configure an action that opens a URL, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Open URL** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, in the URL to Open text box, enter the URL that you want to open.
4. Save the action by pressing Ctrl+S.

Phone Call

To configure an action that places a Phone Call, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Phone Call** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, in the **Phone Number** field, you can either select `set` and enter the phone number to which you want to place a call, or select `choose` and select an existing phone number from the list.
4. Save the action by pressing Ctrl+S.

Show Alert

To configure an action that shows an alert, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, click **Show Alert** from the Client section. The action is added to the action sequence and is the current action of focus.
3. In the properties pane of the Action Editor, enter the appropriate values for the following:

Field	Description
Alert Type	From the Alert Type drop-down list, select either **Confirmation**, **Error**, or **Info**.

Alert Title	The text that the user will see in the title bar of the alert. Alternately, if you are localizing the app and have set up an internationalization (i18n) key for this value, select the key you want from the Alert Title i18n drop-down list.
Yes Label	The button text that the user will see to accept the information of the alert, commonly configured as *Yes* or *OK*. Alternately, if you are localizing the app and have set up an internationalization (i18n) key for this value, select the key you want from the Yes Label i18n drop-down list.
No Label	The button text that the user will see to reject the information of the alert, commonly configured as *No* or *Cancel*. Alternately, if you are localizing the app and have set up an internationalization (i18n) key for this value, select the key you want from the No Label i18n drop-down list.
Alert Icon	If you want an icon to accompany the alert message, click **Browse**, select the .png format image you want to use (or provide a URL to an externally located image), and then click **OK**.
Message Type	From the Message Type drop-down list, select either **Constant** or **Expression**.
Alert Message	The body of the alert, informing users of what you want them to know. Alternately, if you are localizing the app and have set up an internationalization (i18n) key for this value, select the key you want from the Alert Message i18n drop-down list.

4. Save the action by pressing Ctrl+S.

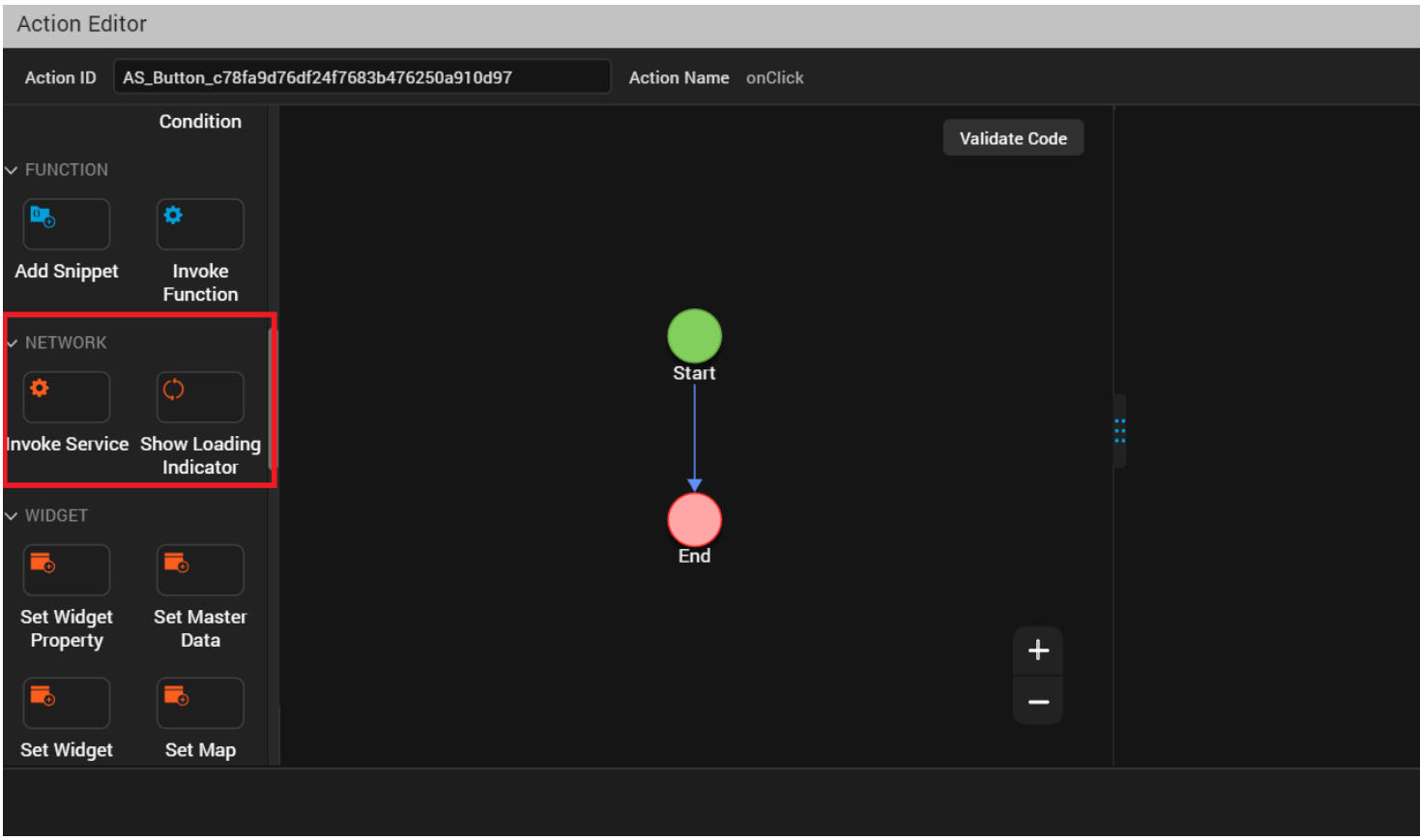
Network Actions

The **Network** category of actions is available in the list of actions on the left pane of Action Editor. You can add the following two actions from the Network section. Click an action for instructions on how to add it to an action sequence.

- Invoke a Service
  - Invoke an Identity Service
  - Invoke an Integration Service

- Invoke an Orchestration Service
- Invoke an Object Service

- Show Loading Indicator



Invoke a Service

The **Invoke Service** action is available under the **Network** category of Action Editor. You can also search for this action by typing *Invoke a Service* in the Search box, on the left pane of Action Editor.

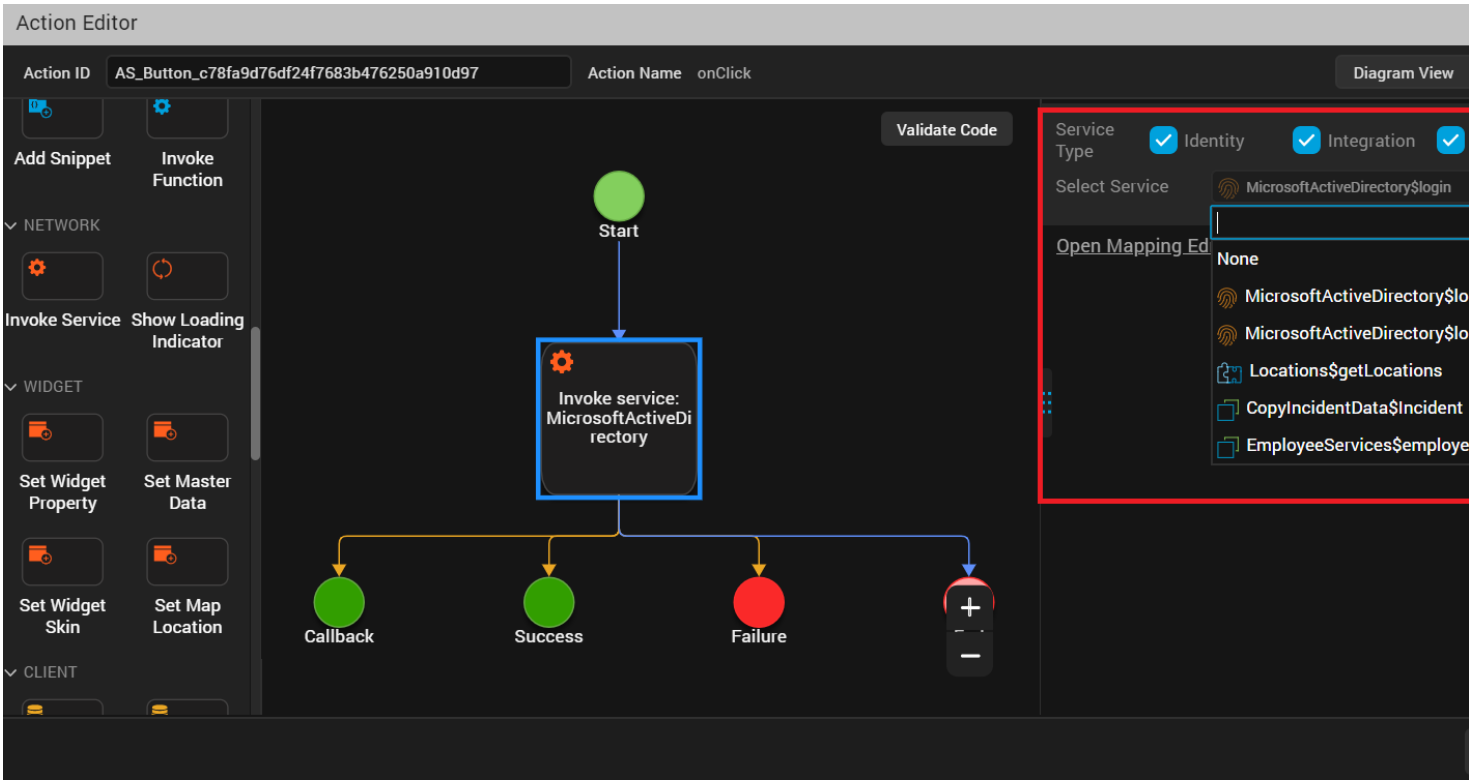
You can use the Invoke Service option to define an action sequence for invoking the following types of services:

- Identity
- Integration

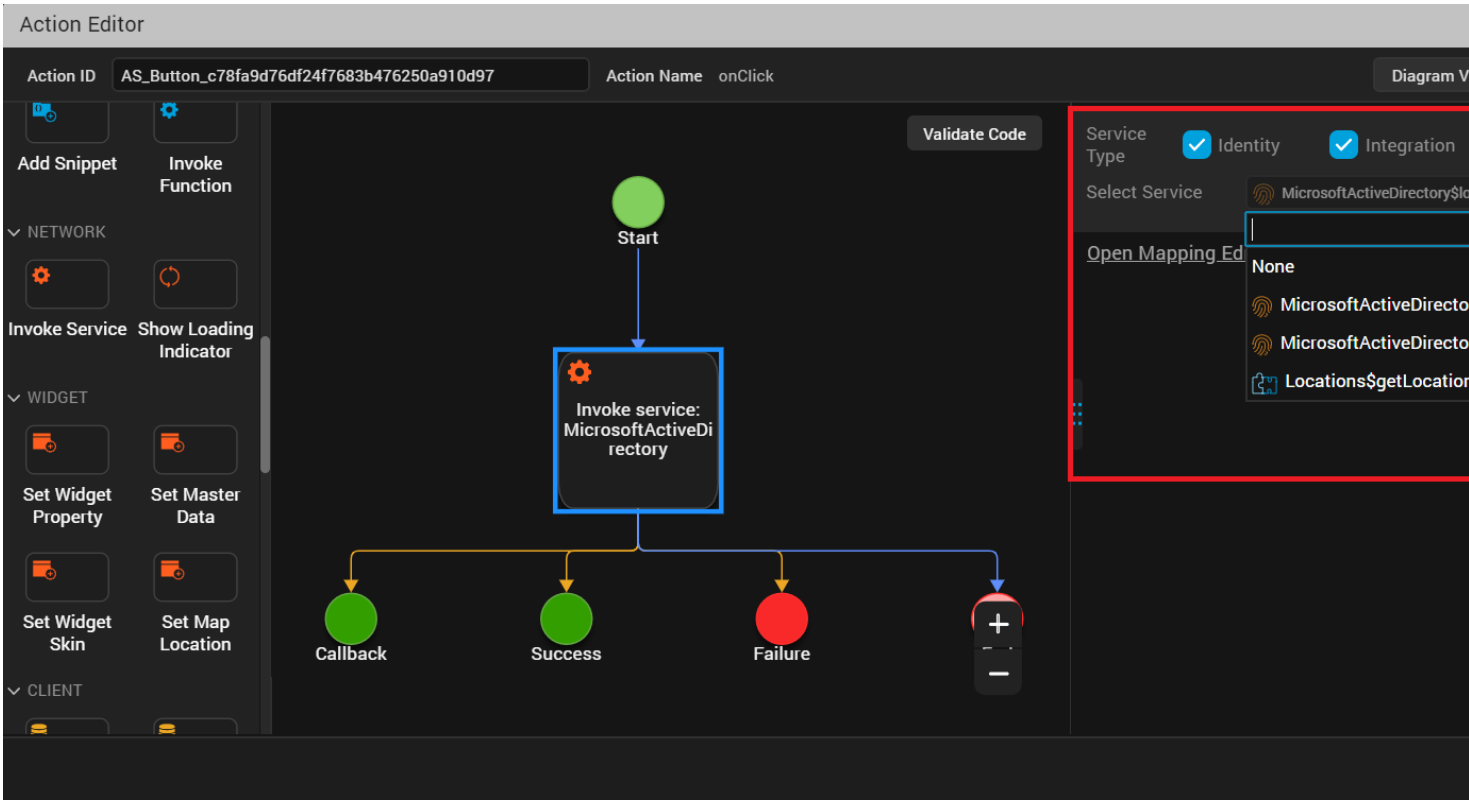
- Orchestration
- Object

When you add Invoke Service to an action sequence, all four types of services are displayed on the right pane of the Action Editor. In addition, the check boxes of all the service types are selected by default.

If you click the Search icon for the **Select Service** box, every service that is available in your Kony Fabric account is displayed in the search result.

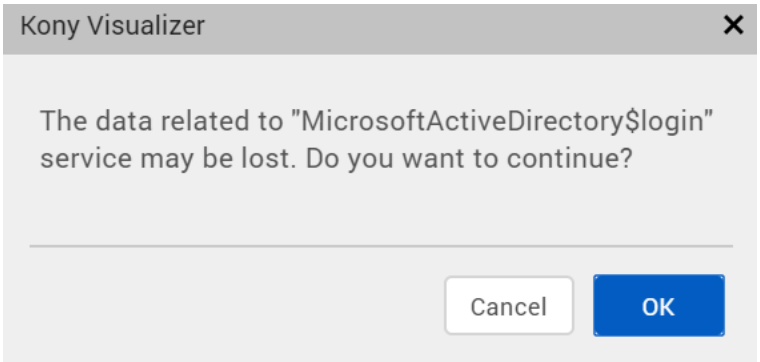


However, if you clear any service type check boxes and keep certain check boxes selected, a filtered list of services appears when you click the Search icon.



Consider a scenario where you have added the *MicrosoftActiveDirectory$login* Identity service to the action sequence of a form or widget. If you then try to change the service of that same action sequence to another service (say, *MicrosoftActiveDirectory$logout*), Kony Visualizer displays a dialog box saying that all the data related to the *MicrosoftActiveDirectory$login* service will be lost.

You can click **OK** to modify the service of the action sequence, or click **Cancel** to continue with the original service.



Invoke an Identity Service

To invoke an Identity service when for or widget on a form, use this action sequence in Action Editor. When you select only the **Identity** check box on the right pane of Action Editor, all the Identity services that are available in your Kony Fabric account appear in the **Select Service** drop-down list box. You can choose a specific Identity service and define the required mappings in Mapping Editor for the action sequence of a form or widget. You can choose either the **login** or **logout** operation for an Identity service.

To define an action sequence for invoking an Identity service, follow these steps:

1. From the **Project** tab of the Project Explorer, select the form and widget for which you want to invoke the Identity service action.
2. Go to the **Properties** panel > **Action**. For the required action (such as, **onClick**), click **Edit**. The Action Editor window opens and an action sequence is created in Diagram View, by default.
3. In either Diagram View or Design View, on the left pane of the Action Editor, under the **Network** category, locate the **Invoke Service** action.
4. You can either drag and drop the **Invoke Service** action onto the action sequence, or simply click the **Invoke Service** action. The Invoke Service action is added to the action sequence, and all the types of services are displayed on the right pane of the Action Editor.
5. The check boxes of all the service types are selected by default. Keep only the **Identity** check box selected, and clear the check boxes of all the other service types.
6. For the **Select Service** box, click the Search icon. The list of all available Identity services is displayed.

> ***Note:*** Identity services are displayed with this icon:  . This icon is displayed beside all Identity services in the search results, and in the Select Service box after you select an Identity service. The presence of this icon helps you to easily identify the type of service that you want to select. So if different types of services have duplicate names, the accompanying icons help you in not selecting the wrong service by mistake.

7. Select the required Identity service; for example, *MicrosoftActiveDirectory$login*. The **Open Mapping Editor** button is displayed.
8. The Mapping Editor opens. Map the service parameters to the required form, variable, or widget on the form from which the service was invoked.
9. Click **Save**. An action sequence for invoking an Identity service is defined in the selected widget.

Invoke an Integration Service

To invoke an Integration service when an event is triggered for a form or widget, use this action sequence in Action Editor. When you select only the **Integration**check box on the right pane of Action Editor, all the Integration services that are available in your Kony Fabric account appear in the **Select Service** drop-down list. You can choose a specific Integration service for the action sequence of a form or widget. You can either define the required mappings of the selected Integration service in Mapping Editor or define custom HTTP headers and HTTP config parameters (timeout).

To define an action sequence for invoking an Integration service, follow these steps:

1. From the **Project** tab of the Project Explorer, select the form or widget for which you want to apply the action.
2. Go to the **Properties** panel > **Action**. For the required action (such as, **onTouchStart**), click **Edit**. The Action Editor window opens and an action sequence is created in Diagram View, by default.

3. In either Diagram View or Design View, on the left pane of the Action Editor, under the **Network** category, locate the **Invoke Service** action.
4. You can either drag and drop the **Invoke Service** action onto the action sequence, or simply click the **Invoke Service** action. The Invoke Service action is added to the action sequence, and all the types of services are displayed on the right pane of the Action Editor.
5. The check boxes of all the service types are selected by default. Keep only the **Integration** check box selected, and clear the check boxes of all the other service types.
6. For the **Select Service** box, click the Search icon. The list of all available Integration services is displayed.

> ***Note:*** Integration services are displayed with this icon:  . This icon is displayed beside all Integration services in the search results, and in the Select Service box after you select an Integration service. The presence of this icon helps you to easily identify the type of service that you want to select. So if different types of services have duplicate names, the accompanying icons help you in not selecting the wrong service by mistake.

7. Select the required Integration service. The **Open Mapping Editor** button and the **Custom HTTP Headers** and **HTTP Config Parameters** sections are displayed.
8. You can perform any of the following actions:
    - Click **Open Mapping Editor**. The Mapping Editor opens. Map the service parameters to the required form, variable, or widget on the form from which the service was invoked.
    - Add your own **Custom HTTP Headers**. Define any information that you want to pass in the header of the service call, apart from the default header parameters. Each row allows you to add a custom header parameter along with its value. Click the **+** icon to add more rows, and click the **X** icon to delete existing rows.
    - In the **timeout** box, enter a value for the time in milliseconds (ms) after which the timeout of the action sequence must occur.

9. Click **Save**. An action sequence for invoking an Integration service is defined in the form or widget.

Invoke an Orchestration Service

To invoke an Orchestration service when an event is triggered for a form or widget, use this action sequence in Action Editor. When you select only the **Orchestration** check box on the right pane of Action Editor, all the Orchestration services that are available in your Kony Fabric account appear in the **Select Service** drop-down list. You can choose a specific Orchestration service and method to add to the action sequence of a form or widget.

To define an action sequence for invoking an Orchestration service, follow these steps:

1. From the **Project** tab of the Project Explorer, select the form or widget for which you want to apply the action.
2. Go to the **Properties** panel > **Action**. For the required action (such as, **onTouchStart**), click **Edit**. The Action Editor window opens and an action sequence is created in Diagram View, by default.
3. In either Diagram View or Design View, on the left pane of the Action Editor, under the **Network** category, locate the **Invoke Service** action.
4. You can either drag and drop the **Invoke Service** action onto the action sequence, or simply click the **Invoke Service** action. The Invoke Service action is added to the action sequence, and all the types of services are displayed on the right pane of the Action Editor.
5. The check boxes of all the service types are selected by default. Keep only the **Orchestration** check box selected, and clear the check boxes of all the other service types.
6. For the **Select Service** box, click the Search icon. The list of all available Orchestration services is displayed.

> ***Note:*** Orchestration services are displayed with this icon:  . This icon is displayed beside all Orchestration services in the search results, and in the Select Service box after you select an Orchestration service. The presence of this icon helps you to easily identify the type of service that you want to select. So if different types of services have duplicate names, the accompanying icons help you in not selecting the wrong service by mistake.

7. Select the required Orchestration service. The **Open Mapping Editor** button and the **Custom HTTP Headers** and **HTTP Config Parameters** sections are displayed.
8. You can perform any of the following actions:
   - Click **Open Mapping Editor**. The Mapping Editor opens. Map the service parameters to the required form, variable, or widget on the form from which the service was invoked.
   - Add your own **Custom HTTP Headers**. Define any information that you want to pass in the header of the service call, apart from the default header parameters. Each row allows you to add a custom header parameter along with its value. Click the **+** icon to add more rows, and click the **X** icon to delete existing rows.
   - In the **timeout** box, enter a value for the time in milliseconds (ms) after which the timeout of the action sequence must occur.
9. Click **Save**. An action sequence for invoking an Orchestration service is defined in the form or widget.

Invoke an Object Service

To invoke an Object service when an event is triggered for a form or widget, use this action sequence in Action Editor. When you select only the **Object** check box on the right pane of Action Editor, all the Object services that are available in your Kony Fabric account appear in the **Select Service** drop-down list. You can choose a specific Object service and method to add to the action sequence of a form or widget.

To define an action sequence for invoking an Object service, follow these steps:

1. From the **Project** tab of the Project Explorer, select the form or widget for which you want to apply the action.
2. Go to the **Properties** panel > **Action**. For the required action (such as, **onTouchStart**), click **Edit**. The Action Editor window opens and an action sequence is created in Diagram View, by default.
3. In either Diagram View or Design View, on the left pane of the Action Editor, under the **Network** category, locate the **Invoke Service** action.
4. You can either drag and drop the **Invoke Service** action onto the action sequence, or simply click the **Invoke Service** action. The Invoke Service action is added to the action sequence, and all the types of services are displayed on the right pane of the Action Editor.
5. The check boxes of all the service types are selected by default. Keep only the **Object** check box selected, and clear the check boxes of all the other service types.
6. For the **Select Service** box, click the Search icon. The list of all available Object services is displayed.
7. Select the required Object service. The **Select Method** box is displayed.

> ***Note:*** Object services are displayed with this icon:  . This icon is displayed beside all Object services in the search results, and in the Select Service box after you select an Object service. The presence of this icon helps you to easily identify the type of service that you want to select. So if different types of services have duplicate names, the accompanying icons help you in not selecting the wrong service by mistake.

8. For the **Select Method** box, click the Search icon. The list of all available methods for the selected Object service is displayed.
9. Each Object service contains the following four methods: **get**, **create**, **update**, and **delete**.
10. If you select the **get** method, the **Construct Filter Query** and **Advanced** buttons are displayed. Follow any of these steps:

- Click **Construct Filter Query**. The Mapping Editor opens. Map the service parameters to variables, data store keys, or widgets on the form from which the service was invoked.
- Click **Advanced** to manually specify the values of **Request Parameters**, add your own **Custom HTTP Headers**, and define the **HTTP Config Parameters** (timeout):
    - **Request Parameters**: Configure the appropriate values for the available request parameters.
    - **Custom HTTP Headers**: Define any information that you want to pass in the header of the service call, apart from the default header parameters. Each row allows you to add a custom header parameter along with its value. Click the **+** icon to add more rows, and click the **X** icon to delete existing rows.
    - **HTTP Config Parameters**: In the **timeout** box, enter a value for the time in milliseconds (ms) after which the timeout of the action sequence must occur.

11. If you select either the **create**, **update**, or **delete** methods, the **Open Mapping Editor** and **Advanced** buttons are displayed. Follow any of these steps:
    - Click **Open Mapping Editor**. The Mapping Editor opens. Map the service parameters to the required form, variable, or widget on the form from which the service was invoked.
    - Click **Advanced** to manually add your own **Custom HTTP Headers** and define the **HTTP Config Parameters** (timeout). You can define any information that you want to pass in the header of the service call, apart from the default header parameters. Each row allows you to add a custom header parameter along with its value. Click the **+** icon to add more rows and the **X** icon to delete existing rows. In the **timeout** box, enter a value for the time in milliseconds (ms) after which the timeout must occur.
12. Click **Save**. An action sequence for invoking an Object service is defined in the form or widget.

Show Loading Indicator

To show/dismiss the loading indicator of an action sequence, follow these steps:

1. From the **Project** tab of the Project Explorer, select the form or widget for which you want to apply the action.
2. Go to the **Properties** panel > **Action**. For the required action (such as, **onTouchStart**), click **Edit**. The Action Editor window opens and an action sequence is created in Diagram View, by default.
3. In either Diagram View or Design View, on the left pane of the Action Editor, under the **Network** category, locate the **Show Loading Indicator** action.
4. You can either drag and drop the **Show Loading Indicator** action onto the action sequence, or simply click the **Show Loading Indicator** action. The Show Loading Indicator action is added to the action sequence, and various associated fields are displayed on the right pane of the Action Editor.
5. Perform the following actions for the displayed fields:
   - **Loading Indicator**: To show the loading indicator in the action sequence, select the **Show** option. This option is selected by default.
     If you do not want to display the loading indicator in the action sequence, select the **Dismiss** option. All the remaining fields are hidden if you select this option.
   - **Skin**: Select the skin of the loading indicator from the drop-down box.
   - **Text**: Type the text that must appear in the loading indicator.
   - **Position**: If you select the **Full Screen** option, the loading indicator will appear in the entire screen of your app. If you select the **Center Only** option, the loading indicator will appear only in the center position of the screen.
   - **isBlocked**: Select either the **True** or **False** option.
   - **Show Progress Indicator**: Select either the **True** or **False** option to configure whether the progress of the loading indicator will be displayed.
6. Click **Save**. The loading indicator configuration details are successfully saved for the selected form or widget.

Animation Actions

From the Animation section of the list of actions available along the left column of the Action Editor, you can add any of the following six actions. Click an action for instructions on how to add it to an action sequence.

Action	Property
Flex Move	Moves the widget from its original position.
Flex Scale	Resizes a widget.
Flex Layout	Moves, scales, and rotates a widget with a single action along an X and Y axis (two dimensional).
Transform	Moves, scales, and rotates a widget with a single action along an X, Y, and Z axis (three dimensional).
Rotate	Rotates a widget along an X and Y axis (two dimensional).
Rotate 3D	Rotates a widget along an X, Y, and Z axis (three dimensional).
Set Style	Changes the background color of a widget.

Flex Move

To move a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Flex Move** from the Animation section of available actions.
3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	Move_left
Left	Specifies the distance a widget moves to the left from its original position.	-20 px, 100 dp, 15%
Right	Specifies the distance a widget moves to the right from its original position.	10 px, 175 dp, 25%
Top	Specifies the distance a widget moves to the top from its original position.	-15 px, 200 dp, 7%
Bottom	Specifies the distance a widget moves to the bottom from its original position.	23 px, 225dp, 6%
Center X	Specifies the distance a widget's center moves horizontally from its original position.	-21 px, 263 dp, 11%
Center Y	Specifies the distance a widget's center moves vertically from its original position.	-15 px, 200 dp, 7%
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000
Time Function	The following time functions are available: • **Ease.** Specifies a transition effect with a slow start, then fast, then end slowly. • **Linear.** Specifies a transition effect with the same speed from start to end. • **Ease-In.** Specifies a transition effect with a slow start. • **Ease-Out.** Specifies a transition effect with a slow end. • **Ease-In-Out.** Specifies a transition effect with a slow start and end.	
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000

Property	Description	Examples
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate
Start	Specifies how an animation action starts. The following options are available: • **Immediately.** Start the animation immediately, with delay, if selected. • **With Animation.** Action is run along with another action. • **After Animation.** Action is run after completing another action. ***Note:*** **With Animation** and **After Animation** properties are ignored if only single action is available in the **Action Sequence** pane.	Immediately

Property	Description	Examples
Inherit	The following options are available: • **None.** Upon completing this action, widget returns to its original size. • **Forward.** Upon completing this action, widget stays at final position. • **Backward.** Upon completing an action, widget will come return to the original position. • **Both.** Applies Forward and Backward options.	Both

> ***Note:*** For the following properties Left, Right, Top, Bottom, Center X and Center Y, you can specify the values in Device Independent Pixels (dp), pixels (px), and percentage (%). You can also specify negative values for these properties.

Flex Scale

To scale a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Scale** from the Animation section of available actions.
3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	Scale_ widget
Width	Specifies the width of a widget.	100 px
Height	Specifies the height of a widget.	250 dp
Min Width	Specifies the minimum width of a widget.	20%
Min Height	Specifies the minimum height of a widget.	15%
Max Width	Specifies the maximum width of a widget.	540px
Max Height	Specifies the maximum width of a widget.	300px
Anchor X	Specifies the horizontal anchor point from where widget rotation begins.	15%
Anchor Y	Specifies the vertical anchor point from where widget rotation begins.	25%
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000
Time Function	The following time functions are available: • **Ease.** Specifies a transition effect with a slow start, then fast, then end slowly. • **Linear.** Specifies a transition effect with the same speed from start to end. • **Ease-In.** Specifies a transition effect with a slow start. • **Ease-Out.** Specifies a transition effect with a slow end. • **Ease-In-Out.** Specifies a transition effect with a slow start and end.	
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5

Property	Description	Examples
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate
Start	Specifies how an animation action starts. The following options are available: • **Immediately.** Start the animation immediately, with delay, if selected. • **With Animation.** Action is run along with another action. • **After Animation.** Action is run after completing another action. ***Note:*** **With Animation** and **After Animation** properties are ignored if only a single action is available in the **Action Sequence** pane.	Immediately

Property	Description	Examples
Inherit	The following options are available: • **None.** Upon completing this action, widget returns to its original size. • **Forward.** Upon completing this action, widget stays at final position. • **Backward.** Upon completing an action, widget will come return to the original position. • **Both.** Apply Forward and Backward options.	Both

Note: For the following properties Width, Height, Min Width, Min Height, Max Width, Max Height, you can specify the values in Device Independent Pixels (dp), pixels (px), and percentage (%). You can also specify negative values for these properties.

Flex Layout

Using this single action, you can move, scale, and rotate a widget in two dimensions.

To use the Flex Layout animation, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Flex Layout** from the Animation section of available actions.
3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	FlexLayoutAnimation
Left	Specifies by how much a widget moves to the left from its original position.	-20 px, 100 dp, 15%
Right	Specifies by how much a widget moves to the right from its original position.	10 px, 175 dp, 25%
Top	Specifies by how much a widget moves to the top from its original position.	-15 px, 200 dp, 7%
Bottom	Specifies by how much a widget moves to the bottom from its original position.	23 px, 225dp, 6%
Center X	Specifies by how much a widget's center moves horizontally from its original position.	-21 px, 263 dp, 11%
Center Y	Specifies by how much a widget's center moves vertically from its original position.	-21 px, 263 dp, 11%
Width	Specifies the width of a widget.	100 px
Height	Specifies the height of a widget.	250 dp
Min Width	Specifies the minimum width of a widget.	20%
Min Height	Specifies the minimum height of a widget.	15%
Max Width	Specifies the maximum width of a widget.	540px
Max Height	Specifies the maximum width of a widget.	300px
Rotate	Specifies the angle at which a widget is rotated.	45
Anchor X	Specifies the horizontal anchor point from where widget rotation begins.	15%
Anchor Y	Specifies the vertical anchor point from where widget rotation begins.	15%
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000

Property	Description	Examples
Time Function	The following time functions are available: • **Ease.** Specifies a transition effect with a slow start, then fast, then end slowly. • **Linear.** Specifies a transition effect with the same speed from start to end. • **Ease-In.** Specifies a transition effect with a slow start. • **Ease-Out.** Specifies a transition effect with a slow end. • **Ease-In-Out.** Specifies a transition effect with a slow start and end.	
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate

Property	Description	Examples
Start	Specifies how an animation action starts. The following options are available: • **Immediately.** Start the animation immediately, with delay, if selected. • **With Animation.** Action is run along with another action. • **After Animation.** Action is run after completing another action. ***Note:*** **With Animation** and **After Animation** properties are ignored if only single action is available in the **Action Sequence** pane.	Immediately
Inherit	The following options are available: • **None.** Upon completing this action, widget returns to its original size. • **Forward.** Upon completing this action, widget stays at final position. • **Backward.** Upon completing an action, widget will come return to the original position. • **Both.** Apply Forward and Backward options.	Both

Note: For the following properties Left, Right, Top, Bottom, Center X and Center Y, Width, Height, Min Width, Min Height, Max Width, and Max Height, you can specify the values in Device Independent Pixels (dp), pixels (px), and percentage (%). You can also specify negative values for these properties.

Transform

Using this single action, you can move, scale, and rotate a widget in three dimensions.

To use the Flex Layout animation, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Flex Layout** from the Animation section of available actions.
3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	TransformAnimation
Left	Specifies by how much a widget moves to the left from its original position.	-20 px, 100 dp, 15%
Top	Specifies by how much a widget moves to the top from its original position.	-15 px, 200 dp, 7%
Width	Specifies the width of a widget.	100 px
Height	Specifies the height of a widget.	250 dp
Rotate	Specifies the angle at which a widget is rotated. This can range between 180 and -180.	45
Anchor X	Specifies the horizontal anchor point from where widget rotation begins.	15%
Anchor Y	Specifies the vertical anchor point from where widget rotation begins.	15%

Property	Description	Examples
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate

Property	Description	Examples
Fill Mode	Defines the appearance of the animated widget after the animation has executed, outside of the time the animation takes place. The following options are available: • **Forward.** Causes the animated widget to carry forward, after the animation, the properties it had when the animation ended. • **Backward.** Causes the animated widget to go back, after the animation, to the properties it had when the animation ended. • **Both.** Causes the animated widget to carry forward, after the animation, the properties it had when the animation ended, except for whatever property or properties are key-framed to revert back after the animation, as defined in additional code.	-21 px, 263 dp, 11%

Note: For the following properties Left, Top, Width, and Height, you can specify the values in Device Independent Pixels (dp), pixels (px), and percentage (%). You can also specify negative values for these properties.

Rotate

To rotate a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Rotate** from the Animation section of available actions.

3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	Rotate_ angle
Rotate	Specifies the angle at which a widget is to be rotated.	45
Anchor X	Specifies the horizontal anchor point from where widget rotation begins.	15%
Anchor Y	Specifies the vertical anchor point from where widget rotation begins.	15%
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000
Time Function	The following time functions are available: • **Ease.** Specifies a transition effect with a slow start, then fast, then end slowly. • **Linear.** Specifies a transition effect with the same speed from start to end. • **Ease-In.** Specifies a transition effect with a slow start. • **Ease-Out.** Specifies a transition effect with a slow end. • **Ease-In-Out.** Specifies a transition effect with a slow start and end.	
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5

Property	Description	Examples
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate
Start	Specifies how an animation action starts. The following options are available: • **Immediately.** Start the animation immediately, with delay, if selected. • **With Animation.** Action is run along with another action. • **After Animation.** Action is run after completing another action. ***Note:*** **With Animation** and **After Animation** properties are ignored if only single action is available in the **Action Sequence** pane.	Immediately

Property	Description	Examples
Inherit	The following options are available: • **None.** Upon completing this action, widget returns to its original size. • **Forward.** Upon completing this action, widget stays at final position. • **Backward.** Upon completing an action, widget will come return to the original position. • **Both.** Apply Forward and Backward options.	Both

Rotate 3D

To rotate a widget along the X, Y, and Z axes, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Rotate 3D** from the Animation section of available actions.
3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	Rotate3D_angle
Rotate 3D	Specifies which axes rotate, and the angle of their rotation. For each axis—X, Y, and Z—you specify either 0, which means you don't want that axis to rotate, or 1, which means you do want it to rotate. You also specify the number of degrees by which the widget should rotate upon the chosen axes.	
Perspective	Sets the perspective transform and the vanishing point to center of the widget; this can be platform-specific. Perspective creates the illusion of distance between the user and the object.	1000
Anchor X	Specifies the horizontal anchor point from where widget rotation begins.	15%
Anchor Y	Specifies the vertical anchor point from where widget rotation begins.	15%

Property	Description	Examples
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000
Time Function	The following time functions are available: • **Ease.** Specifies a transition effect with a slow start, then fast, then end slowly. • **Linear.** Specifies a transition effect with the same speed from start to end. • **Ease-In.** Specifies a transition effect with a slow start. • **Ease-Out.** Specifies a transition effect with a slow end. • **Ease-In-Out.** Specifies a transition effect with a slow start and end.	
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000

Property	Description	Examples
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5

Property	Description	Examples
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate

Property	Description	Examples
Inherit	The following options are available: • **None.** Upon completing the action, the widget returns to its original position. • **Forward.** Upon completing the action, the widget stays at its final position. • **Backward.** Upon completing the action, the widget animates backwards back to its original position. • **Both.** The Forward and Backward options are both implemented.	Both

Set Style

The Set Style animation changes the background color of a widget.

To change the background color of a widget, do the following:

1. From the Project tab of the Project Explorer, select the widget you want to apply the action to. Once it's highlighted on the Visualizer Canvas, right-click it, and then select one of the action sequences, such as onTouchStart. Doing so opens the Action Editor and creates an action sequence for you to configure.
2. From the list of actions available along the left column of the Action Editor, scroll down, and then click **Set Style** from the Animation section of available actions.
3. Update the properties with the following details, and then save your changes to the action sequence by pressing **Ctrl+S**:

Property	Description	Examples
Animation ID	The name or ID of the animation. You can use the default ID, or you can enter your own ID, ensuring that it has no spaces, and does not duplicate the name of an existing animation.	SetStyle_ Color
Background Color	Select a color from the color picker. Only Single color .	
Opacity	Specifies the opacity of the background color.	15%
Time	Specifies the number of milliseconds (ms) in which an animation is completed.	1000
Time Function	The following time functions are available: • **Ease.** Specifies a transition effect with a slow start, then fast, then end slowly. • **Linear.** Specifies a transition effect with the same speed from start to end. • **Ease-In.** Specifies a transition effect with a slow start. • **Ease-Out.** Specifies a transition effect with a slow end. • **Ease-In-Out.** Specifies a transition effect with a slow start and end.	
Delay	Specifies the number of milliseconds (ms) to wait before starting an animation.	2000

Property	Description	Examples
Repeat	Specifies the number of times an action is repeated. To run the action indefinitely, select the **Infinite** check box.	5
Direction	Specifies whether the direction of the animation goes in one direction or alternates. When the animation alternates, the animated widget moves toward the end position specified in the positional properties, and then returns to its starting point at the same rate. The following options are available: • **None.** The animation moves from its starting point to the end position specified in the positional properties. • **Alternate.** When the **Repeat** property is set to greater than 1, the animation action moves toward the end position specified in the positional properties during odd occurrences, and then returns back to the starting point during even occurrences.	None, Alternate
Start	Specifies how an animation action starts. The following options are available: • **Immediately.** Start the animation immediately, with delay, if selected. • **With Animation.** Action is run along with another action. • **After Animation.** Action is run after completing another action. ***Note:*** **With Animation** and **After Animation** properties are ignored if only single action is available in the **Action Sequence** pane.	Immediately

Property	Description	Examples
Inherit	The following options are available: • **None.** Upon completing this action, widget returns to its original size. • **Forward.** Upon completing this action, widget stays at final position. • **Backward.** Upon completing an action, widget will come return to the original position. • **Both.** Apply Forward and Backward options.	Both

Map Widget Properties to One Another

Mapping provides an easy way of linking a widget property (or skin, variable, or an i18n key) to another widget. Mapping also allows you to achieve a uniform look and feel across the design, and reduces the effort required to design an app since any modification made to a widget property results in modifying all its linked widget properties. Click any of the following topics for more information.

- Mapping Overview
- Important Considerations
- Map Properties to One Another
- Add an Expression
- Unlink Mapped Properties
- Locate Properties by Filtering and Searching
- Map Data with the Help of Expressions
- Map a Collection to a Widget or a Service Parameter
- Set Data for a Component with Contract

- Map a Segment's Section Header Template Widgets
- Map Data to the Segment in a Component

Mapping Overview

The Mapping Editor consists of two panes: the source and the target.

Source

The left pane of the Mapping Editor is called the Source, and contains the following folders:

- Forms: Lists all the forms and their child widgets. All the properties of a form and its child widgets that can be mapped are displayed.
- Popups: Lists all the pop-ups and their child widgets. All the properties of a pop-up and its child that can be mapped are displayed.
- Headers: List all the header templates and their child widgets. All the properties of a header and its child that can be mapped are displayed.
- Footer: List all the footer templates and their child widgets. All the properties of a header and its child that can be mapped are displayed.
- Variables: List all the variables created for a project.
- Skins: List all the skins available for a form (or pop-up) and its child widgets.
- i18n Keys: List all the i18n keys available for the project.

Target

The right pane of the Mapping Editor is called the Target, and contains the following folders:

- Forms: Lists all the forms and their child widgets. All the properties of a form and its child widgets that can be mapped are also displayed.

- Popups: Lists all the pop-ups and their child widgets. All the properties of a pop-up and its child that can be mapped are also displayed.
- Headers: List all the header templates and their child widgets. All the properties of a header and its child that can be mapped are also displayed.
- Footer: List all the footer templates and their child widgets. All the properties of a header and its child that can be mapped are also displayed.
- Variables: List all the variables created for a project.

> ***Note:*** Skins and i18n keys folders are unavailable in the Target pane.

Important Considerations

- The datatype for both the source and the target has to be the same. If they're not, Kony Visualizer notifies you.
- You cannot map across channels. That is, you cannot map the widget text of a Mobile channel to the widget text of a Tablet (or Desktop) channel.
- If a form is forked (e.g. iOS:Native), only when the source and target are of same platform (iOS:Native) are the mapped values are applied during functional preview.
- A source element can have one-to-many mappings, but a target element can have only one-to-one mapping. That is, a source element can be linked to multiple target elements. But a target element can only be linked to one source element.
- While mapping skins, source and target widgets should be of the same type. That is, you can map a skin of a Button to another Button only.
- The data type of the source and target should be same. That is, if the source data type is String, you can map it only to a target element of data type String.
- When you map a collection to another collection, you must first map at the parent-level (map one collection to the other collection) and then map the elements.

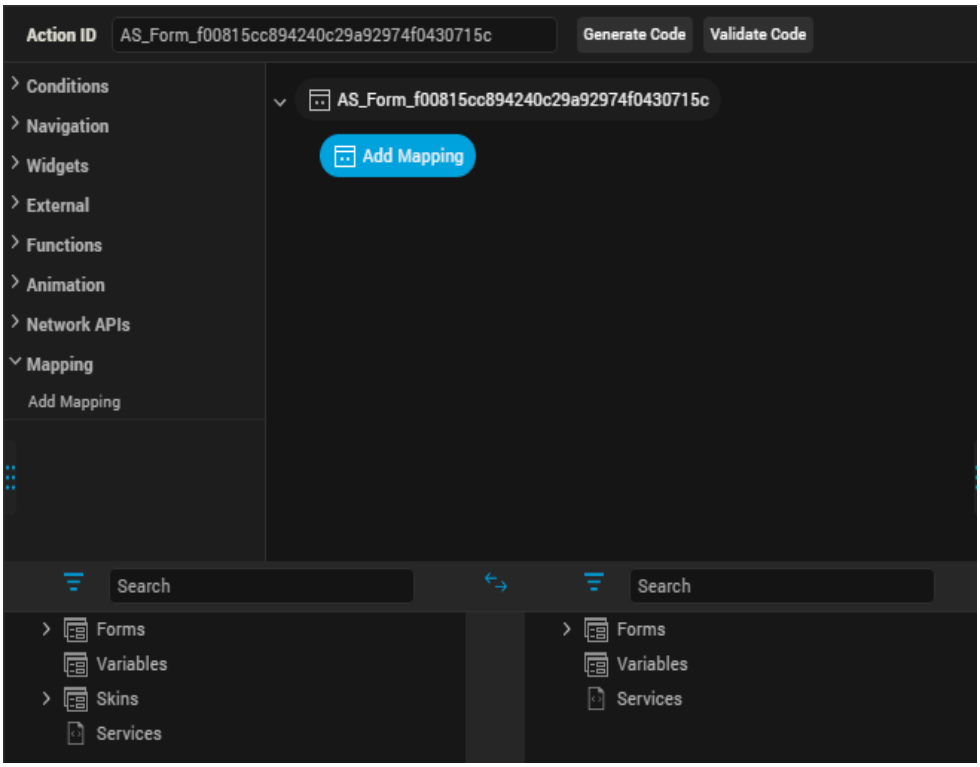
- If the source and target widgets are of the same type, and the values of the two properties being mapped are identical, you cannot map them.
- Since tabs are listed under form elements, you cannot map any data to a tab widget using the Mapping Editor.
- For service mappings, do not use structures that have more than simple key value pairs and attributes. Break out complex structures and use code to get the lost values.
- Kony service mappings cannot support dynamic associations or hierarchical structures. Use code to work around this limitation

Map Properties to One Another

It's important to note that, after mapping, any changes you make to the target element effects the source element as well.

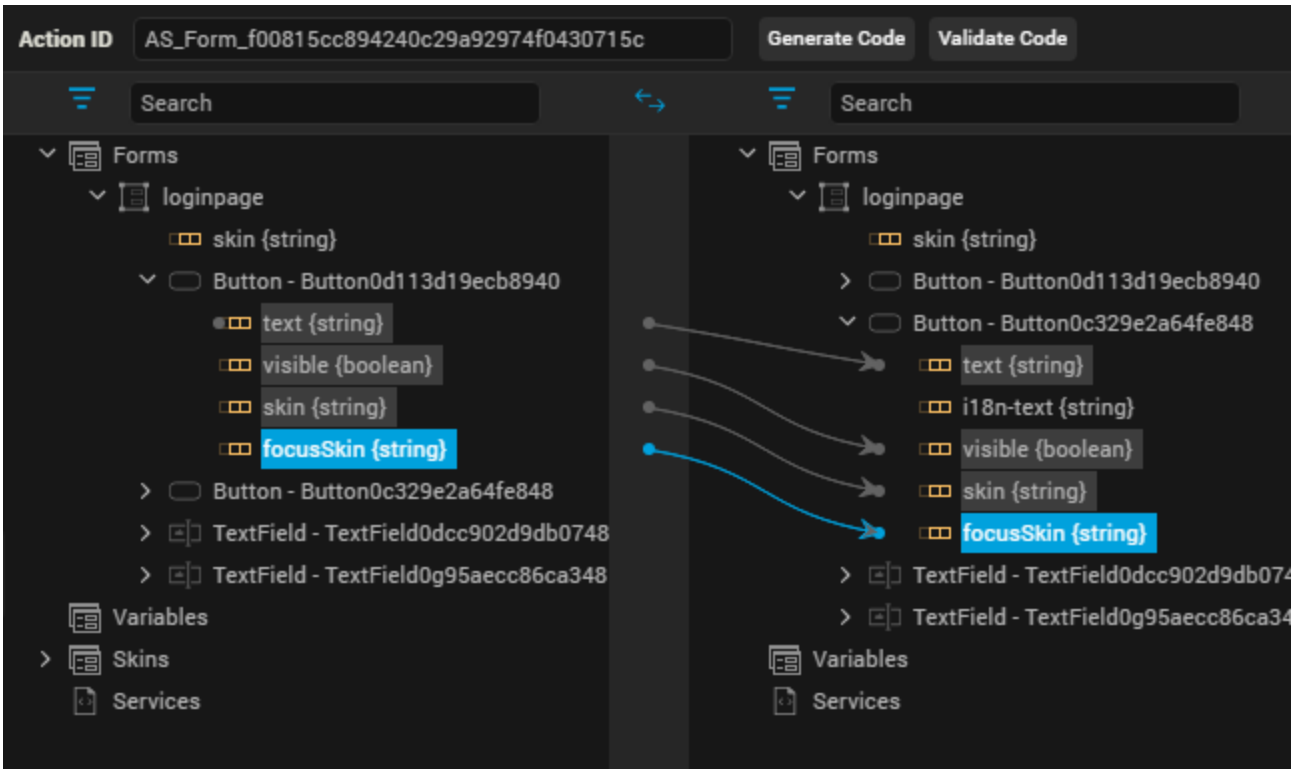
To map properties, do the following:

1. From the left pane of the Action Editor, under the Mapping section, click **Add Mapping**. Doing so opens the Mapping Editor.



2. From the Source pane, navigate to a widget's property, and then click it. You can make it easier to locate the property you want by using the Filter drop-down list to filter the app's forms and widgets according to any of the following criteria: Widget ID, Widget Category, Variables, Skins, I18N Keys, Mapped Elements, and Services.

3. From the Target pane, navigate to a widget's property, and then click it. Doing so establishes a mapping relationship between the source element and the target element. You can make it easier to locate the property you want by using the Filter drop-down list to filter the app's forms and widgets according to any of the following criteria: Widget ID, Widget Category, Variables, Skins, I18N Keys, Mapped Elements, and Services.



4. To save the changes you have made to the action sequence, press Ctrl+S.

Add an Expression

You can write a code snippet on a target element for manipulating a widget property.

To add an expression, do the following:

1. Right-click the target element.
2. Click **Add Expression**.
3. Type the code and save the project. Example of an expression:

```
"Welcome " + loginPage.FirstName.text + loginPage.LastName.text;;
```

Unlink Mapped Properties

To unlink a mapping or to remove any expression, do the following:

1. In the Target pane, right-click a property that is mapped to a source property.
2. Click **Unbind**.
3. To save the changes you have made to the action sequence, press Ctrl+S.

Locate Properties by Filtering and Searching

The elements under Source and Target in the Mapping Editor can be sorted by either filtering or searching. When you search, the pane lists only those elements that match what you type in the Search box. The Filter feature includes the following filtering parameters:

Filter Category	Description
All	Displays all the elements
By Widget Id	Displays widgets sorted by their ID
By Widget Category	Displays widgets sorted by their categories
Variables	Displays only variables, both local and global
Skins	Displays only skins
I18N	Displays only i18n keys
Services	Displays only services

1. To save the changes you have made to the action sequence, press Ctrl+S.

Map Data with the Help of Expressions

You can also map data to widgets or service parameters by creating an expression. When you write an expression to map the data, you can perform the required operations on the data before mapping it to the selected element.

> ***Note:*** You can add an expression only to the elements that have a data type associated with it. These elements can be service parameters or attributes of a widget. You cannot add an expression at the widget name or service name level.

To map data with the help of an expression, do the following:

1. In the Mapping Editor, select the element in the Target to which you want to map data.
2. Right-click this element, and then click **Add Expression**. Doing so displays the Expression Editor.
3. Define the required expression, and then click outside of the Expression Editor. The data derived out of the expression is mapped to the selected element, and a red circle appears next to the element indicating that it has an expression.
4. To save the changes you have made to the action sequence, press Ctrl+S.

Map a Collection to a Widget or a Service Parameter

When you want to map elements within a collection to a widget or a service parameter, the first element in the collection gets mapped to the widget by default.

> ***Note:*** You have to explicitly edit the expression to modify the index of the collection that needs to be mapped.

To edit the collection index value, follow these steps:

1. On the Source pane of the Mapping Editor, select an element within the collection as the source by clicking it.
2. On the Target pane, click a widget (or service parameter) that has the same data type as the collection element. The first element within the collection gets mapped to the widget you selected.
3. Right-click the mapped widget (or service parameter), and then click **Edit Expression**. Doing so opens the Expression Editor.
4. Delete the default index value in the expression and provide the required index value. For example, students[3]["name"] indicates that you have mapped the value at the third index of name within the students collection.

5. Click **OK**.
6. To save the changes you have made to the action sequence, press Ctrl+S.

iOS Navigation Bar

Applies to *Kony Visualizer Classic*.

With Kony VisualizerV8 SP1 release, you can customize the navigation bar in iOS. Before Kony Visualizer V8 SP1 release, a user could only modify the Left button, Right button, and the title. However, iOS natively provides an option to add other UI controls to the navigation bar. To support the UI controls in the navigation bar, Kony has modified the navigation bar to provide the same features in Kony Visualizer.

TitleBar properties for forms created on older version of Visualizer (< V8 SP1) will be carried over to the new NavigationBar when the projects is migrated to Visualizer V8 SP1.

The following are the various new keys that are supported in the NavigationBar tab of Properties pane in Visualizer for iOS.

Key	Description
Render Title Text	Switches the form title on or off.
Bar Style	Specifies the UI bar style to apply to the navigation bar.
Master Data	Collection of bar button items that can be set to the properties (Left Bar Button Items, Right Bar Button Items, and Back Bar Button Item).
Tint Color	Controls the tint color of the navigation bar.
Translucent	Specifies whether the navigation bar is translucent
Prompt	A single line of text displayed at the top of the navigation bar

Key	Description
Shadow Image	Represents the image used as a shadow beneath the navigation bar. This image is stretched horizontally to match the width of the bar.
Extended View	This is a reference to a component without a contract which is displayed at the bottom of the navigation bar.
Back Indicator Image	Specifies the image that appears at the leading edge of the back button. This attribute must be used in combination with the **Back Indicator Transition Mask Image** attribute.
Back Indicator Transition Mask Image	Specifies the mask associated with the **Back Indicator Image** attribute. This is used to control the appearance of the Back button during animated transitions, and therefore must be used in conjunction with the **Back Indicator Image** attribute.
Hides Back Button	Indicates whether the left items are displayed in addition to the back button.
Left Items Supplement Back Button	Sets whether the back button is hidden, optionally animating the transition.

To configure the items on the navigation bar, you must use the **Edit** button in from **Master Data**.

Using the following steps, you can configure any of the Left Bar Button Items, Right Bar Button Items, and the Back Bar Button Item.

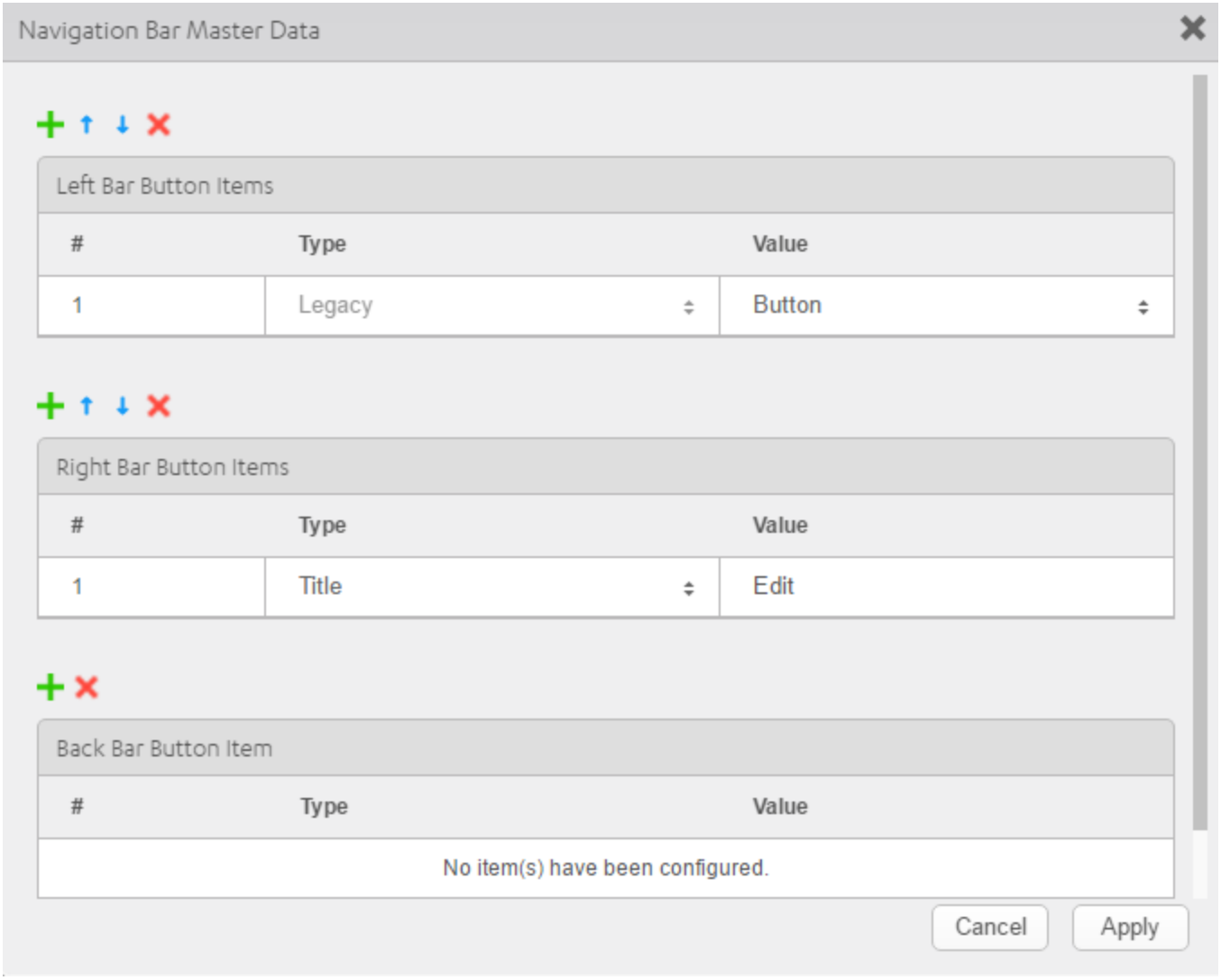
> ***Note:*** You can add multiple items for the Left Bar and the Right Bar, however, you can only add one item to the Back Button Item.

To add an item for the navigation bar, do the following:

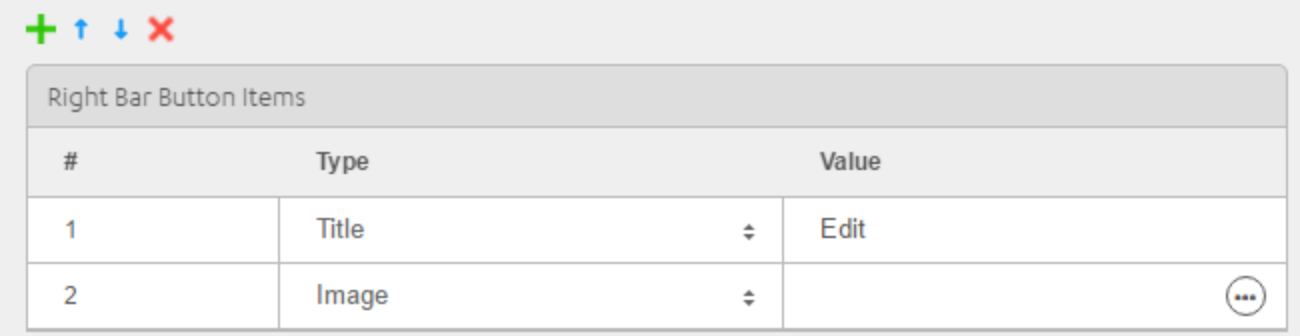
1. From the Project tab of the Project Explorer, select the form you want to modify the navigation bar.

Ensure that your current form is for iOS.

2. Select the Navigation bar on the Visualizer canvas.
3. In the **Properties** pane, click the **NavigationBar** tab.
4. Click the **Edit** button next to **Master Data**. The Navigation Bar Master Data window displays. This will display the existing bar button items. Migrated title bar data appears as type **Legacy**. You can not modify the legacy items type and index.

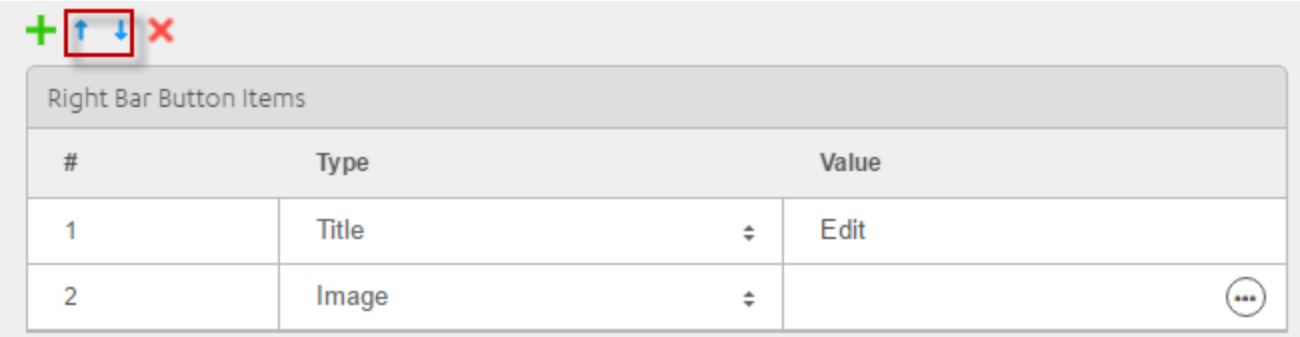


5. For the item you want to add, for example, Right Bar Button Items, click the + symbol. A new row is added.



6. Select the type from the **Type** list. Options are Image, Title. System Item, and Custom. Based on your selection, how you enter the value pair changes.
    i. **Image**: Select an image from your project using the image browser.
    ii. **Title**: Enter the tile in the text box.
    iii. **System Item**: Select an option from the predefined list.
    iv. **Custom**: Select the widget type, Button or Label.

7. If you want to change the order of the Bar Button Items, you can move them up and down using the up arrow and down arrow icons. You must select the Bar Item to move it up or down.



8. Once you are done with your changes, click **Apply**. Changes made will reflect in the Visualizer Canvas.

After you have configured the navigation bar button items, you can select then individually and configure their settings independent of the navigation bar.

Understanding Skins and Themes

A widget's appearance is defined by the skin that is applied to it. Every widget has a skin, even if it's just the Kony Visualizer default skin. Skins give you the ability to establish visual continuity in your app. For instance, you may want all buttons in your app to use the same skin. Skins can also be applied depending on the state of a widget–for instance, when it's at rest, receives focus, or is pressed. Several visual settings comprise a skin, such as the widget background color and opacity; the widget border color, opacity, size, and shape; the widget shadow settings; and in the case of certain widgets, font type, size, color, opacity, and shadow.

You can apply and manipulate skins in many different ways to meet your design needs, including editing, duplicating, copying and pasting, and forking. And by grouping skins together, you can create a theme that you can apply to widgets across the entire application.

You interact with widgets in two primary ways: on the widget level, using the Skins tab of an individual widget, and on a global level, using the Skins tab on the Project Explorer, where the skins of all widgets are listed.

Kony Visualizer also provides a default theme that has a set of skins defined for all the widgets within an application. The skins in this theme can be applied to widgets to get a different look and feel for the widgets. Themes are accessed from the Skins tab of the Project Explorer.

For information on impact of upgrade from any previous Kony Visualizer to Kony Visualizer V8, click here.

The following topics describe how to work with skins and themes:

Use Default Skins

Create a New Skin

Configure a New Skin

Edit a Skin

Apply a Skin

Reuse Skins

Fork a Skin

Delete a Skin

Copy and Paste a Color or Gradient

Using Themes - Applying a Collection of Skins as a Group

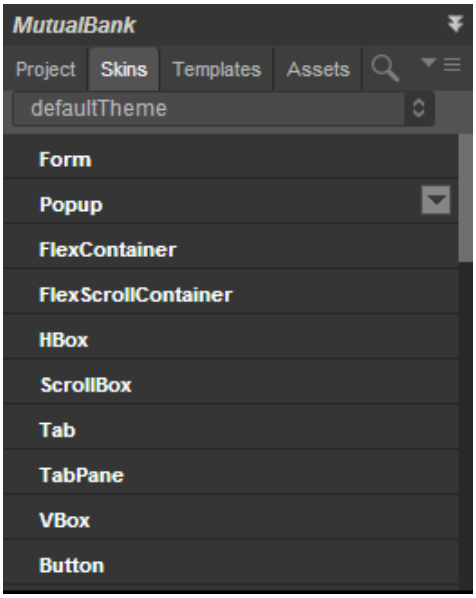
Use Default Skins

The default skins allow you to specify common skins for widgets in states like Normal and Focus. Normal state is the default look of the widget, and Focus state occurs when the user touches or clicks the widget in the application. With Kony default skins, you can provide a uniform look and feel for the widgets across various platforms.

For example, if you specify btnSkinNormal as the Normal skin for Button in Themes, all the buttons in the application will use btnSkinNormal as the Normal Skin for buttons.

Though the Application themes define a common skin for a particular widget across the application, you can still specify a different skin for a widget by selecting a different skin under the widget properties.

To view the default skins navigate to Project Explorer and click the Skins tab.



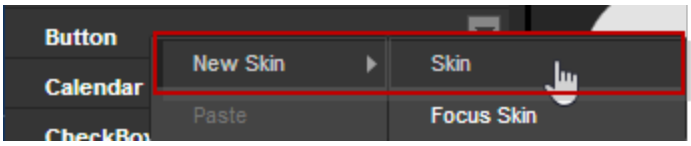
Click on a widget to view the default forms.

Create a New Skin

You can create as many skins in Kony Visualizer as you want so that you can achieve the balance between continuity and flexibility that your app design needs.

To create a new skin, do the following:

1. On the Project Explorer, click the **Skins** tab.
2. On the **Skins** tab, locate the type of widget that you want to create the new skin for, such as a Button, and then click that widget type's context menu arrow .
3. From the menu, click **New Skin**, and then click the type of new skin you want to create. Many widget types offer just the standard Skin, while others also offer a Focus Skin, or some variation.



   Your new skin is created and is ready for you to configure.

Configure a New Skin

Now that you have created a new skin, you need to configure its various properties with the look that you want. Here's what you can do:

Rename the Skin

Set the Skin Background

Set the Skin Border

Set Individual Corners in Borders

Set the Skin Shadow

Set the Skin Font

Set the Skin Font Shadow

Rename the Skin

When you create a new skin, Kony Visualizer generates a unique name for it. And while you're welcome to keep the name that Kony Visualizer generates, you might find it more useful and easily recognizable if you rename it. You can rename widgets in two primary ways: using the Skins tab of an individual widget, and using the Skins tab on the Project Explorer, where the skins of all widgets are listed.

To rename a skin using the Skins tab on the Project Explorer, do the following:

1. On the Project Explorer, click the **Skins** tab.
2. On the **Skins** tab, locate the type of widget that the skin you want to rename is categorized under. If the skins currently don't appear under the type of widget, click the widget type to open the category and display the skins.
3. From the list of skins, hover over the skin you want to rename and click its context menu arrow to open the skin's context menu.
4. From the skin's context menu, click **Rename**.
5. Type a new name for the skin, and then press **Enter**.

To rename a skin using the Skins tab of an individual widget, do the following:

1. On the Visualizer Canvas, locate a widget that's using the skin you want to rename, and then select it.
2. in the Properties Editor, click the **Skins** tab.
3. In the Name text box, type a new name for the skin, and then press **Enter**.

Set the Skin Background

Kony Visualizer gives you a variety of ways to configure the background of a skin, which are described in the following table.

> ***Note:*** Not all background options are available for all widgets.

Background Type	Description
Single Color	Applies a uniform, single color as the background of the skin that you choose.
Two Step Gradient	Applies two equally-applied colors as the background of the skin that you choose.
Image	Applies an image of your choosing as the background of the skin. The image stretches to fill the dimensions of whatever widget the skin is applied to.
Multi Step Gradient	Applies two or more colors as the background of the skin that you choose. A color can take up whatever percentage of the background you want. You can also adjust the opacity of each color, and you can change the orientation of the colors on a 360 degree axis.

To set the skin background, do the following:

1. On the Visualizer Canvas, locate a widget that's using the skin you want to change the background for, and then select it.
2. On the Properties pane of the widget, click the **Skins** tab.
3. Click the state that you want the background to apply to, as follows:

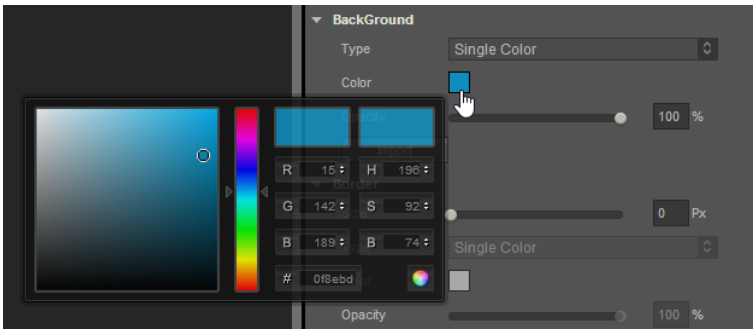
State	Description
Normal	Refers to the state the widget is in when it is at rest and not being engaged by the user.
Focus	Refers to the state of the widget in instances when it is tabbed to or hovered over. This state is not available for all widgets.
Blocked UI	Refers to the state of the widget when it is still visible to the user but is inactive. Designers customarily give a faded background to a Blocked UI to indicate to the user that it isn't currently available, such as a Sign In button being faded because the User Name and Password fields are empty. This state is not available for all widgets.

Pressed	Refers to the state of the widget when it is being pressed. For example, changing the background as a button widget is being pressed provides visual feedback to the user that the button is working and that they successfully pressed the button. This state is not available for all widgets.

4. In the Background section of the Skins tab, click the Type drop-down list and select the type of background you want. Instructions for how to configure each type are found in the following sub procedures.

Single Color

1. Click the square color icon to open the Color Palette window.

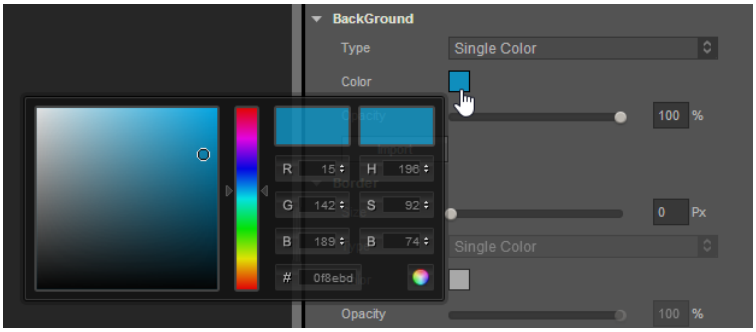


2. To select a particular hue, on the vertical color bar drag the two opposing arrows to the hue you want, and then change the hue's lightness, darkness, and saturation by dragging to the desired location on the large color square. You can also assign a color by changing the RGB values, the HSB values, or pasting a hexadecimal value from another program into the **#** text box. To close the Color Palette window, click anywhere outside of it.

3. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

Two Step Gradient

1. Click the color preview bar to open the Two-Step Gradient Editor.
2. Click the left gradient icon.
3. Click the square of color next to **Color** to open the Color Palette window.



4. To select a particular hue, on the vertical color bar drag the two opposing arrows to the hue you want, and then change the hue's lightness, darkness, and saturation by dragging to the desired location on the large color square. You can also assign a color by changing the RGB values, the HSB values, or pasting a hexadecimal value from another program into the **#** text box. To close the Color Palette window, click anywhere outside of it.
5. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or click the arrow just to the right of the text box to display the opacity slider, and then drag the opacity slider to the degree of opacity that you want.
6. Click the right gradient icon, and then repeat steps 3 through 5 for the right gradient step.
7. When you have completed configuring the colors and their opacity, click **Apply**.

Image

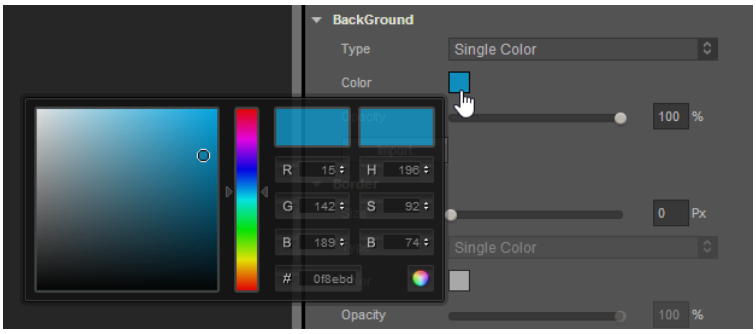
1. Copy the images you want to use to the following path in your workspace:
   `<Workspace Name>\<App Name>\resources\common`

2. On the Project Explorer, click the Assets tab, and then click the **Refresh** button to bind any newly added images to the project.
3. On the Skins tab of the widget you are changing the background for, in the Background section, click the Type drop-down list and select **Image**.
4. Click **Choose Image**. The Background Image dialog box displays.
5. From the Background Image dialog box, select the platforms you want to use the background image for. The image you select for the Default platform option is used if you don't specify a custom image for a specific platform.
6. For each platform option you selected, click its corresponding Value field. Doing so brings up the Select Image dialog box. Click the image you want to use, and then click **OK**, and then click **OK** again.

   The image you selected displays as the background, and stretches to fill the dimensions of whatever widget the skin is applied to. The file name of the image is also listed as the image being used in the Background section of the Skin tab in the Properties Editor.

Multi-Step Gradient

1. Click the color preview bar to open the Multi-Step Gradient Editor.
2. Click one of the gradient step icons.
3. Click the square of color next to **Color** to open the Color Palette window.



4. To select a particular hue, on the vertical color bar drag the two opposing arrows to the hue you want, and then change the hue's lightness, darkness, and saturation by dragging to the desired location on the large color square. You can also assign a color by changing the RGB values, the HSB values, or pasting a hexadecimal value from another program into the **#** text box. To close the Color Palette window, click anywhere outside of it.

5. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or click the arrow just to the right of the text box to display the opacity slider, and then drag the opacity slider to the degree of opacity that you want.

6. Repeat steps 3 through 5 for the other gradient steps.

7. Add additional gradient steps by clicking anywhere just below the color bar, and you can delete a step by selecting it and then clicking **Delete**.

8. Change the portion of the background that the gradient color affects by dragging its gradient step icon to a different position on the color bar.

9. Change the orientation of the colors by trying out the Angle options, and then press **Enter** to apply the change. The available orientations are as follows:

Angle	Description
To top ↑	Orients the bottom color to be the topmost color.
To right →	Orients the bottom color to be the rightmost color.
To bottom ↓	Orients the bottom color to be the bottommost color.
To left ←	Orients the bottom color to be the leftmost color.
Custom	Orients the bottom color to the degree you specify in the numeric field that appears, along a 360 degree axis. For example, 225 orients the bottom color half way between the bottom angle (180), and the left angle (270).

10. When you have completed configuring the colors and their opacity, click **Apply**.

Set the Skin Border

The Border properties on the Skins tab become activated when the border width is greater than 0. Upon setting the border size to 1 or greater, the other border properties become editable. For the border color you can pick a single color or a multi-step gradient. And you can change its opacity.

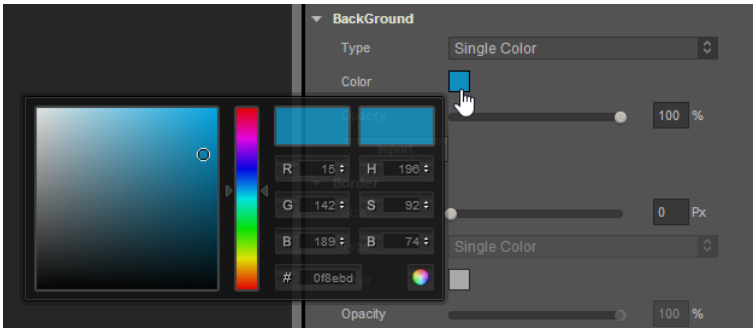
Configuring the skin border for an SPA environment is slightly different. For more information, see SPA Borders.

To configure the skin border, do the following:

1. In the Border section of the Skins tab, set the Size property to at least 1 pixel by either entering a number in the Px text box, or by moving the Size slider to the desired position.
2. In the Border section of the Skins tab, click the Type drop-down list and select the type of border you want, either Single Color or Multi Step Gradient. Instructions for how to configure each type are as follows:

   Single Color

   i. Click the square color icon to open the Color Palette window.



   ii. To select a particular hue, on the vertical color bar drag the two opposing arrows to the hue you want, and then change the hue's lightness, darkness, and saturation by dragging to the desired location on the large color square. You can also assign a color by changing the RGB values, the HSB values, or pasting a hexadecimal value from another program into the # text box. To close the Color Palette window, click anywhere outside of it.

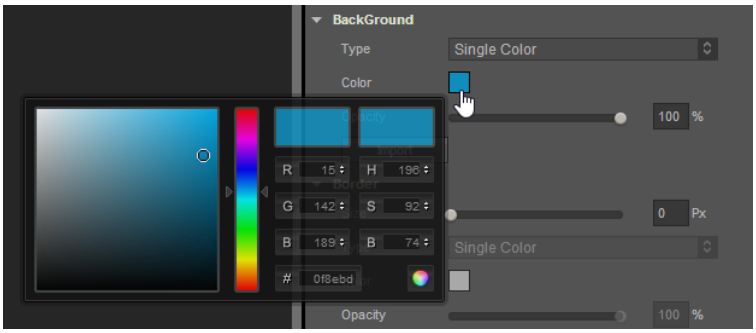
iii. By default, the opacity is set to 100, making the background completely opaque with no transparency. However, if you want the background to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or drag the opacity slider to the degree of opacity that you want.

Multi Step Gradient

i. Click the color preview bar to open the Multi-Step Gradient Editor.

ii. Click one of the gradient step icons .

iii. Click the square of color next to **Color** to open the Color Palette window.



iv. To select a particular hue, on the vertical color bar drag the two opposing arrows to the hue you want, and then change the hue's lightness, darkness, and saturation by dragging to the desired location on the large color square. You can also assign a color by changing the RGB values, the HSB values, or pasting a hexadecimal value from another program into the **#** text box. To close the Color Palette window, click anywhere outside of it.

v. By default, the opacity is set to 100, making the border completely opaque with no transparency. However, if you want the border to have a degree of transparency, you can decrease its opacity. To do so, type a value between 0 and 100 in the Opacity text box, or click the arrow just to the right of the text box to display the opacity slider, and then drag the opacity slider to the degree of opacity that you want.

vi. Repeat steps 3 through 5 for the other gradient steps.

vii. Add additional gradient steps by clicking anywhere just below the color bar, and you can delete a step by selecting it and then clicking **Delete**.

viii. Change the portion of the border that the gradient color affects by dragging its gradient step icon to a different position on the color bar.

ix. Change the orientation of the colors by trying out the Angle options, and then press **Enter** to apply the change. The available orientations are as follows:

Angle	Description
To top ↑	Orients the bottom color to be the topmost color.
To right →	Orients the bottom color to be the rightmost color.
To bottom ↓	Orients the bottom color to be the bottommost color.
To left ←	Orients the bottom color to be the leftmost color.
Custom	Orients the bottom color to the degree you specify in the numeric field that appears, along a 360 degree axis. For example, 225 orients the bottom color half way between the bottom angle (180), and the left angle (270).

x. When you have completed configuring the colors and their opacity, click **Apply**.

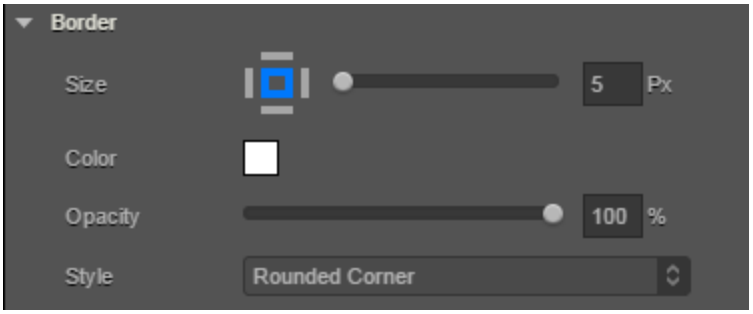
3. In the Border section of the Skins tab, click the Style drop-down list and select the type of corners you want, either Plain, Rounded Corner, Complete Rounded Corner, or Custom. Instructions for how to configure each type are found in the following table.

Border Style	**Instructions**
Plain	• Plain border corners have a sharp 90 degree angle. To give a border plain border corners, click the Style drop-down list, and then select **Plain**.
Rounded Corner	Rounded border corners with an opacity of 0 look like this:  • Click the Style drop-down list, and then select **Rounded Corner**.

Complete Rounded Corner	Complete Rounded border corners with an opacity of 0 look like this:  • Click the Style drop-down list, and then select **Complete Rounded Corner**.
Custom	With Custom border corners you can round the corners as deeply or shallowly as you want. 1. Click the Style drop-down list, and then select **Custom**. 2. Set the Custom rounded property between 1 and 100 pixels by entering a number in the Px text box, or by moving the Radius slider to the desired position.

SPA Borders

SPA borders can be applied when you have forked a form to the SPA platform. For more information, see Forking a Form.



You can modify the following border properties of a widget on an SPA platform:

1. **Size**: When you fork a widget skin for an SPA platform, you can either set unique values for each side of the widget border or set the same value for all the border sides.

   To set unique values for each border side, follow these steps:

1. Click the left border icon (|). The icon changes from gray to blue.
2. Either type the value or using the slider, set the left border size.
3. Perform the above steps for all the border sides.

To set identical values for all border sides, do the following:

1. Click the border indicator  . The indicator changes from grey to blue.
2. Either type the value or using the slider, set the uniform border values.

> ***Note:*** Border values are not retained if you unfork a widget skin.

2. **Color**: To modify the border color, do one of the following:
   - Update the border color of the widget.
   - Reuse an existing color. For more information, see Copy and Paste a Color.
3. **Opacity**: Change how opaque or transparent the border color is.
4. **Style**: When you define the border size, the Style property is enabled. The following types of styles are available for widgets:
   - **Plain**: Widget corners have a standard, 90-degree angle.
   - **Rounded Corner**: Widget corners are rounded rather than having a sharp, 90-degree angle. If you select this option, you can either set unique values for each corner of the widget border or set the same value for all the border corners.

     To set unique values for each corner, do the following:

1. Click the left-top corner icon ( ). The icon changes from grey to blue.
2. Either type the value or using the slider, set the left-top corner size.
3. Perform the above steps for all the border corners.

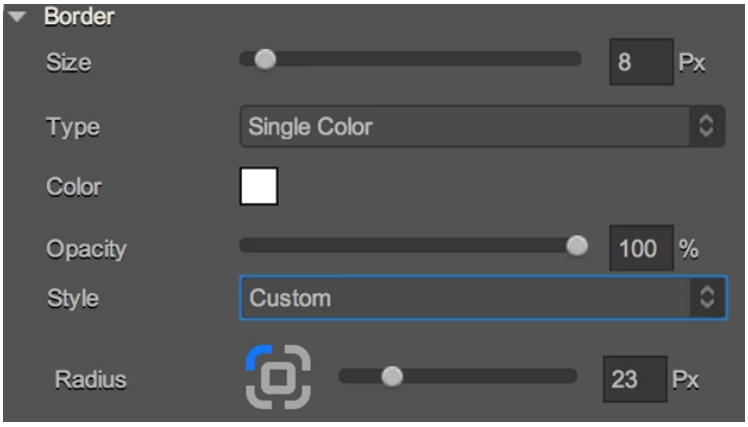
To set identical values for all border corners, do the following:

1. Click the corner indicator . The indicator changes from grey to blue.
2. Either type the value or using the slider, set the uniform corner values.

Set Individual Corners in Borders

Rounded edges can be applied for widgets on skins forked for Android and Windows native platforms.

> ***Note:*** Android/Windows native platforms support configuring individual corner radius. Configuring border properties for each of the four edges of a given widget is not supported by Android/Windows native platforms.



To set unique values for each corner, do the following:

1. Click the border indicator  . The indicator changes from grey to blue.
2. Either type the value or using the slider, set the uniform border values.
3. Border values are not retained if you unfork a widget skin.
4. Perform the above steps for all the border corners.

Set the Skin Shadow

> ***Note:*** Shadow feature is available only when a skin is forked.

You can modify the following shadow properties of a widget:

1. **Dist X**: Denotes the horizontal shadow distance away from the widget. When a positive **Dist X** value is entered the shadow is moved to the right-side of a widget. When a negative value is entered the shadow is moved to the left-side of a widget.
2. **Dist Y**: Denotes the vertical shadow distance away from the widget. When a positive **Dist Y** value is entered, shadow is moved to the top-side of a widget. When a negative value is entered the shadow is moved to the bottom-side of a widget.
3. **Blur**: Denotes the shadow blur of the widget by allowing you to soften shadow.
4. **Color**: Denotes the color of the shadow. You can select a shadow color by clicking the **Color Picker**.

5. **InnerShadow**: Click this check box if you want the shadow to display inside the widget. Otherwise, clear the check box to display the shadows outside the widget.

Set the Skin Font

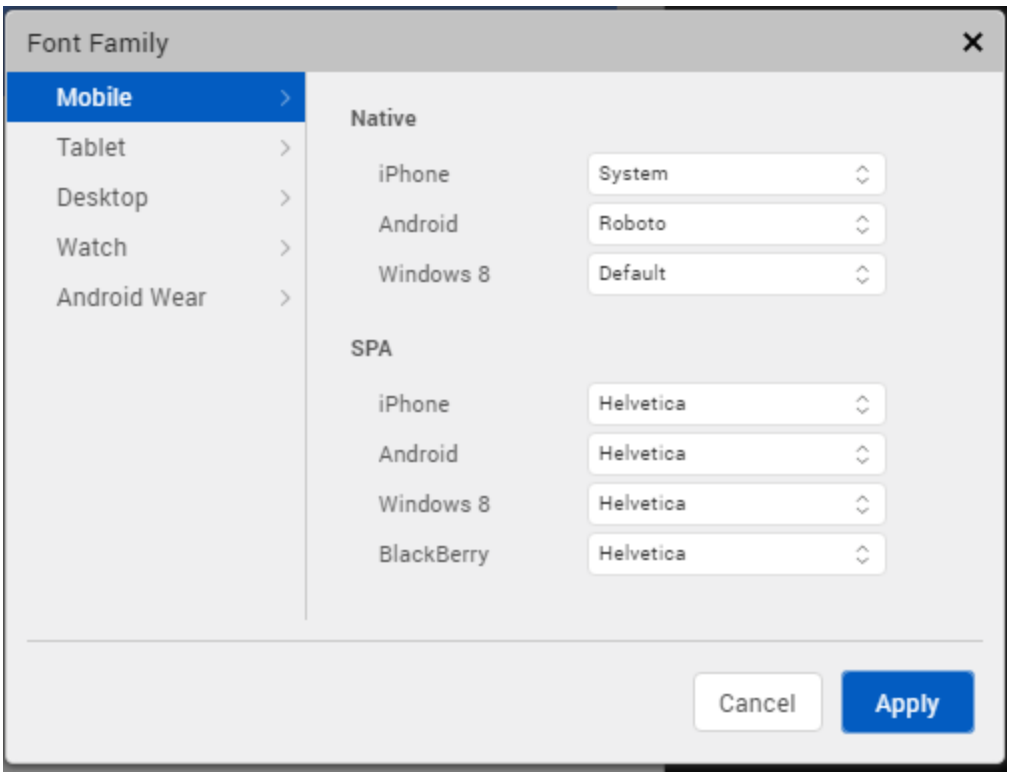
You can modify the following font properties of a widget:

- **Color**: Denotes the font color. To modify the font color, do one of the following:
    - **Select a color.** To do so, click the square color icon to open the Color Palette window. To select a particular hue, on the vertical color bar drag the two opposing arrows to the hue you want, and then change the hue's lightness, darkness, and saturation by dragging to the desired location on the large color square. You can also assign a color by changing the RGB values, the HSB values, or pasting a hexadecimal value from another program into the **#** text box. To close the Color Palette window, click anywhere outside of it.
    - **Reuse an existing color.** For more information, see Copy and Paste a Color or Gradient.
- **Opacity**: You can change the opacity level of the font color.
- **Size**: You can set the font size by pixels (0 to 600) or as a percentage (0 to 600) of the baseline font size of 28 pixels.

    > ***Note:*** The Kony framework, however, consumes only percentage-based values for font size. As the px values are purely representational, you must use percentage-based values while setting the skin's font size.

    > ***Note:*** In iOS devices with OS 13, for certain font sizes, you will observe spacing.

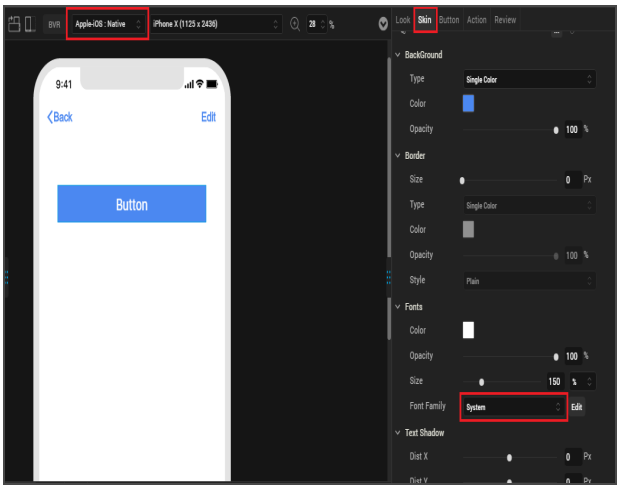
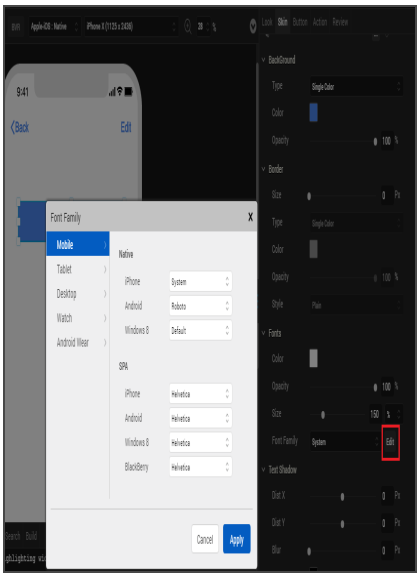
- **Font Family**: Click the **Edit** button of the **Font Family** property to open the **Font Names** window.



Here, you can select the fonts for individual platforms for both Mobile and Tables

- **Font Family**: You can either assign a default font family for all platforms by selecting a font from the **Font Family** drop-down list, or assign a font by platform using the **Edit** button. In assigning fonts by platform, you can use either of the two following methods:

Assign a font to a single platform	Assign a font to multiple platforms
To assign a font to an individual platform, do the following:	To assign a font to multiple platforms, do the following:

Assign a font to a single platform	Assign a font to multiple platforms
	
1. On the **Visualizer Canvas**, from the **Platform** drop-down list, select a platform. 2. From the widget **Properties** pane, click **Skin** tab. 3. From the **Font Family** drop-down list, select a font. 4. Save the project. Repeat the above steps for assigning fonts to other platforms.	1. Click **Edit**, next to the Font Family drop-down list. 2. On the **Mobile** tab, select the required font for each of the native and SPA platforms. 3. On the **Tablet** tab, select the required font for each of the native and SPA platforms. 4. On the **Desktop** tab, select the required font for Web and Windows platforms. 5. Click **Apply**.

- **Weight**: Select whether text on the widget should be normal or bold.
- **Style**: Select whether text on the widget should be underlined or italicized.

Set the Skin Font Shadow

Configure the text properties of a skin using Distance X, Distance Y, Blur, and Color properties -

> ***Note:*** The Text Shadow feature is available only for specific widgets such as Button, Label, and TextBox2.

You can modify the following text shadow properties of a widget:

a. **DistX**: Denotes the horizontal shadow distance away from the widget text.
   - A positive value places the text shadow to the right of the text.
   - A negative value places the text shadow to the left of the text.

b. **DistY**: Denotes the vertical shadow distance away from the widget.
   - A positive value places the text shadow below the text.
   - A negative value places the text shadow above the text.

c. **Blur**: Denotes the shadow blur of the widget and allows you to soften text shadow.

d. **Color**: Denotes the color of the text shadow. You can select a text shadow color by clicking the **Color Picker**.

Location of Skin Files

Skins are grouped into a theme and the files are available in the themes folder in your application's workspace. See the following folder structure:

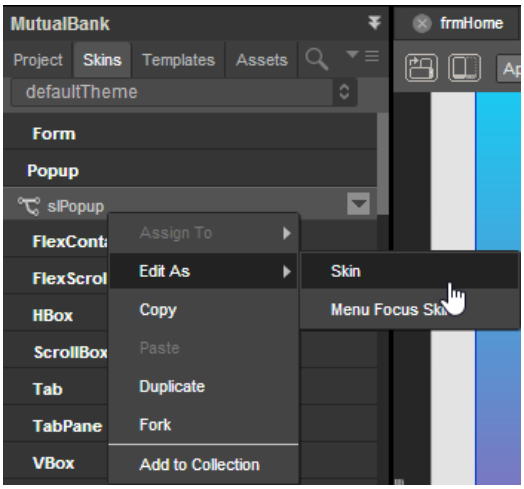
```
<Workspace><your application name><themes>
```

For example, if your workspace is `C:/Users/your username/KonyVizEWS` and your application name is `Application1`, then the themes are located at: `C:/Users/username/KonyVizEWS/Application1/themes`.

Edit a Skin

To edit a skin you created in Kony Visualizer, do the following:

1. Right-click the skin, and select **Edit As**.



2. Select an option from the menu. The options are:
    a. **Skin**: Skin of the widget when it is not focused.
    b. **Menu Focus Skin**: Skin of the widget when it is focused by the user.
3. Configure the skin as required.
4. Save the skin. After you save the skin, a new skin file is created in Project Explorer. You can use this skin for widgets in your application.

Apply a Skin

To apply a skin to a widget, do the following:

1. Select a widget from the Visualizer Canvas.
2. In the Properties pane, click the Skins tab.
3. Select the state of the widget that you want the skin to apply to, such as Normal or Focus.
4. Click the Search icon to the right of the Name text box. The drop-down list that appears lists all the available skins. To filter out unwanted skins from the list, begin typing the name of the skin you want.
5. From the drop-down list of skins, select the skin you want to use.
6. Go to **Skins** tab.
7. Navigate to your widget of choice.
8. Right-click and select **Assign To > Skin / Focus Skin**. If you select **Skin**, the skin will be applied to the normal state of the widget. If you select **Focus Skin**, the skin will be applied to the focused state of the widget.

Reuse Skins

Skins can be used to display different views of widgets in different platforms.

- **Import**: Allows you to import a CSS file to be used as skin. This action can be performed using CSS Hat plug-in and Adobe Photoshop. You can find the list of CSS elements supported for import here.
- **Copy**: Allows you to copy the current skin of a widget to a clipboard.
- **Paste**: Allows you to paste the copied skin to a selected widget.

> ***Note:*** Copying and pasting a skin creates another version of the original skin. The pasted

skin becomes an independent skin. Any changes made to the original skin do not affect the pasted skin.

Note: The paste function can be applied across the widget states. For example, a Normal state skin of a Button Widget can be copied and pasted to the Focus skin state of the Button Widget.

- **Assign**: Allows you to assign the copied skin to a widget.
- **Duplicate**: Allows you to duplicate the current skin.

Differences between the Copy-Paste and Copy-Assign functions:

- When a skin is copied and pasted, a new skin is created. However, when a skin is copied and assigned, no new skin is created.
- Changes to the original skin do not affect the pasted skin. However, changes made to the original skin affects the assigned skin (because the copied and assigned skin are one and the same).

Supported CSS Properties for Import

CSS Rule	CSS Values
background-image	linear-gradient, -webkit-linear-gradient Multiple gradients are not supported. For gradients that have two color stops, angles other than 0, 90, 180, and 270 are not supported.
background	
-webkit-background-image	
-webkit-background	
background-color	All
opacity	All

CSS Rule	CSS Values
border-radius	All
-webkit-border-radius	All
border	All
border-color	All
border-width	All
color	All
font-size	All
font-weight	bold, normal
font-style	italic, none
text-decoration	underline, line-through, overline, none
text-shadow	All
box-shadow	All
-webkit-box-shadow	All

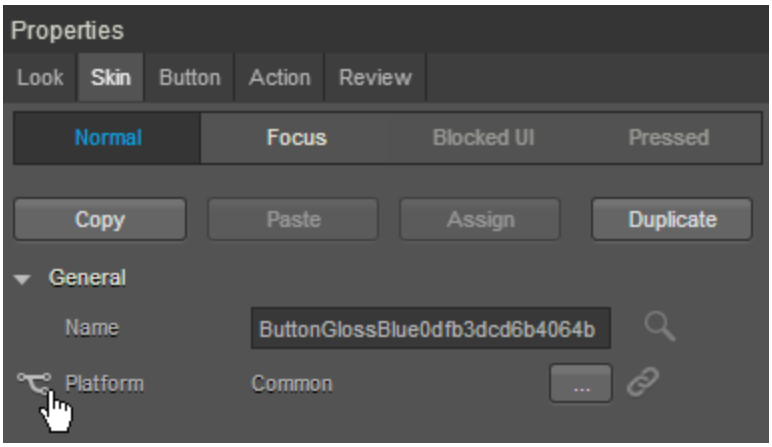
Fork a Skin

Through forking you can provide different values for the same skin across different platforms. However, only certain widget properties can be forked. You can identify if a property can be forked by the presence of a wishbone-shaped icon to the left of the property's name. Two types of forking are available:

Simple Forking. Fork one platform at a time.

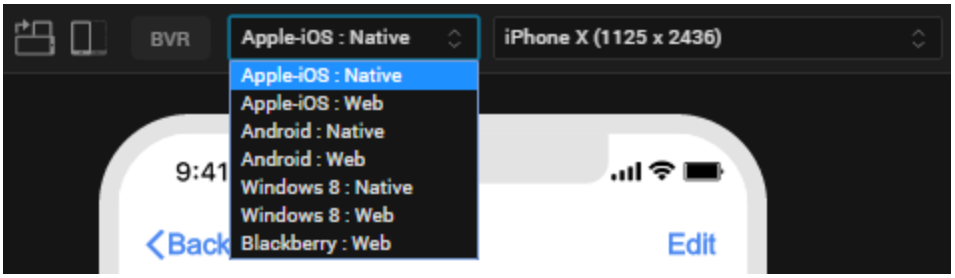
Complex Forking. Forks multiple platforms of your choosing at the same time.

You can configure a skin and make it common to all platforms or apply it to a particular platform using the fork option in the **Properties > Skins** tab.



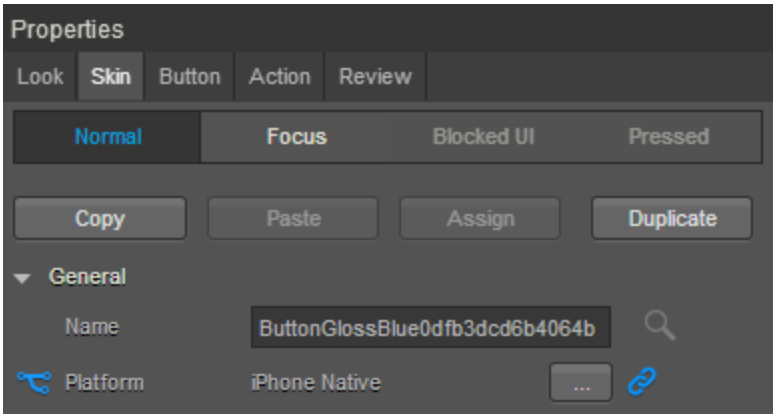
To fork a skin, do the following:

1. From the Application Canvas, select the platform for which you want to fork.



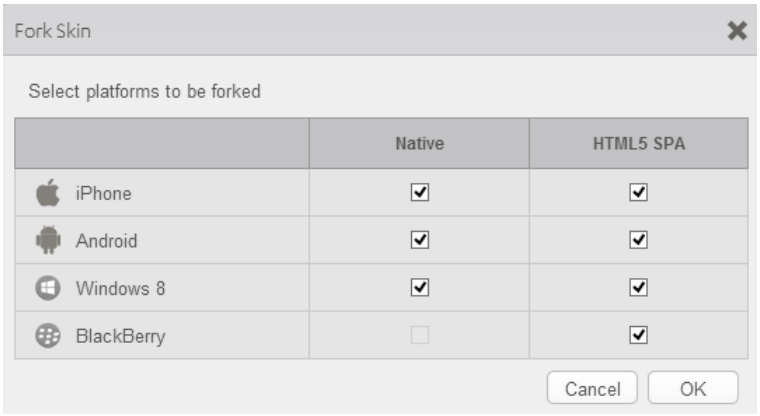
2. In the Properties Editor on the **Skin** tab, in the General section, click the **Fork** icon to fork the skin for a selected platform. If the option is selected, the fork icon turns blue. The platform for which the skin is forked also appears at the platform.

> ***Note:*** You can also go to the Skins tab in Project Explorer, and right-click a skin to fork the skin.



3. Click the lock icon to lock a skin. If you lock a skin, changes made to a widget's skin in one platform are applied to the remaining platforms.

4. Click the ellipsis icon. The Fork Skin dialog appears.



> ***Note:*** For widgets like Calendar, which has the View Type property while forking. You can set the View Type property to different values.

5. Select the platforms to fork and click **OK**. The skin will be forked for selected platforms.

Important Considerations in Forking

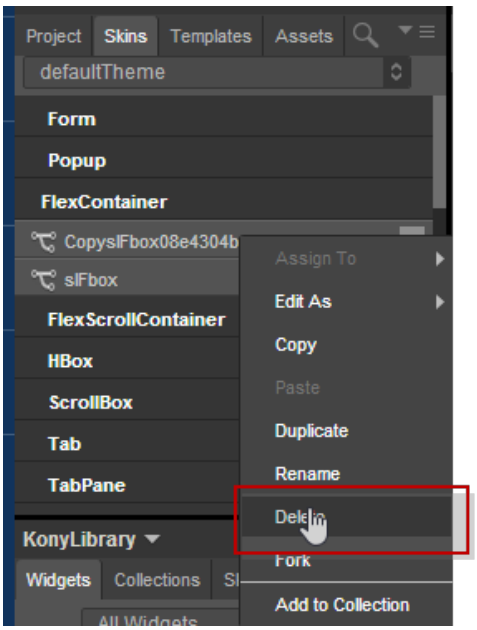
- Widgets that have background type as Multi-Step Gradient are by default forked (for example, Button and Phone Widgets).

- When you fork and lock a widget skin, changes made to a widget's skin in one platform get applied to the remaining platforms, unless you specifically, unlock a platform.
- To provide a unique skin for a platform, you apply fork and unlock properties for that platform.
- By default all the widgets (except widgets that have background type as Multi-Step Gradient), will have a **Common** skin. Changes made to the Common skin will be applied across the platforms.
- You can select the platforms to be forked by clicking the ellipsis button. From the **Fork Skin** dialog, select the desired platforms to be forked.

Delete a Skin

To delete a skin from the project library in Kony Visualizer, do the following:

1. Go to the **Skins** tab in the Project Explorer.
2. Select the widget that has the skin you want to delete.
3. Right-click the skin, and then click **Delete**.



4. Select **Yes** in the confirmation window that appears. The selected skin is deleted.

> ***Note:*** If a skin is deleted, widgets attached to the skin revert to the default skins.

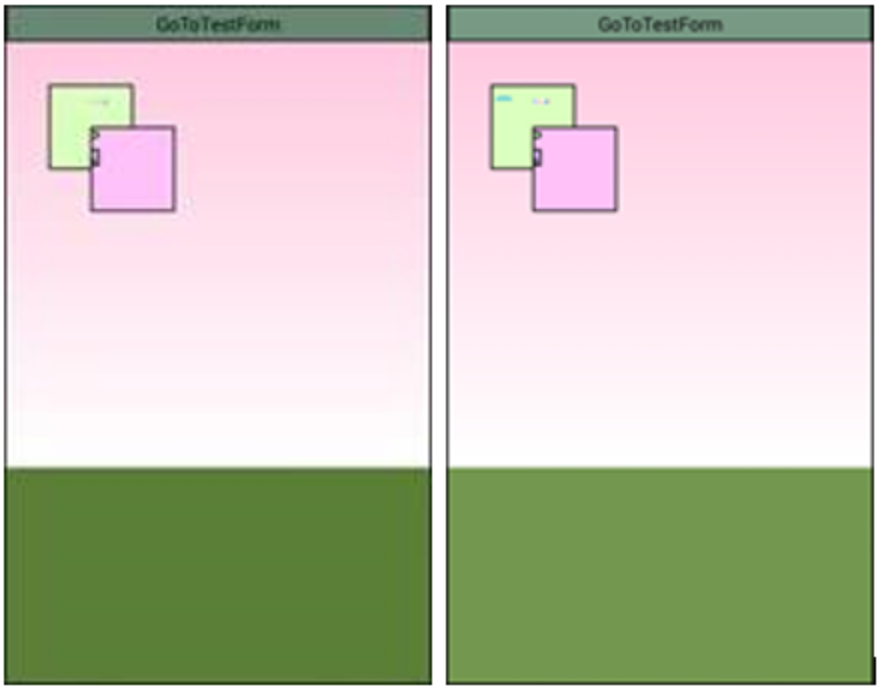
Impact of Upgrade to Kony Visualizer V8 on Skins in Widgets

When you upgrade from any previous version of Kony Visualizer to Kony Visualizer V8, for widgets which don't have any skins or themes applied on them, the following changes may be noticed.

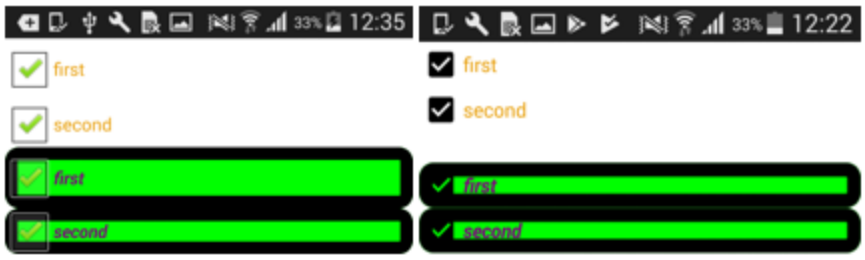
- Button/Label themes change

- Height and width of buttons change.

- Background color of the form/flex changes to a lighter shade when we have the opacity set for Form/Flex which is present as root widget in form.

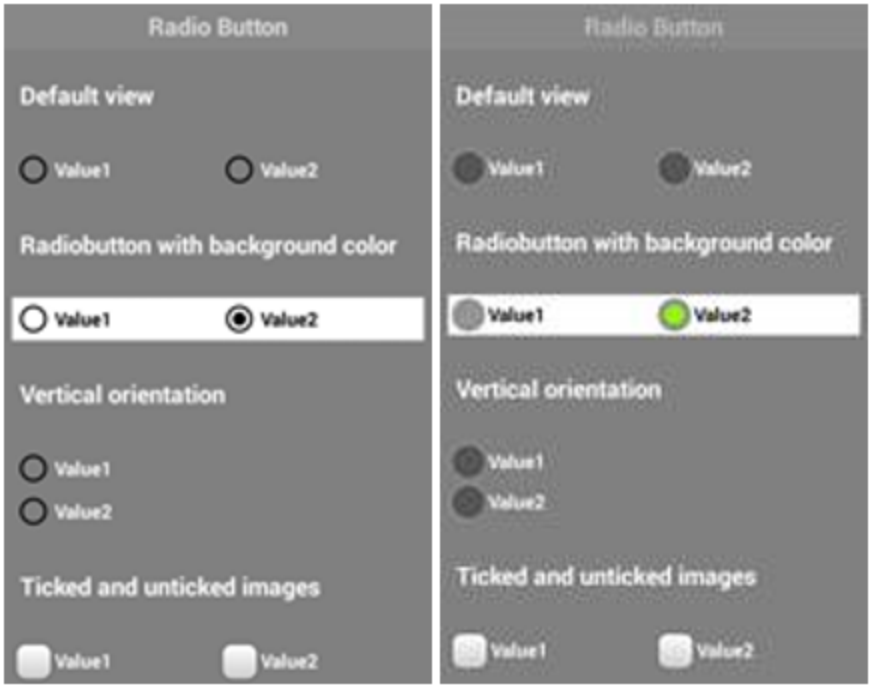


- Checkbox group and radio button group default selection color is changed to black.



- Combobox opened list view background color will change.

- Radio button default value selected color will change



Importing Photoshop Styles and Colors

With Kony Visualizer, you can take advantage of styles that you have already created in Photoshop. You can also copy the hexadecimal value for a color in Photoshop, and then paste that into Kony Visualizer to achieve the same color.

You can use one of the following methods to use a color from Photoshop in Kony Visualizer:

- Import Photoshop Styles
- Copy a Photoshop Color

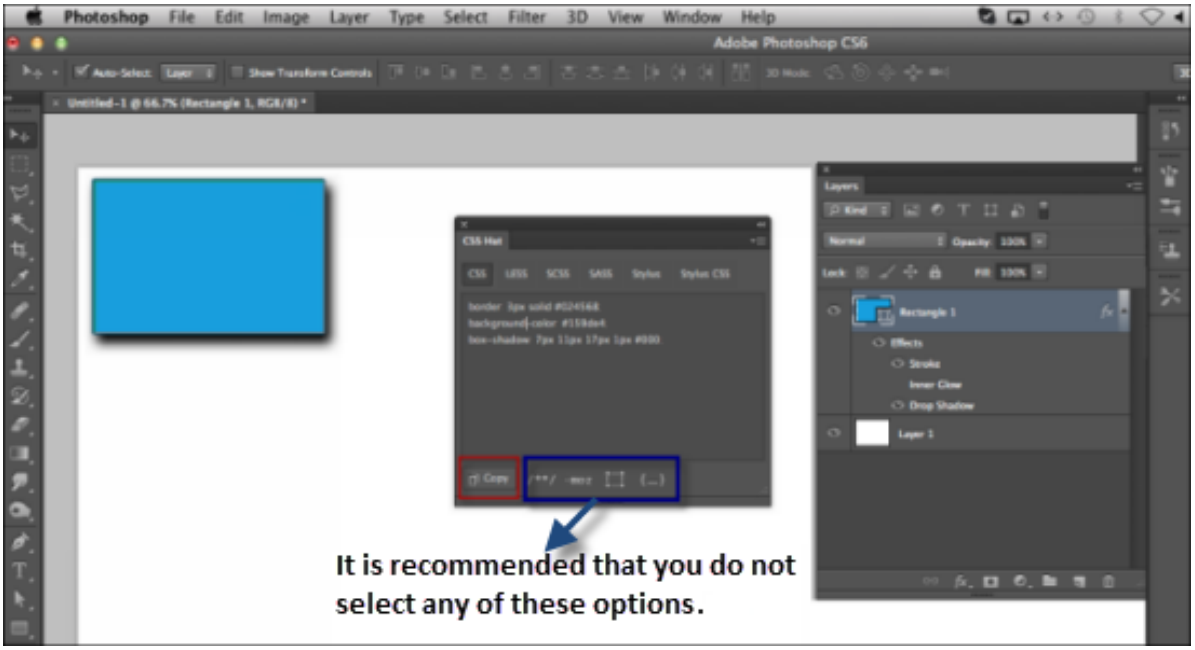
Import Photoshop Styles

You import Photoshop styles into Kony Visualizer by converting them into a cascading style sheet (CSS) file using the CSS Hat plug-in for Photoshop, and then importing that CSS file into Kony Visualizer. If you do not already have the CSS Hat plug-in for Photoshop, you can install it from the CSS Hat website.

To import Photoshop styles into Kony Visualizer, follow these steps:

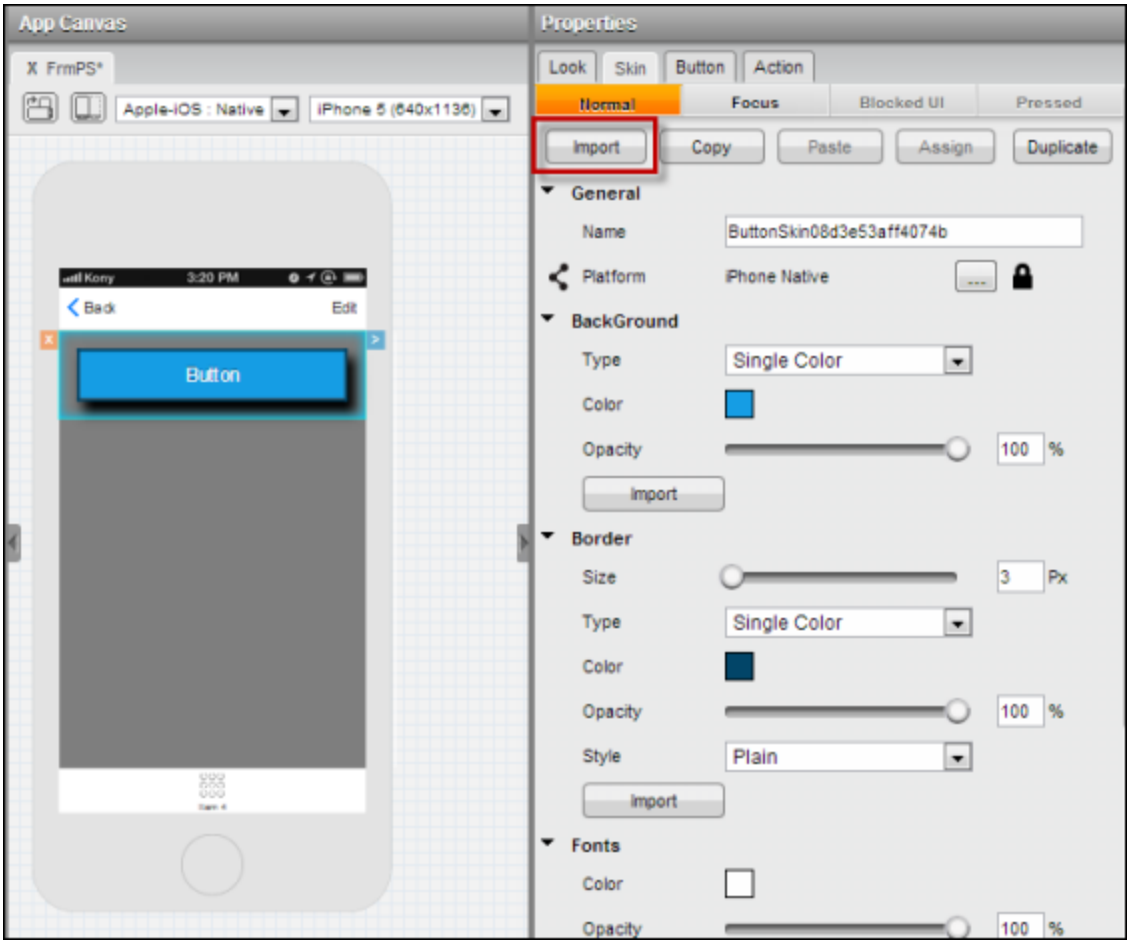
1. In Photoshop, select the layer you want to export to Kony Visualizer.
2. Open the CSS Hat extension, and ensure that the equivalent CSS properties are populated. Click **Copy**. The CSS values are copied to the clipboard.

   > ***Note:*** Do not select any of these options  in the CSS Hat palette.

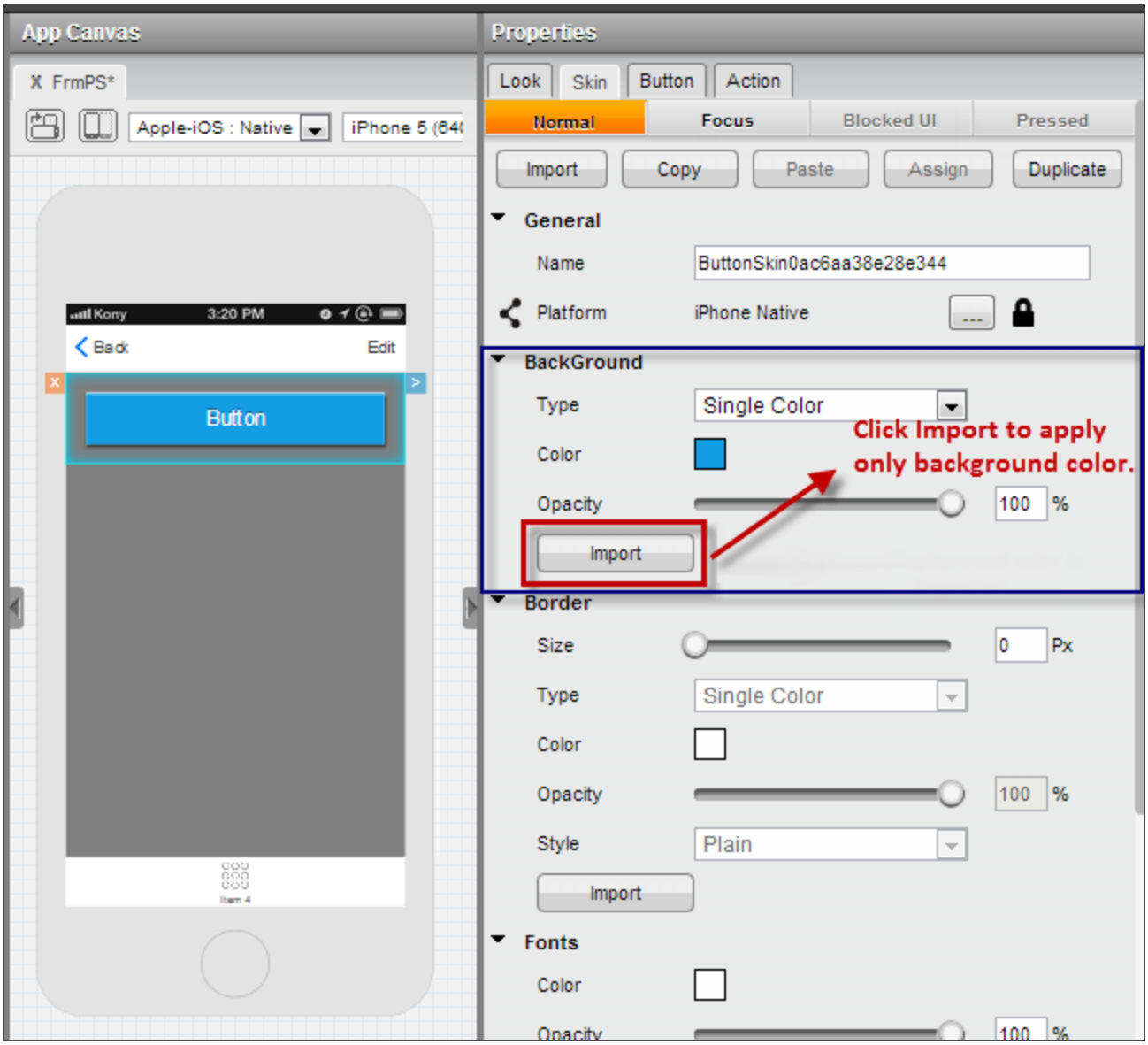
   

3. In Kony Visualizer, select a widget to which you want to apply Photoshop styles.
4. On the **Properties** tab, click **Skin**.
5. Select a skin state such as Normal, Focus, or Blocked UI.

6. To import all the Photoshop styles (such as background, border, and shadow), click **Import**. The widget skins are updated based on the Photoshop CSS.



Selective Photoshop CSS import: To import a selective Photoshop style (such as a background, a border, or a shadow style), click **Import** on the individual property tabs. Based on the Photoshop style, the individual property is updated.



Naming Conventions

Some of the properties in Kony Visualizer are referred to differently in Photoshop. The following table outlines these differences.

Kony Visualizer	Photoshop
Border	Stroke

Kony Visualizer	Photoshop
Background: Multi-step Gradient Background: Two-step Gradient	Gradient Overlay
Background: Single Color	Color Overlay
Shadow	Drop Shadow
Shadow	Outer Glow
Shadow (Inset)	Inner Shadow
Shadow (Inset)	Inner Glow

Limitations

Not all CSS values are imported, and Kony Visualizer ignores the values that are unsupported. The unsupported CSS values are as follows:

- Font families are not imported from Photoshop. It is recommended that you provide the font family for a widget in Kony Visualizer.
- Borders with a Multi-Step Gradient (MSG) are not properly imported from Photoshop. It is recommended that you provide the border MSG in Kony Visualizer.
- If you create non-uniform borders, you need to import the CSS styles in the Native and Web channels.

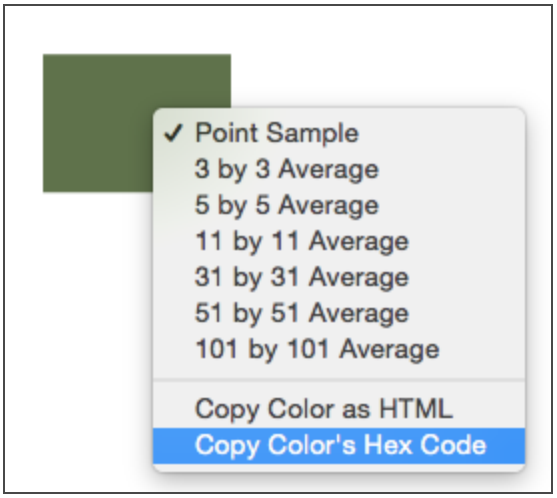
Copy a Photoshop Color

Kony Visualizer gives you the ability to copy a color from Photoshop and assign this color to any of the widget's skin properties such as the Background, Border, Font, Shadow, and Font Shadow.

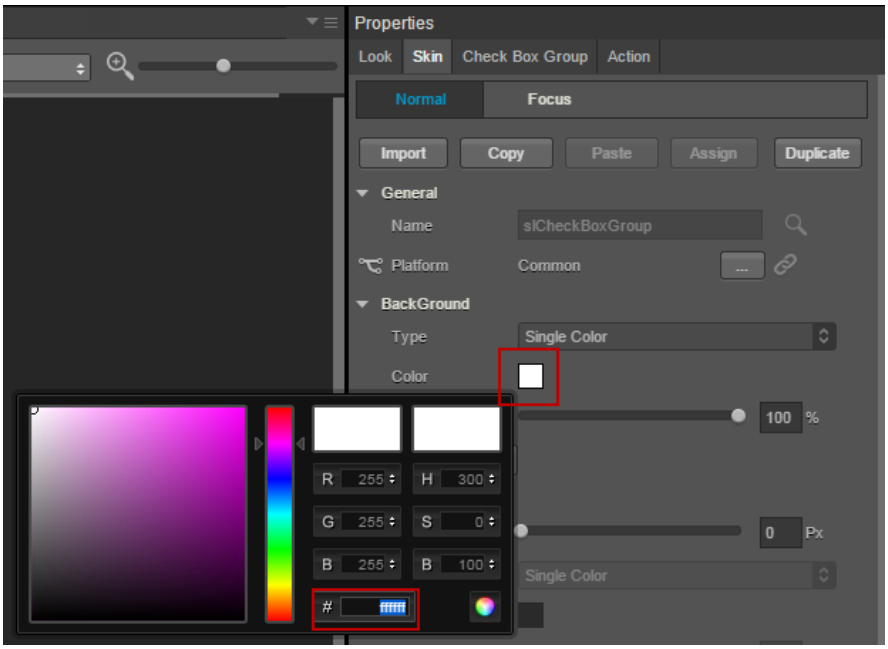
> ***Note:*** Kony Visualizer is not able to copy a two-step or multi-step gradient color from Photoshop.

To copy a color from Photoshop Studio to your widget skin, do the following:

1. In Photoshop Studio, use the **Eyedropper** tool to copy the color's hexadecimal value. To do so, hover the **Eyedropper** tool over the object, right-click, and then click **Copy Color's Hex Code**.



2. In Kony Visualizer, go to the **Skin** tab of a widget's Properties pane, click **color**, and then paste the hex code.



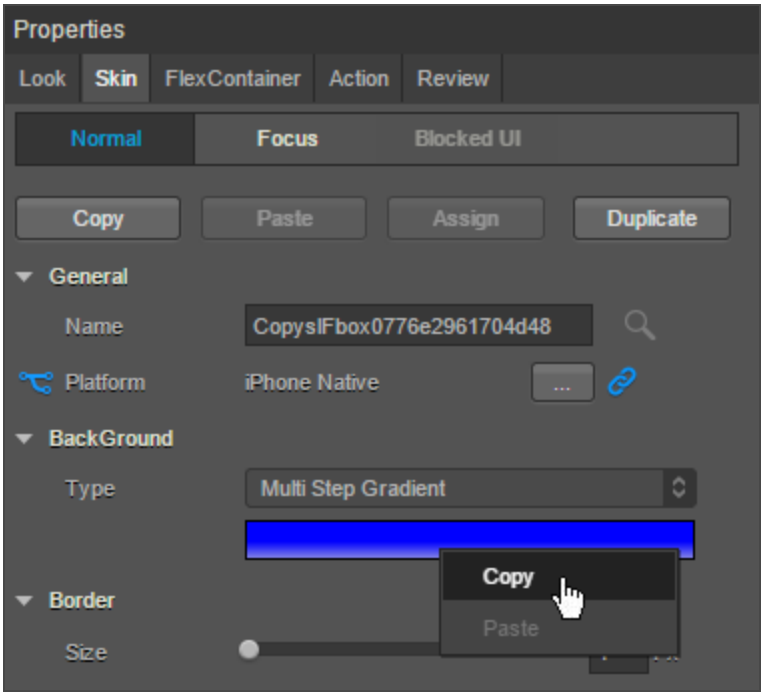
> ***Note:*** You can copy the color to any of the widget's properties such as Background, Border, Font, Shadow, and Font Shadow.

Copy and Paste a Color or Gradient

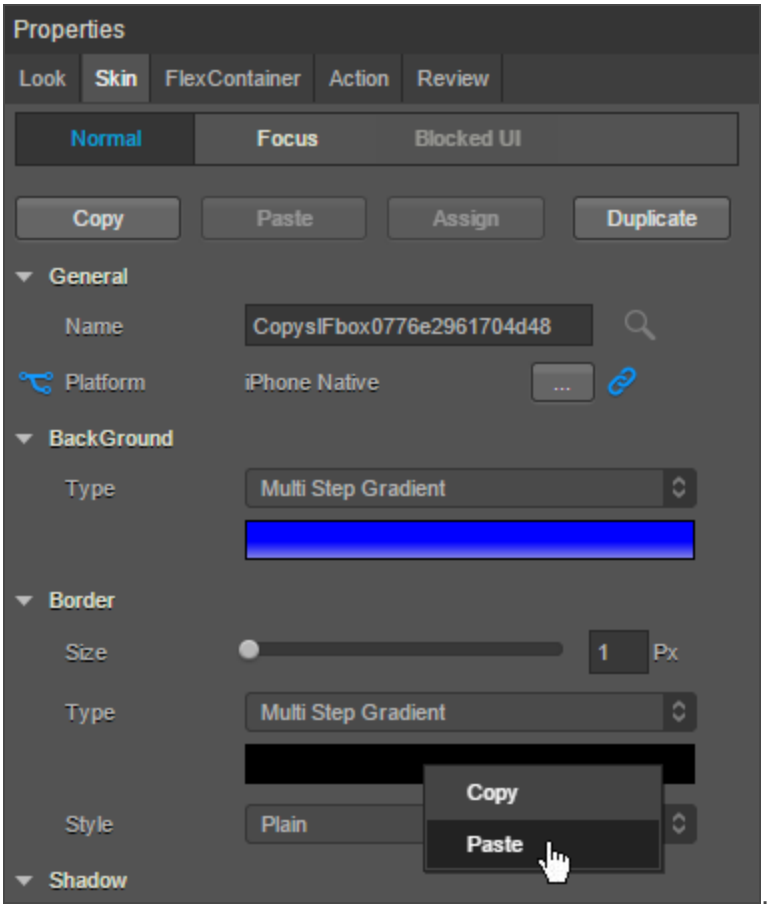
With Kony Visualizer, you can reuse a color or gradient by copying it from one widget property and pasting it to another property, such as from a background to a border. Gradients are especially tedious to replicate, so being able to copy and paste them can save you considerable time and effort.

To reuse a color or gradient, follow these steps:

1. On the Properties tab of a widget, click **Skin**.
2. Right-click the color palette of a property and then click **Copy**.



3. Depending on whether you want to paste the color and gradient within the same widget or another widget, do one of the following:
   - **Within the same widget.** Right-click the color palette of a property, and then click **Paste**.
   - **Different widget.** Navigate to the Skin tab of the widget, right-click the color palette of a property, and then click **Paste**.



Notes:

- The color palette of the copied property and the pasted property should be of the same type. That is, if the copied color is of multi-step gradient, the pasted property color should also be multi-step gradient.
- You can reuse a color across the widgets. That is, you can copy the color of a button widget and paste it to a Flex Container widget.
- You cannot copy and paste an image.

Using Themes - Applying a Collection of Skins as a Group

Kony Visualizer provides a default theme that has a set of skins defined for all the widgets within an application. The skins in this theme can be applied to widgets to get a different look and feel for the widgets. Themes are accessed from the Skins tab of the Project Explorer. A drop-down list at the top of the Skins tab indicates the current theme. When you create a theme, it is created using the default theme as the base. All the skins available in the default theme are copied to the created theme. Kony Visualizer allows you to add, modify, rename, copy, fork, and delete skins within a theme.

> ***Note:*** Any changes you make to the skins within a theme are limited only to that theme.

Kony Visualizer provides a set of APIs that allow you to perform different actions with themes. The theme you create using the Kony Visualizer can only be applied using the `kony.theme.setCurrentTheme` API. For more information about the APIs related to Theme, see Kony API Reference Guide in Kony Documentation Library.

What are themes and why would I want them?

A theme is a collection of skins. By default, Kony provides you with skins for all the widgets. If required, you can configure a new set of skins for your company and group them as a theme. Themes can be exported and imported in Kony Visualizer. Using themes you can apply skins to all the widgets in your application in one single action. For example, if you have configured skins for Button, Form, Label, Segment Widgets based on your company's branding, then you can create a new theme and export the theme. The exported theme can be sent to a designer or developer in your company to maintain consistency while developing the applications.

> ***Important:*** The default Theme and the default skins cannot be modified, deleted, or renamed.

If you attempt to modify a default skin it results in creating a new skin. These new skins can be used within the same project. If you require these skins for other projects, save them in a user-defined theme, and then export them.

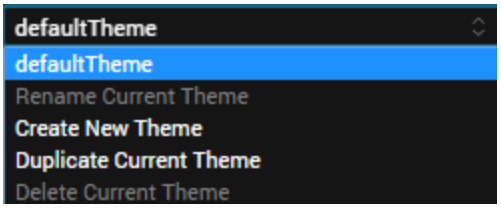
Create a Theme

For storing default as well as modified skins you create a new theme. In the Skin tab of the Project Explorer, click **Default Theme** list and then click **Create New Theme**. A new theme with a default name is created.

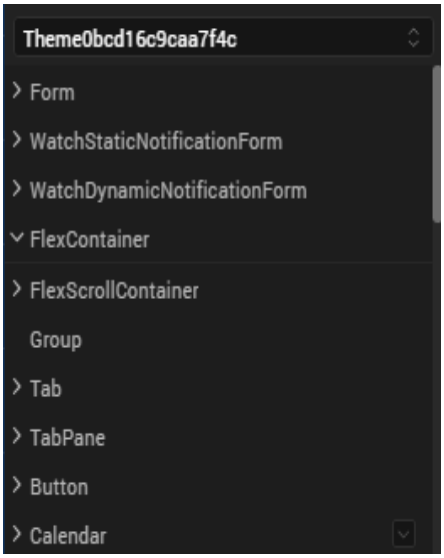
> *Important:* All the skins (default and customized) that are used in the project until the time of creating this new theme are added automatically to the newly created theme.

To create a new theme, navigate to Skins tab in the project explorer.

1. Navigate to **Skins** tab in **Project Explorer**.
2. Click the theme list and select **Create New Theme**.



A new theme will be created with a random name. This theme contains all the skins you have created in Kony Visualizer.



3. Configure the skins as required and save the theme. The skins modified or created will be available for all the applications in Kony Visualizer.

> ***Note:*** The themes you have created in Kony Visualizer appear in the Theme drop-down list and you can easily navigate to other themes using the list.

Location of Theme Files

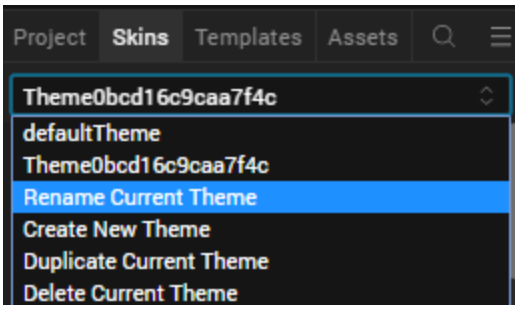
The theme files for a project are located in the following folder:

```
<Workspace>\<ProjectName>\<themes>
```

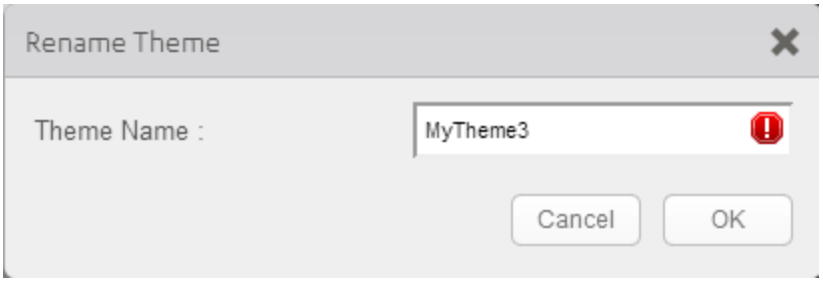
Rename a Theme

To rename a theme, do the following:

1. Navigate to **Skins** tab in **Project Explorer**.
2. Click the theme list and select **Rename**.



The Rename Theme dialog appears.

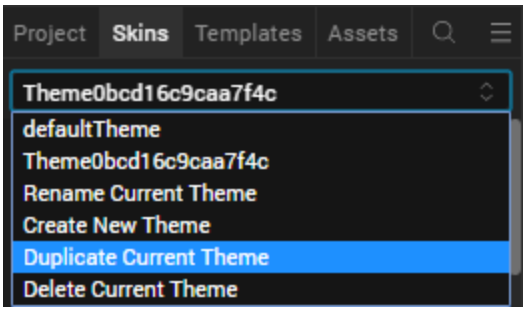


3. Type a name for the name and click **OK**. The theme will be renamed.

Duplicate the Current Theme

To duplicate the current theme, do the following:

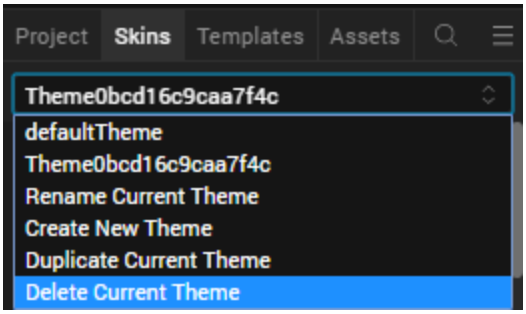
1. Navigate to **Skins** tab in **Project Explorer**.
2. Click the theme list and select **Duplicate Current Theme**. The current theme will be duplicated and a random name is generated. You can rename the theme to a meaningful name.



Delete a Theme

To delete an existing theme, do the following:

1. Navigate to **Skins** tab in **Project Explorer**.
2. Click the theme list and select **Delete Current Theme**. The current theme will be deleted.



> ***Note:*** You can view the theme drop-down list to confirm if the theme is deleted.

Templates Tab

Templates provide an easy way to display the same information across the forms, pop-ups, calendars, segments, or maps.

The following templates are supported on various channels:

Headers

Footers

Segments

Maps

Gridcalendars

Grids

Tab Headers

Context Menus

Menus

Headers

A header is a section of the form that is docked at the top of the form, and can be reused across the forms. You can create multiple headers for a project, but you can assign only one header to a form. You can reuse the headers across the channels. That is, a header created in the Mobile channel can be reused in the Tablet or Desktop channels.

To create a header template, follow these steps:

1. Click the **Templates** tab from **Project Explorer**.
2. Expand **Mobile**, **Tablet**, or **Desktop** (depending on which channel you want to create a header template), point to **Headers** to display a down-arrow. Click this down-arrow and then click **New Template**. Rename the template, if required.
3. Drag and drop a container widget on the template.

   > ***Note:*** A template requires that you add a container widget (for Flex form: **FlexContainer** and for VBox form: **HBox**) before adding other widgets.

4. Drag and drop the required widgets on the container.
5. Set the properties of these widgets and save the header template.

Footers

A footer is a section of the form that is docked at the bottom of the form and can be reused across the forms.

To create a footer template, follow these steps:

1. Click the **Templates** tab from **Project Explorer**.
2. Expand **Mobile**, **Tablet**, or **Desktop** (depending on which channel you want to create a map template), point to **Footers** to display a down-arrow. Click this down-arrow and then click **New Template**. Rename the template, if required.

3. Drag and drop a container widget on the template.

> ***Note:*** A template requires that you add a container widget (for Flex form: **FlexContainer** and for VBox form: **HBox**) before adding other widgets.

4. Drag and drop the required widgets on the container.
5. Set the properties of these widgets and save the footer template.

Adding a Header or a Footer to a Form

To insert a header or a footer on a form, follow these steps:

1. From the **Project** tab of **Project Explorer**, expand **Mobile**, **Tablet**, or **Desktop** channel (depending on whether you have created the header or a footer for a mobile, tablet, or desktop channel).
2. Expand **Forms** and click on a required form.
3. From the form properties, click **Form2** tab.
4. For inserting header,
    a. Click the **Edit** button against the **Header** field to open the **Header** dialog box.
    b. From the available list, choose a header and then click **OK**.
5. For inserting footer,
    a. Click the **Edit** button against the **Footer** field to open the **Footer** dialog box.
    b. From the available list, choose a footer and then click **OK**.

Segments

A Segment template enables you to define a template for section headers and rows of the segment. This is primarily useful for achieving common look and feel of section headers along with few widgets added as part of section header of a segment. For Segment, two sample templates are available starting from Kony Visualizer V8 SP3 GA. One sample for the row and another for the header.

> ***Important:*** When you add a segment to a form, you can create a new template without going to the Templates section using the **Create New** option from the drop-down list. You can also edit a template inline in the segment.

Click here to watch a video on using Segment templates in Kony Visualizer.

To create a segment template, from the Template tab, follow these steps:

1. Click the **Templates** tab from **Project Explorer**.
2. Expand **Mobile**, **Tablet**, or **Desktop** (depending on which channel you want to create a map template), point to **Segments** to display a down-arrow. Click this down-arrow and then click **New Template**. Rename the template, if required.
3. Drag and drop a container widget on the template.

   > ***Note:*** A template requires that you add a container widget (for Flex form: **FlexContainer** and for VBox form: **HBox**) before you can add other widgets.

4. Drag and drop the required widgets on the container.
5. Set the properties of these widgets and save the Segment template.

Creating a Segment Template within a Segment Widget

To create a segment template, follow these steps:

1. From the **Project** tab of **Project Explorer**, expand either **Mobile**, **Tablet**, or **Desktop**.
2. Expand **Forms** and navigate to the form that contains a segment widget.
3. Navigate to the Segment tab, depending on your requirement, click the down-arrow of the row template or a section header template and then click **Create New** .

   Screen navigates to the Templates section with a blank template. Rename the template, if required.
4. Drag and drop a container widget on the template.
5. Drag and drop the required widgets on the container.
6. Set the properties of these widgets and save the Segment template.

Edit a Segment Template within a Segment Widget

To edit a segment template, follow these steps:

1. From the **Project** tab of **Project Explorer**, expand either **Mobile**, **Tablet**, or **Desktop**.
2. Expand **Forms** and navigate to the form that contains a segment widget with a template. If such a form with a segment and segment template does not exist, create one. The template can be a row template or a Section Header template.
3. In the **Properties** pane, go to the **Segment** tab.
4. On the template (row or Section header template) you want to edit, click the corresponding **Edit** button.

   Kony Visualizer navigates to the Templates section in the Project properties pane.
5. You can modify the template here. Changes made here will reflect across all the forms which use the template you modified.

Edit a Segment Template inline within a Segment Widget

To edit a segment template, follow these steps:

1. From the **Project** tab of **Project Explorer**, expand either **Mobile**, **Tablet**, or **Desktop**.
2. Expand **Forms** and navigate to the form that contains a segment widget with a template. If such a form with a segment and segment template does not exist, create one. The template can be a row template or a Section Header template.
   If your segment has both a Row Template and a Section Header Template, you can edit them inline.
3. Right-click on the segment and then select **Enable Template Editing.**, and then select Row or a Section Header.
   You can only edit one template at a time. You must disable the template you are editing to start editing the other template.
4. You can click on each individual item in the template and edit it.
   You can also add additional widgets to the template.

Inserting a Segment Template to a Segment Widget

To insert a Segment template to a Segment widget, follow these steps:

1. From the **Project** tab of **Project Explorer**, expand **Mobile**, **Tablet**, or **desktop** channel (depending on whether you have created the segment template for a mobile, tablet, or desktop).
2. Expand **Forms** and navigate to the form containing a segment widget.
3. Expand this form, and click the segment widget.
4. From the Segment properties, click **Segment** tab.
5. The segment templates are available under **General** > **Row Template** and **Section Header Template** list. You may choose to use the template as either Row Template or Section Header

Template for the **Segment** widget.

> ***Note:*** Kony Visualizer provides some sample templates that you can use.

Duplicate a Template

When you are using a template, if you make any changes to that template, it affects all segments where the template is used. To avoid that, you can duplicate a template and modify that specific template.

To duplicate a template, do the following:

1. From the **Project** tab of **Project Explorer**, expand either **Mobile**, **Tablet**, or **Desktop**.
2. Expand **Forms** and navigate to the form that contains a segment widget with a template. If such a form with a segment and segment template does not exist, create one. The template can be a row template or a Section Header template.
3. Right-click on the segment and then select **Duplicate Template.**. The template is duplicated and is visible in your list of templates.

Maps

A map template gives you a standard and consistent way of presenting map-related content in your app, such as a map callout (i.e. a pin) that indicates a particular location on a map.

To create a map template, do the following:

1. On the Project Explorer, click the **Templates** tab.
2. On the **Templates** tab, open the channel you want to add a map template to, either Mobile, Tablet, or Desktop.
3. Within the channel of choice, hover over Maps until its menu arrow appears, click it, and then click **New Template**. Kony Visualizer creates a new, empty map template.

4. Add a FlexContainer to the new map template to hold all the widgets that you want to characterize the map template. To do so, from the Widget pane of the Library Explorer, in the Container Widgets section, click on **FlexContainer** and then drag it onto the map template.
5. Now that the map template has a FlexContainer, drag other widgets onto the FlexContainer. Arrange them and set their properties according to how you want them to display.
6. Save the map template by clicking **Save** on the **File** menu (the **Project** menu in *Kony Visualizer*).

Inserting a Map Template to a Map Widget

To insert a Map template to a Map widget, follow these steps:

1. From the **Project** tab of **Project Explorer**, expand either **Mobile**, **Tablet**, or **Desktop** (depending on whether you have created the map template for a mobile, tablet, or desktop).
2. Expand **Forms** and navigate to the form that contains a map widget, and to which you want to add the map template.
3. Expand this form and click the map widget.
4. From the Map properties, click **Map** tab.
5. Click **Edit** button against **Callout Template** field. This results in opening **calloutTemplate** dialog box.
6. From the available list, choose a map and click **OK**.

Gridcalendars

A Gridcalendar template enables you to define a template for Calendar Day cell. Only one template can be used for each Calendar. The Gridcalendar templates are used:

- to define a Calendar Day cell with custom look and feel.
- to achieve the behavior of having widgets such as an Image and a label for a Calendar Day cell.

- to perform an action on the event of an onclick of a Calendar Day cell.

To create a Gridcalendar template, follow these steps:

1. Click the **Templates** tab from **Project Explorer**.
2. Expand **Mobile** or **Tablet** (depending on which channel you want to create a Gridcalendar template), point to **Gridcalendar** to display a down-arrow. Click this down-arrow, and then click **New Template**. Rename the template, if required.
3. Drag and drop a container widget onto a template.

   > ***Note:*** You need to add a FlexContainer or HBox to a template before adding other widgets.

4. Drag and drop the required widgets on the HBox.
5. Set the properties of these widgets and save the Gridcalendars template.

Grids

A Grid template enables you to define a template for DataGrid widget. You can specify a template for cell and cell headers.

Important Considerations:

- You use a Grid template to update the master data of a DataGrid widget.
- A Grid Template is available only when you fork a form for the Desktop : Web Channel.

To create a Grid template, follow these steps:

1. Click the **Templates** tab from **Project Explorer**.
2. Expand **Desktop**, point to **Grids** to display a down-arrow. Click this down-arrow and then click **New Template**. Rename the template, if required.
3. Drag and drop a container widget on the template.

> ***Note:*** A template requires that you add a container widget before adding other widgets.

4. Drag and drop the required widgets on the container.
5. Set the properties of these widgets and save the template.

Tab Headers

> ***Note:*** Tab Headers templates are supported only on Desktop Web platform.

A Tab header template enables you to define a template for tab headers with specified widgets. You can use the template on individual tab headers of a Tab pane. This is primarily useful for developers to achieve common look and feel of the tab headers of a Tab Pane widget.

A Tab header templates are used in the following cases:

- To have uniform look and feel of the tab headers with the specified widgets.
- To display different tab headers for different Tab panes.
- To perform an action on a tab header.

Creating a Template for Tab Header

When you want information to be displayed or customized in a tab header of a Tab pane in the application, you have a provision to add a tab headers template.

To create a Tab Header template at the application-level, follow these steps:

1. Click the **Templates** tab from the **Project Explorer**.
2. Expand **Desktop**, point to **Tab Header** to display a down-arrow. Click this down-arrow, and then click **New Template**. Rename the template, if required.
3. Drag and drop a container widget on the template.

> ***Note:*** A template requires that you add a container widget (**HBox**) before you can add other widgets.

4. Drag and drop the required widgets.
5. Set the properties of these widgets and save the **Tab Header** template.

Context Menus

> ***Note:*** Context Menu templates are supported only on Desktop Web platform.

What is a Template for a Context Menu

In Desktop Web, when you right-click a Menu Container or a Box the context specific menu will be displayed with the array of menu items. A Context Menu Template is essentially a template having a Menu Container in which the developer customizes the overall look and feel of the context menu.

Where to use a Context Menu Template

Context Menu Templates are used to display how the data is presented to the end user when a context menu pops up.

Creating a Template for Context Menu

When you want the same template to be displayed across all the Context Menus in an application, you have a provision to add a Context Menu Template.

To create a context menu template at the application-level, follow these steps:

1. Click the **Templates** tab from the **Project Explorer**.
2. Expand **Desktop**, point to **Context Menus**. A down-arrow appears. Click this down-arrow, and then click **New Template**. Rename the template, if required.
3. Drag and drop an MenuContainer on the template.
4. Follow the steps described in creating Menu templates.

Menus

> ***Note:*** Menu templates are supported only on Desktop Web platform.

Menu templates contains a list of menu items that are displayed when you right-click a widget.

To create a menu, follow these steps:

1. Click the **Templates** tab from the **Project Explorer**.
2. Expand **Desktop**, point to **Menus** to display a down-arrow. Click this down-arrow and then click **New Template**. Rename the template, if required.
3. Drag and drop an HBox onto a template.

   > ***Note:*** You need to add an HBox to a template before adding other widgets.

4. Drag and drop the required widgets on the HBox.
5. Set the properties of these widgets and save the Menu template.

Design a Responsive Web App

Overview

Responsive Web Design is an approach to web-page creation that makes use of flexible layouts, flexible images, and cascading style sheet (CSS) media queries. The goal of Responsive Web Design is to build web pages that can detect a user's screen size and orientation, and then change the layout accordingly. Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge a website, to make it look good on all devices (desktops, tablets, and phones).

From Kony Visualizer V8 onwards, the Responsive Web Design feature has been implemented in Kony Visualizer. When you resize a web browser screen, content in the browser resizes based on the resizing design. Using Responsive design, you can design a web page for different form factors in a single FlexForm.

Responsive Web Design supports the following features:

- Browser size changes
- Browser zoom-in and zoom-out
- Screen resolution changes

> ***Note:*** A **Progressive Web App** (PWA) acts as a progression of a Responsive Web app. For example, if you have a website (web page/web app) that is mobile responsive, you can leverage the new features supported by modern web browsers to make it a Progressive Web App. For more information about Progressive Web Apps, click here.

From Kony Visualizer V8 SP2, the existing Responsive Web Design feature has been enhanced. Prior to Kony Visualizer V8 SP2, Responsive Web Design was achieved through code. From Kony Visualizer V8 SP2 onwards, you can build Responsive Web desktop applications from within Kony Visualizer.

Click here to watch a video on the Responsive Web Design feature.

For a more hands-on approach on the Responsive Web Design feature provided by Kony AppPlatform, import and preview the **Resort Feature** sample app by using Kony Visualizer.

The Responsive Web design feature is currently available only for the Desktop Web platform.

> ***Important:*** You must enable the Responsive Web feature in the Desktop Web Properties section here.

When your app is open in a browser, if the browser window resizes, content which is in a bigger layout does not resize according to the smaller layout automatically. Responsive web enables you to design your desktop web applications to fit various layouts and create a glitch free browsing experience.

There are three pre-existing breakpoints: Mobile, Tablet, and Desktop These breakpoints help you to resize your web application to pre-set scales. You can either use them or you can create your own custom breakpoint and use it. The ruler in the form assists you in viewing these breakpoints. When you hover on the breakpoint section, the breakpoint area is highlighted in the ruler indicating the screen size.

You can switch between various breakpoints set on the canvas. When you click a break point, the canvas resizes to the width specified on that breakpoint and all the content in the canvas adjusts according to the specified canvas width. You cannot hide the ruler when breakpoints are set for a form.

You can resize the breakpoints using the resize handle icon on the right side of the canvas. This provides a quick review so that you can determine the exact width where the content design breaks during the resize. You can also use a key (Command + D key in Mac and Ctrl + D key in Windows) when you reach the required width to insert a breakpoint of a specified width when you are resizing a breakpoint. You can drag a breakpoint in a ruler.

For New projects, you can use the three default values or create a new one by using the Add New button.

- Desktop - 1200 PX
- Tablet - 1024 PX
- Mobile - 640 PX
- Add New - Button

For existing projects, you do not have any default breakpoints. You can add a breakpoint by using the Add New button.

Breakpoints are always sorted from largest to smallest. If you add a new breakpoint which is larger than the existing large breakpoint, the newly added largest one goes to the top.

> ***Important:*** A new event **onBreakpointChange** is added to the Actions list in the Properties pane. You can define what happens to the form when the breakpoint changes.

Enable Breakpoint Forking

You can choose the layout for each of the breakpoints separately by using the Breakpoint Forking feature. When this feature is turned on, you can rearrange your layout in the form, specific to each breakpoint. This helps you in defining how you want to display the form in each layout. You can choose the changes at the breakpoint level or a global level for all breakpoints.

> *Important:* When you turn Breakpoint Forking on, you can fork your widgets separately for each breakpoint. If you do not fork your widgets for each breakpoint, the canvas will resize only a few widgets and not all of them.

Limitation

A limitation related to Breakpoint Forking and setting the height of a widget as Preferred/any non-Preferred value is present in a Responsive Web form of Kony Visualizer.

Consider a scenario where you have a Responsive Web form with Breakpoint Forking turned off. You add any widget to the form and under **Properties** > **Look** > **Flex**, set the **Height** of the widget as **Preferred**. When you then turn on Breakpoint Forking and select a breakpoint, it is not possible to change the Height of the widget to any non-Preferred value (such as 10 Dp).

The same limitation exists when you first set the Height of a widget as any non-Preferred value (such as 10 Dp), then turn on Breakpoint Forking, and try to modify the Height of the widget to Preferred.

Enable Grid Lines

You can choose to enable grid lines on your Responsive Web canvas. These grid lines help you to properly align widgets and other elements of your Responsive Web app. You can use grid lines as reference points while placing various app elements according to the breakpoints that you have specified.

After you enable or disable grid lines for one Responsive Web form, the corresponding setting is applied for all Responsive Web forms in Kony Visualizer. Grid lines for Responsive Web forms are disabled by default. Both these behaviors are applicable from Kony Visualizer V8 SP4 Fixpack 28 onwards.

To enable grid lines for Responsive Web, follow these steps:

1. Select a Responsive Web form.
2. At the upper-right corner of the Project Canvas, click the down-arrow icon  . A list of options appears.

3. Click **Show Grid**.

> ***Note:*** If grid lines are already enabled and you want to disable this feature, click **Hide Grid** from the down-arrow icon list.

4. Kony Visualizer enables grid lines for your Responsive Web form.

Working with Skins, Widgets, and Components

You can also fork skins for different breakpoints. For Video and Image widgets, the source of images and videos can be forked based on the breakpoint. For Labels, whenever you change a breakpoint in the canvas, the width of the label changes and the content wrapping is adjusted for the breakpoint.

When you drag and drop a widget to the form and make changes, unless you have selected the Breakpoint Forking feature, all changes reflect across breakpoints. If a breakpoint is forked, changes will not reflect in that breakpoint.

When you drag and drop a widget on any breakpoint, the widget is added to all breakpoints. When you delete or rename a widget on a breakpoint, the widget is deleted or renamed on all breakpoints. You can change properties of a widget specific to each breakpoint.

You can use components in across breakpoints.

Responsive Web Components

To create a responsive web component, you will have to create a responsive web UI on a form, and then convert the set of widgets into a component.

When you add responsive web components to a collection library, the responsive properties (such as custom breakpoints) of these components do not change even when you reuse the component in a different project or form through the collection library. Similarly, widgets with responsive data also retain their responsive configuration when reused. You can use Desktop web templates for the Responsive Web application. You can also create different templates for different breakpoints. Once you publish your responsive web app, you can preview the app in the kdw (desktop web output) mode.

While using Components with Responsive web breakpoints on forms, the following rules are applicable:

1. When the breakpoints on the form are different from the breakpoints on the components, the responsive web app will have all the breakpoints.
   For example, if the form has 1380, 680, 400; if the component has 1150, 800, and 200 when the app is published, all six breakpoints will be supported.

2. In a scenario where the component must be displayed in a break-point for which its layout has not been configured, the component will display the default layout of the component.
   For example, if the component has the layout configured for 800 and 400 and the form has the layout configured for 900 and 700. If the form is displayed on a screen with 900 px, the component will be displayed in its default layout.

Responsive Web APIs

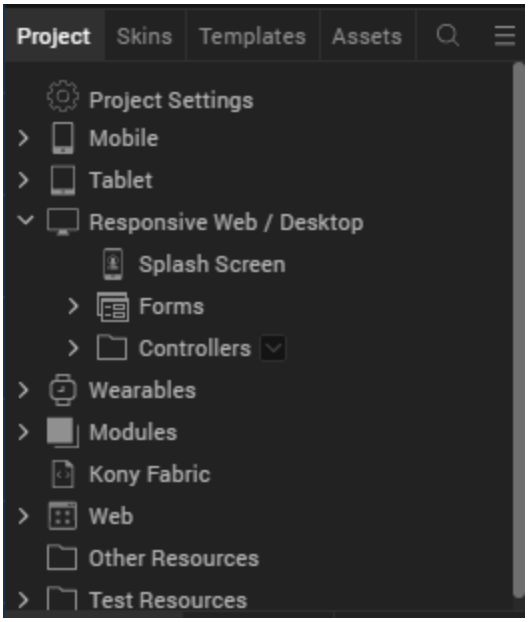
The following are the APIs, keys, and constants associated with Responsive Web.

- kony.application.setApplicationBehaviors (Breakpoint key)
- kony.application.getCurrentBreakpoint();
- constants.BREAKPOINT_MAX_VALUE

Create a Breakpoint

To create a Responsive Design application, do the following:

1. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **New Project** to open the **Start a New Project** screen of the **New Project** wizard.
2. Select **Create Custom App**, and then click **Choose**. Kony Visualizer opens the **Project Type** screen of the **New Project** wizard.
3. In the **Project Name** text box, enter a name for your project.
    - A project name should contain fewer than 14 characters.
    - A project name can be alphanumeric. However, the first character of a project should always be a letter.
    - Do not use any of the following reserved keywords as a project name: authService, workspace, mfconsole, kpns, middleware, accounts, syncservice, syncconsole, services, admin, middleware, and appdownload.
4. Select **Kony Reference Architecture**, and then click **Create**. As this is a new project, in the Project Explorer, you will see Responsive Web / Desktop.



5. Right-click **Form** > **New Form**. This results in creating a new form with a default name assigned to it.
6. You will have three different breakpoints by default.
   You can view these breakpoints in the Properties pane.
7. To add a new breakpoint, click **Add New**. A new row appears.
8. Enter the breakpoint details. You will see that the new breakpoint is added.

   The breakpoint is added and it reflects in the Properties pane and the Canvas.

Create a Form with Breakpoint Forking Off

To create a form with Responsive Web with Breakpoint Forking Off, do the following:

1. In your Visualizer Project, on the form that you want to use Responsive Web design, in Visualizer Canvas, select **Responsive Web**.
   The Responsive web layout appears on Visualizer Canvas.
   Tip: You may want to set the view of Canvas to 75%. Setting the canvas to 75% view makes it easier to navigate through three different breakpoints.
2. Drag and drop a **FlexContainer** to the form.
3. Ensure that this FlexContainer is spread across all three default breakpoints.
   Tip: For better visibility on how the canvas changes the FlexContainer in different breakpoints, you may want to change the background color of the FlexContainer.
4. Add a few TextBox widgets to the FlexContainer.
   Tip: Add three TextBox widgets spread across the entire canvas.
5. Add a few buttons below the FlexContainer in the form.
   Tip: Ensure that three buttons are spread across the entire canvas.
6. Using the resize handle, resize the canvas to different breakpoints, observe how the breakpoints change.

Create a Form with Breakpoint Forking On

To create a form with Responsive Web with Breakpoint Forking On, do the following:

1. In your Visualizer Project, on the form that you want to use Responsive Web design, in Visualizer Canvas, select **Responsive Web**.
   The Responsive web layout appears on Visualizer Canvas.
   Tip: You may want to set the view of Canvas to 75%. Setting the canvas to 75% view makes it easier to navigate through three different breakpoints.
2. Drag and drop a **FlexContainer** to the form.
3. Ensure that this FlexContainer is spread across all three default breakpoints.
   Tip: For better visibility on how the canvas changes the FlexContainer in different breakpoints, you may want to change the background color of the FlexContainer.
4. Add a few TextBox widgets to the FlexContainer.
   Tip: Add three TextBox widgets spread across the entire canvas.
5. Add a few buttons below the FlexContainer in the form.
   Tip: Ensure that three buttons are spread across the entire canvas.
6. On Visualizer Canvas, toggle the Breakpoint Forking button.
7. At this point, your UI elements are already organized for the largest breakpoint.
8. Using the canvas resize handle, resize the canvas to the tablet layout.
9. Reorganize your UI elements for this layout.
10. Using the canvas resize handle, resize the canvas to the mobile layout.
11. Reorganize your UI elements for this layout.
12. Your form now has three different views for three different breakpoints.
13. Using the resize handle, resize the canvas to different breakpoints, observe how the breakpoints change.

14. To view your app locally on an internet browser, from the **Project Menu**, click **Run** and select **Run**.
15. The project is built for the Responsive web channel and you can view the app in your internet browser.

To invoke any function that is not supported by Visualizer, you can do it through code. Click here for more information.

Design a Progressive Web App

Overview

A Progressive Web App (PWA) is the next step of a Responsive Web app. For example, if you have a website (web page/web app) that is mobile responsive, you can leverage the new features supported by modern web browsers to make it a Progressive Web App. These additional features include using service workers, web app manifests, push notifications, and offline support.

A Progressive Web App is an application that feels like a Native app, but is available over web browsers to a user. A user can access a Progressive Web App on any web browser on a mobile device. The app is responsive, functions even when the device is offline or has limited network speed, and does not require any updates.

For information about how to build a Progressive Web app, click here.

> ***Important:*** Before you start to build a Progressive Web App, you must enable the PWA option for a Responsive Web application. When you do so and build the application, a new output is created in the Progressive Web App format. This new output does not replace any Responsive Web output, but instead a new output is created in the same folder. For more information about Responsive Web apps, click here.

This topic consists of the following sections:

- Guidelines to Optimize Progressive Web Applications
- Progressive Web Apps vs. Native Apps vs. Responsive Websites
- Progressive Web Apps - Videos
- Progressive Web Apps-Caching
- Progressive Web Apps - FAQs

A Progressive Web App is always up to date as it uses a Service Worker. A Service Worker is a script which ensures that the features that are not required to appear in the web browser or that don't need any user interaction are linked with the web app. This service worker script runs in the background and helps in synchronizing the data in the background and in displaying push notifications.

When the Progressive Web App feature is enabled, on Kony Fabric, you must configure server assets caching as zero. Furthermore, you must not select the **HttpSessioncaching** option to fetch the latest content from the server. For more information, click here.

A Progressive Web App has the following features:

- **Progressive**: Site URLs work across all mobile web browsers.
- **Responsive**: All web pages are responsive on mobile and tablet devices.
- **Secure**: The web app is served over HTTPS.
- **Offline**: The app URLs load when they are offline.
- **Easy Access**: You can add a Progressive Web app to your home screen on your device.
- **Fast**: When you load the app for the first time on 3G, the app is fast.
- **Unique Links**: Each page in the Progressive Web application has a unique URL.

The Progressive Web App generated by Kony Visualizer meets 100% of Google's PWA baseline checklist.

For a more hands-on approach on the Progressive Web Apps feature provided by Kony AppPlatform, import and preview the Events, Employee Directory, and Resort Feature sample apps by using Kony Visualizer.

- **Events app**: 

- **Employee Directory app**: 

- **Resort Feature app**: 

Progressive Web Apps vs. Native Apps vs. Responsive Websites

The following table illustrates the differences in the availability of various features between PWAs, Native apps, and Responsive websites.

Functionality	Native App	Responsive Website	Progressive Web App
Functions Offline	✓	✗	✓
Push Notifications	✓	✗	✓
Installable on Home screen	✓	✗	✓
Full-screen experience	✓	✗	✓
Indexable by search engines	✗	✓	✓

Functionality	Native App	Responsive Website	Progressive Web App
One place to enter content	✗	✓	✓
Works across all devices	✗	✓	✓
No download required	✗	✓	✓
Does not require updates	✗	✓	✓

Guidelines to Optimize Progressive Web Applications

Progressive Web Apps can be enabled on the Responsive Web channel on Visualizer. When you build Progressive Web Apps, you can use Google Lighthouse to improve the quality of your Progressive Web Apps. The higher the score on Google Lighthouse, the higher the quality of your Progressive Web App. You can achieve a better score on Google Lighthouse by taking care of a few things.

> ***Important:*** To get an ideal evaluation with Google Lighthouse, build your app in the Release Mode and publish the app on your runtime environment.

Optimize Application Load or Startup

By optimizing your application startup or the load page, you can achieve a better score on Google Lighthouse.

- Ensure that the first screen of the app is lightweight and of the lowest size possible.
- Add a Splash screen for the application.

- Avoid placing any unwanted images on the first screen.
- Avoid placing service calls in the first screen of the App that are not required for its functionality

Optimize Web App Manifest

When you optimize your web app manifest file for the Progressive Web app, it can increase your quality score on Google Lighthouse. The optimizations define how your app anchor appears when a user adds your app to their home screen on their device.

- Ensure that you add images for your App to anchor the App on the home screen.
    - For mobile devices, add to Home screen is displayed by the device browser if the manifest contains the App icons.
    - For desktop browsers, use the **beforeinstallprompt** event for Google Chrome to display a notification to the users. You can find more information on **beforeinstallprompt** and similar functionality on other browsers at https://developers.google.com/web/fundamentals/app-install-banners/
- Add a background color in the Manifest file.
- Add a theme color in the Manifest file.

Service Workers

When Progressive Web is enabled, the application is not expected to specifically make any changes for the Service workers to be registered and enabled. The web framework within Visualizer inherently helps your application create and register a service worker. The service worker is created using the default caching mechanism.

Progressive Web Apps - Videos

This section contains a series of videos that explain how to create Progressive Web Apps using Kony Visualizer.

Video One

This is the first of the Progressive Web App tutorial series. Here, we walk you through the steps to build a basic web application by making use of Visualizer's low-code capabilities.

Click here to watch the PWA Series - Video 1 of 3: Building a Basic Web App.

Video Two

In this second tutorial of the PWA series, we walk you through the steps to enhance the web app created in our first video. We will add the search capability, better responsive layout, and improved aesthetics.

Click here to watch the PWA Series - Video 2 of 3: Enhancing the Web App.

Video Three

In this third video tutorial of the PWA series, we will take the app that we built in our previous videos and add PWA capabilities.

Click here to watch the PWA Series - Video 3 of 3: Making the App a Progressive Web App.

Progressive Web Apps-Caching

Caching or offline storage is a method that an application uses to store assets or data in the internal memory of a device temporarily. In Progressive Web applications, this mechanism helps the user to access assets or data from a URL, even when the internet connection is slow.

From Visualizer V9 onwards, you can include custom caching categories in a Progressive Web application.

There are two ways to implement caching in a progressive web app using Kony Visualizer.

- cachingMechanism function in Service Worker Helper file
- Custom Service Worker file

> ***Note:*** In Kony Visualizer, you cannot use both the methods of caching together. You can either use the cachingMechanism function or provide your own service worker file.

cachingMechanism Function in Service Worker Helper File

A Service Worker Helper file extends the functionality of the default Service Worker of a Progressive Web application.

In this section, learn how to create a Service Worker Helper file and add it to an application.

cacheMechanism Function

A Service Worker Helper file defines the *cacheMechanism* function . Each time a URL is accessed in the application, the *cacheMechanism* function is invoked.

The *cacheMechanism* function contains the following details:

Syntax

cacheMechanism(url)

Input Parameters

url [String]- Mandatory

This parameter specifies the URL of the service used in the application.

Return Parameters

JSObject [object]

This object must contain only *cachestrategy* key.

cachestrategy key can have any of the following values.

Constant	Description
constants.NONE	This is the default value of *cachestrategy* key. When the *cachestrategy* key is set to this constant, the data is not stored in the cache. The browser fetches the data each time the URL, specified in the url parameter, is used in the application.
constants.NETWORK_ONLY	When the *cachestrategy* key is set to this constant, the data is not stored in the cache. The browser fetches the data each time the URL, specified in url parameter, is used in the application.
constants.NETWORKFIRST_ CACHELATER	When the *cachestrategy* key is set to this constant, each time the URL, specified in url parameter, is used in the application, the browser will check if there is a network connection. If a network connection has been established, the browser fetches the data from the server. If there is no network connection, the browser updates the application with the data in the cache.
constants.CACHEFIRST_ NETWORKLATER	When the *cachestrategy* key is set to this constant, each time the URL, specified in url parameter, is used in the application, the browser will first check the cache for data. If the cache does not have the data corresponding to the URL, the browser will fetch the data from the server.

Note: The HTTP request method of the URL specified in *url* parameter must be GET. If the URL uses the HTTP request method as POST, the *cacheMechanism* function is not invoked. If you want to implement caching or offline storage for a URL with POST method, you must use, kony.nosql APIs.

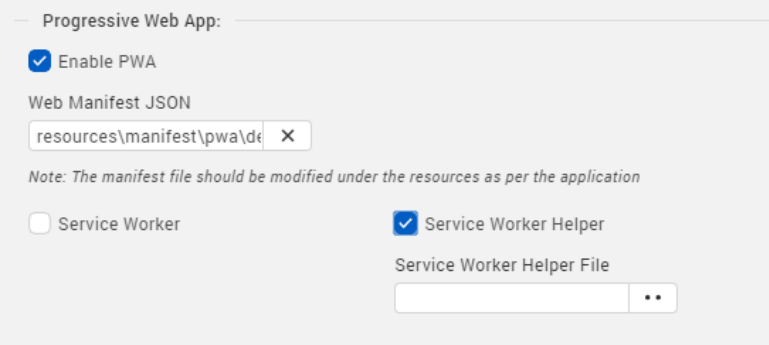
Here is an example of the cacheMechanism function in a Service Worker Helper file.

```
function cacheMechanism(url) {

var returnObj = {
 'cachestrategy': constants.NONE
};

if (url.endsWith('.png') || url.endsWith('.css)) {
  returnObj = {
   'cachestrategy': constants.CACHEFIRST_NETWORKLATER
  };
 }
 else if (url.indexOf('services') > 0) {
  returnObj = {
   'cachestrategy': constants.NETWORK_ONLY
  };
 } else if (any regex can be used) {
  returnObj = {
   'cachestrategy': constants.NETWORKFIRST_CACHELATER
  };
 }
 return returnObj;
}
```

Adding a Service Worker File to an Application

1. Create a Service Worker Helper file. For information on how to create the file click here.
2. Open the application to which you want to add the Service Worker Helper file in Kony Visualizer.
3. From the left navigation bar, click **Project Settings**.
4. In the **Project Settings** window, from the left menu, select **Responsive Web**.
5. Under the **Progressive Web App** section, select the option **Service Worker Helper**. Ensure **Enable PWA** is selected in the **Progressive Web App** section.

6. Under the text **Service Worker Helper file**, provide the location of the Service Worker Helper JS file.



7. Click **Done**.

8. Build and publish the application in release mode to view the changes. For more information on how to build and publish your application in release mode, click here.

Custom Service Worker

Follow these steps to include the Service Worker in your application.

1. Create a new JS file and name it as per your requirement (say serviceWorker.js). Add the following line as the first line of the script.

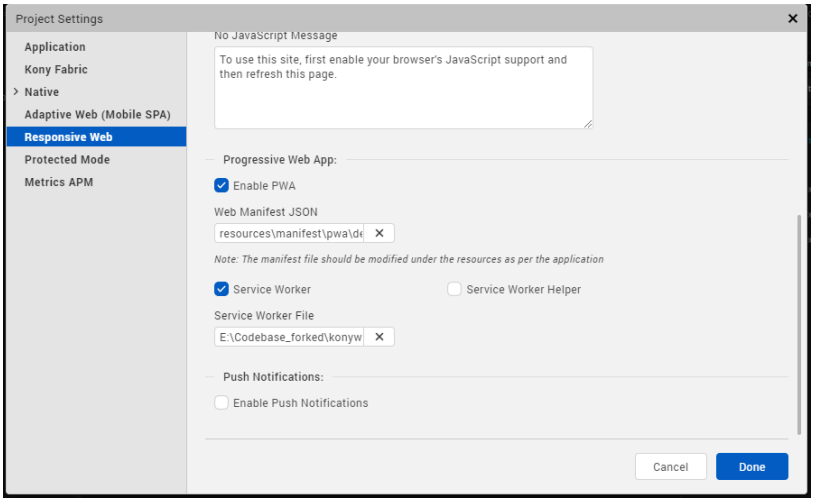
```
importScripts('./nocache/sw-ext.js')
```

This script imports the **sw-ext.js** file into the service worker. sw-ext.js file has the configurations for push notification registration, some default values for cache, etc.

> ***Note:*** When you incorporate your own Service Worker in your application, the default Service Worker file created by the platform is replaced with your file. You must provide the scripts for all the lifecycle events of the service worker. For more information about creating your Service Worker JS file, click here.

2. From the left navigation bar, click **Project Settings**.

3. In the **Project Settings** window, from the left menu, select **Responsive Web**.
4. Under the **Progressive Web App** section, select the option **Service Worker Helper**. Ensure **Enable PWA** is selected in the **Progressive Web App** section.
5. Under the text **Service Worker file**, provide the location of the Service Worker file.



6. Click **Done**.
7. Build and publish the application in release mode to view the changes. For more information on how to build and publish your application in release mode, click here.

Progressive Web Apps - FAQs

- **Is it possible to create Segment templates responsively? Can we use the Breakpoint Forking feature while creating Segment templates?**
  No, it is not possible to do so. Breakpoints are a concept of forms. Form layouts change based on the breakpoint that the page (form) width hits. Breakpoints are only provided at form-level. For Segments, you can use separate templates to achieve a unique layout for every breakpoint.
- **Can we create components with Breakpoint Forking?**
  Breakpoints are only provided at form-level. Components are designed to be cross-platform, and are not specific for a breakpoint. Because breakpoint is a form concept, you can fork a

component for a specific breakpoint within a form, but not outside the form. And prior to run time, you can validate the designs on the form canvas.

- **Is it possible to create a Map Pin callout template responsively?**
  Similar to Segment templates, you can use separate templates if you want a unique layout for each breakpoint.

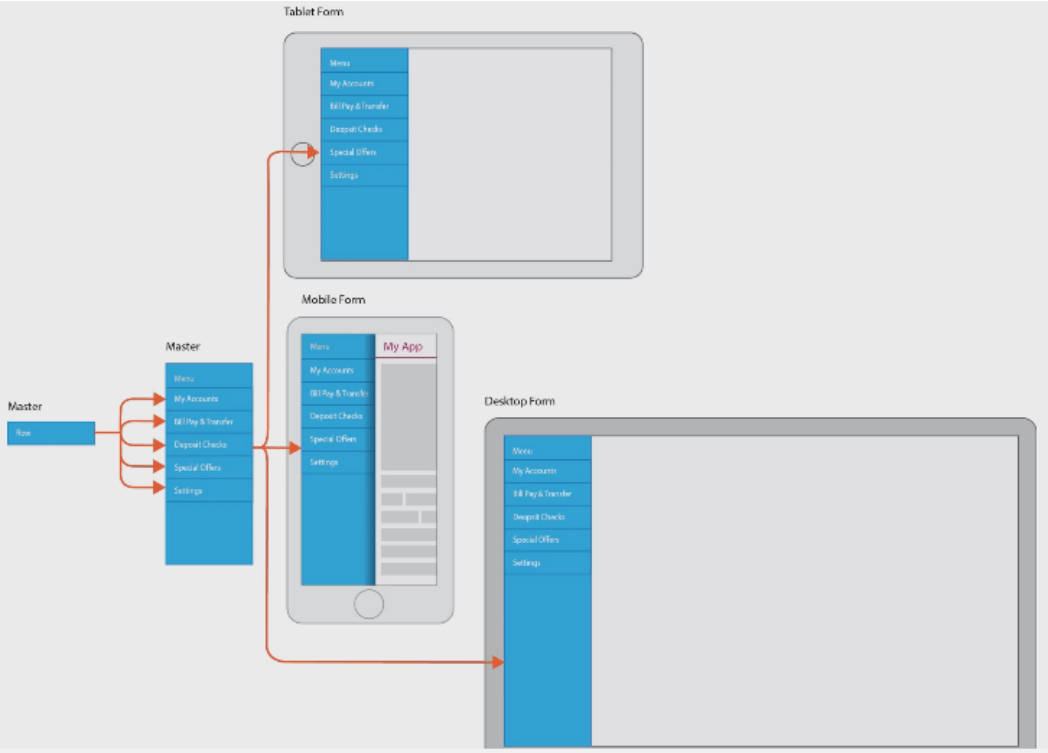
- **If a Progressive Web App is installed on an Android device by using the Google Chrome app install banner, will Kony be able to send push notifications for that app?**
  Yes, Kony can send push notifications in such a scenario. However, you must ensure that the Google Chrome version is 73 or later.

- **A Progressive Web App runs on the Android stock browser. Will Kony be able to send push notifications for that app?**
  Kony can send push notifications for that PWA, if the Android stock browser is compatible with PWA (works properly on Google Chrome).

- **If a Progressive Web App is installed on an Android device through the Google Play store, will Kony be able to send push notifications?**
  If the PWA is installed through the Google Play store, that app opens as a Google Chrome window. So, Kony can send push notifications in that case.

Use Masters

Masters provide the capability to define user interface elements and action sequences in a single definition, and then instantiate them anywhere throughout an application. A master may be nested within other masters, or within forms. Additionally, masters can be built once and then instanced for mobile, tablet, and desktop, as illustrated in the following diagram.

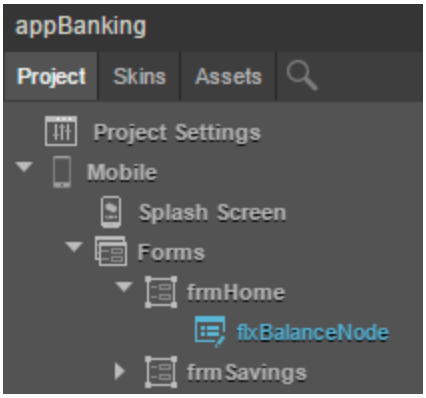


For example, let's say you have a master container for a menu, and that menu is made up of numerous menu items using Segment widgets. By making a Segment widget of a given menu item a master within that master container, changes to that individual master menu item result in just that item being changed in every instance where that master container is used. Editing the properties of a master changes the properties of every instance of the given master. And if you find that, within an instance of a master, you need to make a change to data in a widget, just that widget is changed; the master itself and any other instances of that widget that are based on the master remain unchanged.

Masters may be nested within the following widgets, whether inside of another master or within a form.

- Flex Container
- Flex Scroll Container
- Segment
- Tab Pane

When you insert an instance of a master into a form—or even into another master—the instance appears in the Project Explorer as blue text, making it easy to identify as a master instance. The instance has the same name as the top widget or container in the master.



Masters also support the inclusion of common code for lifecycle events and other actions at the container level. For more information, see Create a Master from Scratch. Also, beginning in Kony Visualizer 7.2, you reference master elements in code using the following hierarchy:

```
FormName.MasterName.WidgetName
```

The use of this hierarchy makes it possible to duplicate widget names between masters and their instances, since inclusion of the entire hierarchy ensures a unique element reference.

For information on using masters in Kony Visualizer, click any of the following topics:

Convert an Existing Container to a Master

Create a Master from Scratch

Insert an Instance of a Master

Control the Cascading of Changes from the Master to Its Instances

Reset the Values in a Master Instance

Use the Segment2 Widget in a Master

Create Widgets from a Master

Edit a Master

Convert an Existing Container to a Master

The easiest way to create a master is to base it off of a container you have already created. You can do so from either an existing form or an existing master.

To convert an existing container to a master, do the following:

1. On the Visualizer Canvas, click the top level container that you want to convert to a master. This can reside either in a form or a master.
2. Right-click it, and then click **Create Master**. Doing so creates a master, and links it within its original parent container.
3. To access and modify the master you have just created, click the **Templates** tab on the Project Explorer, expand the **Masters** category, and then click the master, opening it on the Visualizer Canvas.

> ***Important:*** When you are converting a set of widgets to a master, ensure that you specify the coordinates and dimensions of the widgets in % units.

Create a Master from Scratch

If you want, you can create a master from scratch rather than basing it on existing assets.

To create a Master from scratch, do the following:

1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. Hover over Masters, click its context menu arrow, and then click **New Master**.

   A new master is created, along with a FlexContainer for the widgets that you will add.
3. By design, the FlexContainer is not scrollable. To change the FlexContainer into a FlexScrollContainer, making the container scrollable, on the **Templates** tab, hover over the FlexContainer that you want to convert, click its context menu arrow, point to **Convert To**, and then click **Flex Scroll Container**.
4. Add widgets to the container of the master just as you would on a standard form.
5. Add code to lifecycle events and other actions. To do so, select the flex container, then on the **Properties** pane, on the **Action** tab, click the **Edit** button of the event to which you want to add code. For more information, see Add Actions and Set App Lifecycle Events.

Insert an Instance of a Master

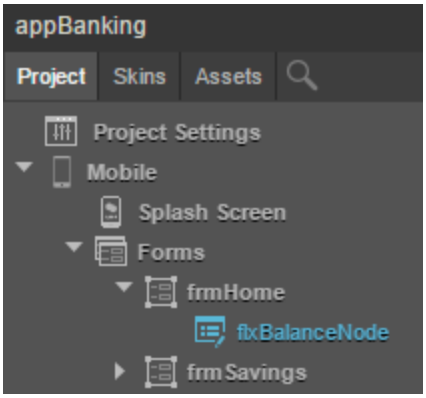
You can insert an instance of a master to virtually any form scenario, including inserting masters within another master.

If it makes it easier for you, you can simplify the process of inserting a master by moving the **Templates** tab from the Project Explorer to the Library Explorer. By doing so, the **Templates** tab (which displays masters) and the Project tab (which displays forms) become viewable at the same time since they're no longer occupying the same pane. To move the **Templates** tab from the Project Explorer to the Library Explorer, on the Project Explorer, right-click the **Templates** tab, and then click **Dock to Library**.

To insert an instance of a master, do the following:

1. On the Project tab of the Project Explorer, locate and open the form that you want to apply the master to. The form displays on the Visualizer Canvas and is now the form with focus.
2. On the **Templates** tab, locate the master that you want to apply to the form.
3. Do either of the following:
   - Click the context menu arrow of the master you want to create an instance of, and then click **Insert Into**. An instance of the master is placed on the form you originally selected.
   - Drag the master from the **Templates** tab over to the Visualizer Canvas and drop it onto the form where you want to insert an instance of a master.

   The instance appears in the Project Explorer as blue text, making it easy to identify as a master instance. The instance has the same name as the top widget or container in the master.



Control the Cascading of Changes from the Master to Its Instances

You can control whether changes to a master are inherited by its instances. The types of changes are divided into two categories: the data contained in the master (called Reference Data), and the properties of the master (called Reference Flex Properties). In some cases, you may want to ensure that an instance does not vary from a master, such as in the case of the positioning and dimensions of a menu bar. In other cases, you may want an instance to vary from the master, such as in the particular data that the instances contain.

By default, an instance of a master does not automatically inherit changes made to the master. To enable such inheritance, you need to select the instance and set it to inherit the reference data and/or reference flex properties of the master.

To control the cascading of changes from the master to its instances, do the following:

1. On either the Project tab of the Project Explorer, or on the Visualizer Canvas, locate and select the parent container of the instance for which you want to change data and properties inheritance.
2. From the Properties pane, Click the FlexContainer tab.
3. Depending on what your need is, do any of the following:

   > ***Important:*** If the data or flex properties of your instance vary from the master, and you set the instance to inherit from the master, all variations will be overwritten, and you cannot undo this operation.

   - To have the data of the instance of the master reference and inherit the data of the master, set Reference Data to **On**.
   - To have the flex properties (such as position and dimensions) of the instance of the master reference and inherit the flex properties of the master, set Reference Flex Properties to **On**.
   - To allow the data of the instance of the master to vary from the data of the master, set Reference Data to **Off**.
   - To allow the flex properties of the instance of the master to vary from the flex properties of the master, set Reference Flex Properties to **Off**.

   The types of widgets in which you can change data are listed in the following table. The changes you make are specialized to that widget without affecting the master. Any future changes made to the master involving the widget are not reflected in the widget you just modified.

Widget	Data
Button	• Button Text • All properties in the Flex section of the **Look** tab on the Properties pane • Action Sequences • **Review** tab
Calendar	• All properties in the Flex section of the **Look** tab on the Properties pane • Default date on the **Calendar** tab • Action Sequences • **Review** tab
Check Box Group	• All properties in the Flex section of the **Look** tab on the Properties pane • Master Data property, on the **Check Box Group** tab • Action Sequences • **Review** tab
Data Grid	• All properties in the Flex section of the **Look** tab on the Properties pane • Data, located on the **Data Grid** tab • Action Sequences • **Review** tab

Widget	Data
Image	• All properties in the Flex section of the **Look** tab on the Properties pane • All properties in the General section of the **Image** tab on the Properties pane • Action Sequences • **Review** tab
Label	• All properties in the Flex section, and the Text property, on the **Look** tab on the Properties pane • All properties in the General section of the **Image** tab on the Properties pane • Action Sequences • **Review** tab
List Box	• All properties in the Flex section of the **Look** tab on the Properties pane • Master Data property, on the **List Box** tab • Action Sequences • **Review** tab
Radio Button Group	• All properties in the Flex section of the **Look** tab on the Properties pane • property, on the **Radio Button Group** tab • Action Sequences • **Review** tab

Widget	Data
Rich Text	• All properties in the Flex section, and the Text property, on the **Look** tab on the Properties pane • Action Sequences • **Review** tab
Slider	• All properties in the Flex section of the **Look** tab on the Properties pane • Min Label and Max Label values on the **Slider** tab • Action Sequences • **Review** tab
TextArea2	• All properties in the Flex section, and the Text property, on the **Look** tab on the Properties pane • The following properties on the **Text Area** tab: Max Characters, Input Mode, Placeholder, Auto Capitalize • Action Sequences • **Review** tab

Widget	Data
TextBox2	• All properties in the Flex section, and the Text property, on the **Look** tab on the Properties pane • The following properties on the **Text Area** tab: Max Characters, Input Mode, Placeholder, Auto Capitalize • Action Sequences • **Review** tab
Browser	• All properties in the Flex section of the **Look** tab on the Properties pane • Master Data property, on the **Browser** tab • Action Sequences • **Review** tab
CordovaBrowser	• All properties in the Flex section of the **Look** tab on the Properties pane • Master Data property, on the **CordovaBrowser** tab • Action Sequences • **Review** tab
Camera	• All properties in the Flex section, and the Text property, on the **Look** tab on the Properties pane • Action Sequences • **Review** tab

Widget	Data
Map	• All properties in the Flex section of the **Look** tab on the Properties pane • Action Sequences • **Review** tab
Phone	• All properties in the Flex section, and the Text property, on the **Look** tab on the Properties pane • Action Sequences • **Review** tab
Picker View	• All properties in the Flex section of the **Look** tab on the Properties pane • Master Data property, on the **Picker View** tab • Action Sequences • **Review** tab
Segment	• All properties in the Flex section of the **Look** tab on the Properties pane • Master Data property, on the **Segment** tab • Action Sequences • **Review** tab

Widget	Data
Switch	• All properties in the Flex section of the **Look** tab on the Properties pane • The following properties on the **Text Area** tab: Left Text, Right Text • Action Sequences • **Review** tab
Video	• All properties in the Flex section of the **Look** tab on the Properties pane • Action Sequences • **Review** tab

Reset the Values in a Master Instance

If you have been changing the values in an instance of a master to differ from those of the master itself, you can reset these values, clearing them and causing the instance of the master to inherit all the values of the master. Resetting these values does not prevent you from continuing to customize the values of the instance, it just resets them to the original master values.

To reset the values in a master instance, do the following

1. On the **Project** tab of the Project Explorer, locate and select the parent container of the instance for which you want to reset its values. The instance becomes selected and is the item of focus on the Visualizer Canvas.
2. Right-click the instance on the Visualizer Canvas, and then click **Reset Instance Values**. all the values in the instance are cleared, and inherit the values of the master.

Use the Segment2 Widget in a Master

One you add a Segment2 widget to a master, you cannot drop other widgets onto that segment widget because it's inside a master. To get around this limitation, you create a segment row template, populate it with the widgets you want it to have, and then you configure the segment widget in the master to reference that row template. Once the segment widget in your master is populated width content from the row template, you can edit the segment widget's master data to have the per-row customized data that you need.

> ***Note:*** A segment template can only apply to one channel. To work around this, create the segment template for one channel, copy it, and then paste it into the other channels.

To use a Segment2 widget in a master, do the following:

1. Create a segment row template. To do so, in the Project Explorer, click the **Templates** tab, open the channel you want to create the segment template for (e.g. Mobile), click the context menu arrow of **Segments**, and then click **New Template**. The template is created and opens on the Visualizer Canvas. It might be a good idea to rename the template to something descriptive, such as rowMobile. To do so, on the **Templates** tab, click the context menu arrow of the newly-created template, and then click **Rename**.
2. Drag a FlexContainer widget onto the new template on the Visualizer Canvas, and configure it how you want.
3. Drag onto the FlexContainer the widgets that you want the segment row template to have, and configure them how you want.
4. Once you have the segment row template configured the way you want it, if you want to copy it to other channels, click the context menu arrow of the configured template, and then click **Copy**. Navigate to and open the channel you want to paste it to, click the context menu arrow of **Segments**, and then click **Paste**. Rename the template if you'd like, and then make any needed modifications so that it looks and behaves correctly in that channel.
5. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **Save All**.

6. Add a Segment2 widget to the master. To do so, in the Project Explorer, on the **Templates** tab, in the Masters section, select the master so that it opens on the Visualizer Canvas, add (or select an existing) container widget in the master, and then drag a Segment2 widget onto it.
7. In the Properties Editor for the Segment2 widget you just added, click the **Segment** tab. Then from the Row Template drop-down list, select the row template you want to use.
8. From the **Segment** tab, click the **Edit** button for Master Data, and make any changes needed to the segment data.
9. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **Save All**.

Create Widgets from a Master

You can use a master as the basis for a form and then create widgets from the master that are no longer associated with it. This gives you the freedom to create a form that is consistent with other forms, but which you can modify totally independent of the master that it was originally an instance of.

To create widgets from a master, do the following:

1. On the **Project** tab of the Project Explorer, locate and click the flex container for the instance of the master you want to create the widgets from. The instance displays on the Visualizer Canvas and is now the container with focus.

   > ***Note:*** You must click the instance's flex container, not its parent nor its children, to create widgets from the master.

2. On the Visualizer Canvas, right-click the instance, and then click **Create Widgets From Master**. The instance of the master is converted to widgets that have no association with the master.

   > ***Note:*** If you do not see **Create Widgets From Master** in the context menu, you probably do not have the correct container selected on the Project tab of the Project Explorer. You likely have either a parent or child selected.

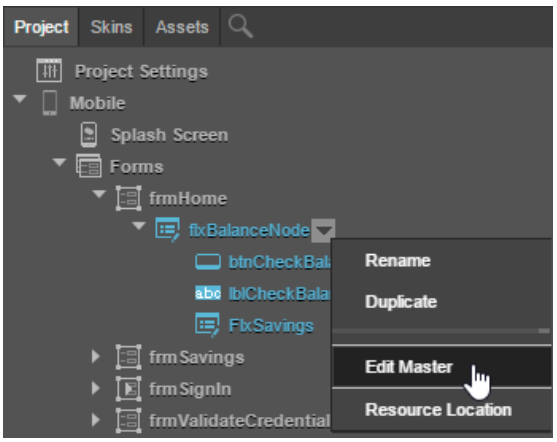
Edit a Master

Editing a master is virtually identical to editing any form. The key difference is that the changes you make to the master can propagate to every instance in your project where that master is used.

You can navigate to the master from the **Templates** tab, where the master resides, or from any instance of a master.

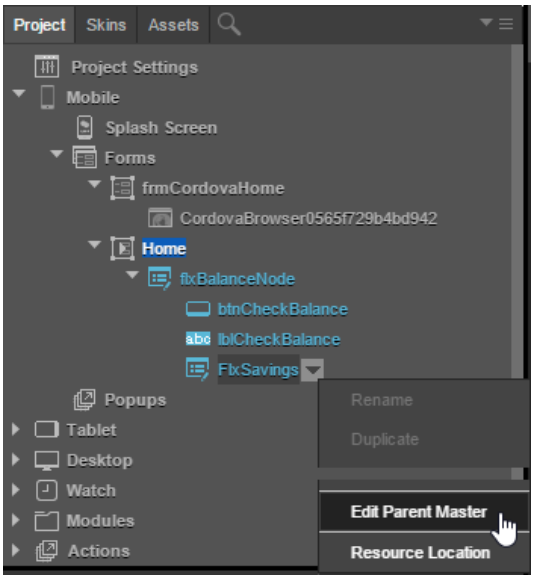
To edit a master, do the following:

1. Navigate to the master, either from the **Templates** tab, or from an instance of the master.
    - **From the Templates tab.** Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
    - From an instance of the master. Instances of masters have blue text. From the **Project** tab of the Project Explorer, click the context menu arrow of the master instance that you want to edit. The select one of the following commands, depending on your what you select.
        - If you select the top-level container of the master, click **Edit Master**.



        - If you select an element under the top-level container, such as a widget or an

embedded master, click **Edit Parent Master**.



Kony Visualizer navigates to the **Templates** tab, selects the top-most flex container of the master you selected, and opens the master on the Visualizer Canvas.

2. If it isn't expanded already, expand the **Masters** category.
3. Open the master you want to edit by double-clicking its name.
4. On the Visualizer Canvas, make the changes you want to make, just as you would to any other form. This includes all activities, such as adding and removing widgets, changing their properties, setting up action sequences, and so on. The changes you make to the master propagate to every instance in your project where that master is used.

Move the Templates Tab

By default, the **Templates** tab (which displays masters) and the **Project** tab (which displays forms) occupy the same pane (i.e. the Project Explorer) and are therefore not viewable at the same time. If you prefer, you can move the **Templates** tab from the Project Explorer to the Library Explorer, which makes it possible to view both tabs at the same time since they're no longer occupying the same pane.

To move the Templates tab, do the following:

1. On the Project Explorer, right-click the **Templates** tab.
2. Click **Dock to Library**. The **Templates** tab moves from the Project Explorer to the Library Explorer.

Duplicate a Master

To duplicate a Master, do the following:

1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. If it isn't expanded already, expand the **Masters** category.
3. Hover over the name of the master you want to duplicate, click its context menu arrow, and then click **Duplicate**.

Identify the Master Being Used

When you are working within an instance of a master, you can identify the name of the master using the Properties Editor.

To identify the master being used, do the following:

1. Open the form that is an instance of a master.
2. In the Properties Editor, click the **Look** tab. The **Master Name** property lists the name of the master being used. If the **Master Name** property is not listed, the form is not an instance of a master, and is therefore not associated with a master.

Disable Widget Data Changes outside a Master

To disable widget data changes outside a master, do the following:

1. Navigate to the form that contains the widgets you want to lock relative to the master.
2. On the Properties pane, click the FlexContainer tab, and then switch Reference Data to **On**.

Hide and Unhide Widgets in Instances of a Master

By default, when you create a new master, the widgets in instances of the master, as they appear on the **Project** tab of the Project Explorer, are hidden; only the top-level flex container is visible. This is especially helpful if you want to help prevent the inadvertent modification of widgets and their properties in instances of a master. These hidden widgets are still visible on the Visualizer Canvas, but they are not individually selectable or editable.

> ***Note:*** Widgets in instances of masters for projects created in versions of Kony Visualizer earlier than 7.2 are, by default, unhidden on the **Project** tab of the Project Explorer.

To hide or unhide the widgets in an instance of a master, do the following:

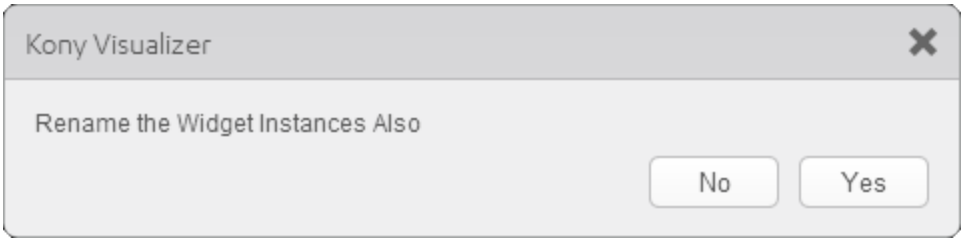
1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. If it isn't expanded already, expand the **Masters** category.
3. Expand the master whose widgets you want to hide or unhide in instances of that master.
4. Click the topmost flex container of the master.
5. In the Properties Editor, on the **FlexContainer** tab, do one of the following:
    - **Hide master widgets.** To hide the widgets in instances of a master, click set Hide Master Widgets to **On**. The widgets in instances of the master are no longer listed in the hierarchy of screen elements on the **Project** tab of the Project Explorer. Only the topmost flex container displays in the hierarchy. On the Visualizer Canvas, the widgets of the master instance are visible, but not selectable or editable.

- **Unhide master widgets.** To unhide the widgets in instances of a master, click set Hide Master Widgets to **Off**. The widgets in instances of the master are listed in the hierarchy of screen elements on the **Project** tab of the Project Explorer, and are selectable and editable on the Visualizer Canvas.

Rename a Widget in a Master

To rename a widget in a master, do the following:

1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. If it isn't expanded already, expand the **Masters** category.
3. Expand the master that has the widget you want to rename.
4. Hover over the name of the widget you want to rename, click its context menu arrow, and then click **Rename**. The name becomes editable and is highlighted.
5. Type the name you want for the widget, and then press **Enter**.
6. A dialog box displays asking if you also want to rename the widget in all the instances where it appears in the project. Depending on what you prefer, click either **Yes** or **No**.



Delete a Master

> ***Important:*** Deleting a master also deletes every instance of that master from your application. Any instances that you don't want to have deleted you should first unlink from the master by creating widgets from it. For more information, see Create Widgets from a Master.

To delete a master, do the following:

1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. If it isn't expanded already, expand the **Masters** category.
3. Hover over the name of the master you want to delete, click its context menu arrow, and then click **Delete**.

Export and Import Masters

With Kony Visualizer, you can export one or all masters, and then import them into another project or onto another computer.

The following table shows what elements are included and excluded from a master export.

Item	Exported	Data Cleared	Comments
Nested masters	Yes	No	-
Templates	Yes	No	-
Skins	Yes	No	-
Images	Yes	No	-
Fonts	Yes	No	-
Action sequences	Yes	No	Only action sequences contained within the master are exported. Indirect references are not exported.
Locales	No	Yes	-
Local files	No	Yes	-
Cordova files	No	Yes	-

Export a Master

To export a master, do the following:

1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. If it isn't expanded already, expand the **Masters** category.
3. Do one of the following, depending on whether you want to export all masters or a single master:
    - **Export all masters.** Click the context menu arrow of the **Masters** category, and then click **Export Masters**.
    - **Export a single master.** Click the context menu arrow of the master you want to export, and then click **Export**.
4. Navigate to the folder where you want to export the .zip file to, rename the file (optional), and then click **Save**.

    A dialog box appears informing you that the file was exported successfully. By clicking the name of the file in the dialog box, a file explorer window opens to the folder that the file was saved to.

Import a Master

To import a master .zip file, do the following:

1. Open the project that you want to import the masters file into.
2. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
3. Click the context menu arrow of the **Masters** category, and then click **Import Masters**.
4. Click the **Browse** button that corresponds to the **File name** text box, navigate to the masters .zip file that you want to import, and then click **Open**.
5. If the master file that you're importing contains components that already exist in the project, check the check box labeled **Duplicate, if master or its components exist**.
6. Click **Import**. Kony Visualizer imports the master(s) in the .zip file.

Add a Master to a Collection

A collection is a visual catalog within the Library Explorer that contains templates. A Kony library can have multiple collections. A template is a group of widgets that you can drag onto a form, providing an easy way to replicate common widget groupings. Along with templates, you can also add masters to a collection, providing you with a visual, simple means of adding a master to a form.

To add a master to a collection, do the following:

1. Depending on where you've placed the **Templates** tab, on either the Project Explorer or the Library Explorer, click the **Templates** tab.
2. If it isn't expanded already, expand the **Masters** category.
3. Click the context menu arrow of the master you want, hover over **Add to Collection**, hover over the library you want, and then click the collection that you want to add the master to.

Additional Information about Masters

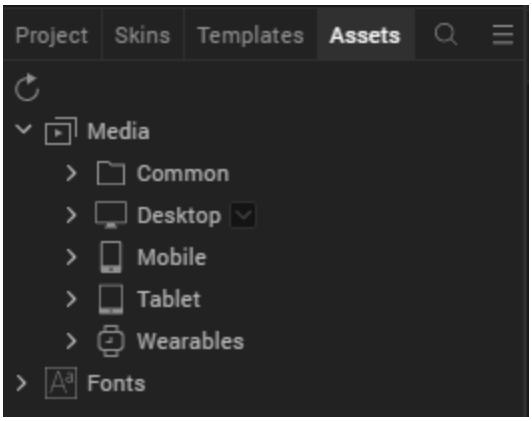
The following are additional facts about masters.

- The nesting of masters is not circular. That is to say, if master B is nested within master A, you cannot then nest master A within master B. An up-level master cannot be nested inside a master that's down-level from it.
- Masters can only be used with flex layout elements; they do not work using the legacy VBox and HBox layout elements, or pop-ups.
- You cannot modify the skin of an instance of a master's widget. If you want to change the skin of a widget that's based on a master, you must make the change to the widget in the master itself.

Add and Manage Images and Other Media

While designing an application, you will have the need to use images, videos, and other media in a form. To do so, you copy the media to a specific folder in the Workspace and then use the appropriate widget to insert the media in a form. You can see what media are a part of your project on the **Assets** tab of the Project Explorer. You can also specify media for a specific channel and then use them in your application.

After adding media to a media folder, to see them reflected in Kony Visualizer, refresh the list of media assets by clicking the **Refresh Folders** icon located on the top left corner of the **Assets** tab in the Project Explorer.



This section covers the following topics:

Conventions for Images

Import and Use Images

Troubleshoot Invalid Images

Manage Android Image Folders

Add a Favorite Icon (FavIcon) for Use in a Browser

Add a Touch Icon for Apple for Use in a Browser

Android App Icons - Mipmap Drawables

Conventions for Images

This section covers the following subjects:

Image Assets Based on Dots Per Inch (DPI) or Form Factors

Applications must have right image assets to handle various DPIs. SPA platform enables automatic detection of the device DPI by default and requests for the right set of images based on the DPI.

For devices with different form factors but same DPI, the SPA platform uses the device screen width as a factor while requesting the appropriate image assets. SPA platform also handles screen orientation specific asset requests.

Gestures Using DPI

Besides using the DPI for the resource identification, SPA platform also uses DPI for calculating the threshold limits to identify various gestures like swipe, scroll, tap, and double-tap.

Retina Display on iPhone

iPhone 4.0 has a high resolution screen (960x640) with a density of 326 ppi. When the screen density is more than 300 ppi, images, videos, text, and other objects appear very smooth with high clarity on the screen. This high resolution screen enables a very clear view of all the objects on the screen, which is similar to high quality printed output. Apple has named this technology Retina Display.

All the iPhone applications are developed for 320x480 resolution. The images used in the application are automatically scaled up to suit the high resolution screen. When the images are scaled up for a high resolution screen, the clarity of images is distorted. The Retina Display feature ensures excellent image clarity by preventing automatic scaling.

To support images within an application for Retina Display, do the following:

1. Create an image with twice the pixel dimensions as the original image.
2. Add the following suffix to the image name: `@2x`.

   For example, if you have a company logo image named `logo.png` that is 50x50 pixels, and you want to use it for a high resolution screen, create the same image at a size of 100x100 pixels, and name it `logo@2x.png`.
3. Place this high-resolution image in the Resources folder along with the other images. When you build your project, the resulting app automatically picks up the high resolution images for those devices that support the retina display.

Apple Touch Icon

The Apple Touch Icon or apple-touch-icon.png is a file used for a web page icon on the Apple iPhone, iPod Touch, and iPad. When someone bookmarks your web page or adds your web page to his or her home screen, the Apple Touch icon is used. If this file is not found these Apple products will use the screenshot of the web page, which often looks like no more than a white square.

This file should be saved as a .png, have dimensions of 58 x 58, and be stored in your resources common directory.

Naming Conventions for Images

This section explains the naming convention you should follow while naming the images. The allowed characters in the image file names are:

- Lowercase alphabetical characters

  > ***Note:*** The image name must start with a lowercase alphabetical character.

- Numerical characters
- Underscore (_)
- Period (.)

> ***Important:*** Do ***not*** use spaces in your image file names. Also, do not use any reserved words or keywords as the file names for the images.

You will want to be sure to use the following conventions when naming image files:

Valid File Names	Invalid File Names	Remarks
myicon.png	Myicon.png	Contains an uppercase character.
icon2.png	icon_2.png	Contains an underscore.
accntsummary.png	accnt&summary.png	Contains a special character.
accountdetails.png	2details.png	Begins with a number.
companylogo.png	company logo.png	Contains a space.
flightstatus123.png	continue.png	Contains a JavaScript keyword.

Import and Use Images

To use images in your application, you must import them into Kony Visualizer. For conventional 2D images, use either PNG, GIF, or JPEG image formats while working with an application. You can also use WebP image format for SPA and Desktop Web applications.

> ***Note:*** You cannot import images with the WebP format directly into your applications.
> To access an image with the WebP format in Desktop Web applications, copy the image into <workspace folder>/<application name> /resources /desktop/common.
> To access an image with WebP format in SPA applications, copy the image into <workspace folder >/<application name> /resources/mobile/web/spaandroid.
> After that you can use the src property of the Image widget to dynamically add the images with WebP format.

You can import an image into Kony Visualizer by using the following methods:

- Add an image to the Assets tab, and then use the image.
- Directly import an image into a widget.

> ***Note:*** GIFs are supported only in Image widget.

The 3D formats Kony Visualizer supports are as follows:

- The Autodesk 3DS Max file format (.3ds)
- The COLLADA file format (.dae)

Following are the subfolders in which you can store images, videos, and other media:

- **Common.** The media assets stored here are available across all the channels (mobile and tablet applications) and modes (native and web).
- **Desktop.** The media assets stored here are available for all Responsive Web applications.
    - **Common.** The media assets stored here are available for native and web applications.
    - **Native.** The media assets stored here are available only for native applications.
    - **Web.** The media assets stored here are available only for web applications.
- **Mobile.** The media assets stored here are available for all mobile devices.
    - **Common.** The media assets stored here are available for native and web applications.
    - **Native.** The media assets stored here are available only for native applications.
    - **Web.** The media assets stored here are available only for web applications.
- **Tablet.** The media assets stored here are available for all tablet devices.
    - **Common.** The media assets stored here are available for native and web applications.
    - **Native.** The media assets stored here are available only for native applications.
    - **Web.** The media assets stored here are available only for web applications.

- **Wearables.** The media assets stored here are available for all wearable devices.
    - **Apple Watch.** The media assets stored here are available for only Apple Watch devices.
        - **Common.** The media assets stored here are available for native and web Apple Watch applications.
        - **Native.** The media assets stored here are available for only native Apple Watch applications.
    - **Android Wear.** The media assets stored here are available for only Android Wear devices.
        - **Common.** The media assets stored here are available for native and web Android Wear applications.
        - **Native.** The media assets stored here are available for only native Android Wear applications.

Import Images to Assets Tab

To add images to the Assets tab and then use them in a Kony Visualizer project, follow these steps:

1. In Kony Visualizer, go to **Project Explorer** > **Assets**.
2. Expand the channel to which you want to add the image files; whether Common (to be available to all channels), Desktop, Mobile, Tablet, or Wearables.
3. Expand the environment that you want the image files to be accessible to; whether Common (to be available for all environments), Native, or Web.
4. If you selected the Native environment, expand the platform that you want the SQLite database to be accessible to, whether Common (to be available to all platforms), Android, iOS, or Windows.

   Once the node is fully expanded, a **raw** folder is visible.
5. Click the context menu arrow of the **raw** folder, and then click **Import Media**.

6. Navigate to the folder containing the images you want to import, and then select them, holding down the **Ctrl** key as you click to select non-contiguously listed images, or the **Shift** key to select contiguously listed images.
7. Click **Open**.
8. To see the added files reflected in Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer), click **Refresh**.

Directly Import Images into a Widget

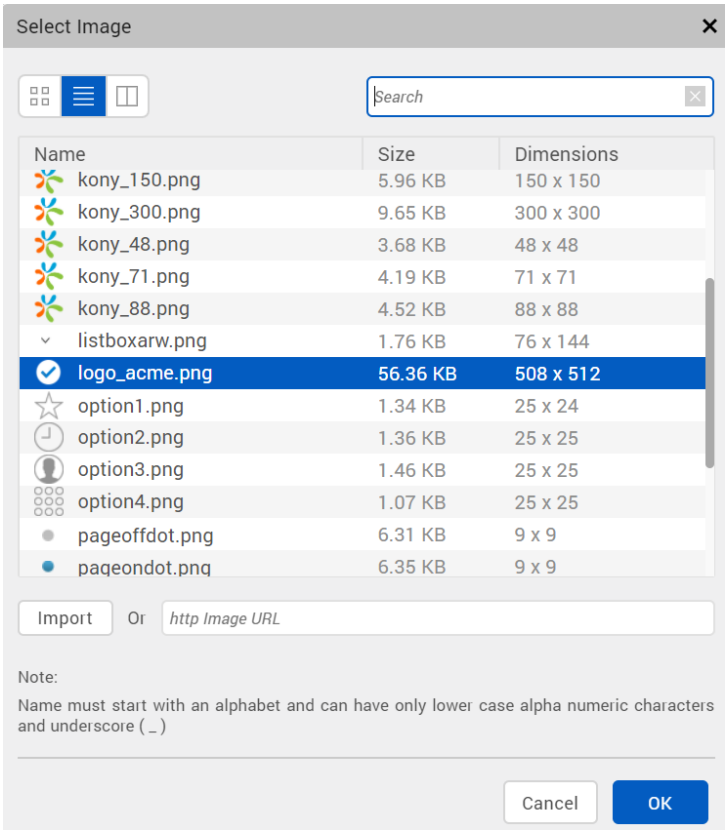
Alternatively, you can directly import images into a widget; without the need to add them to the Assets tab first. You can directly import images into an Image widget, as the skin of a widget (for example, Button), or into any widget property that takes an image name (for example, the selected and unselected images in a CheckBoxGroup). As an example, we have discussed the process of how to directly import images into an Image widget later in this section.

Furthermore, you can directly import images into the widgets inside a component. For more information, refer the Import and Add Images to a Component section.

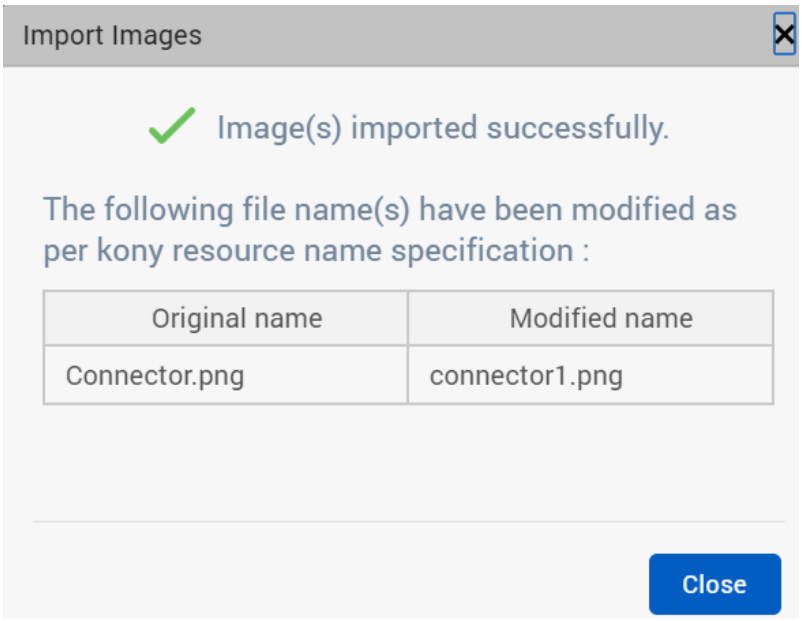
To directly import images into an Image widget, follow these steps:

1. From the **Default Library**, drag and drop an Image widget to your form.
2. You can perform any of the following actions:
   - On the Project Canvas, double-click the Image widget.
   - In the Project Explorer, expand your form, right-click the Image widget, and then click **Edit Image Source**.

The **Select Image** dialog box appears, containing the list of available imported images.



3. Click **Import**. The File Explorer of your local computer appears.
4. Select the required image, and then click **Open**. The Import Images dialog box appears, indicating that the image has been imported successfully. If required, Kony Visualizer automatically modifies the file name of the imported image according to the Kony resource name specification.



5. The imported image appears in the list of images of the **Select Image** dialog box. Select the image, and then click **OK**. The image is added to the Image widget.

Troubleshoot Invalid Images

If any of the images in the Media folder of the **Assets** tab have an invalid file name or are not a PNG file, the Media folder icon displays a yellow triangle with an exclamation mark. Similarly, the icon of any Media subfolder that contains such an invalid image displays a yellow triangle with an exclamation mark. To resolve this, you need to identify the invalid files and correct them.

To troubleshoot invalid images, do the following:

1. On the Project Explorer, click the **Asset** tab. Observe if the Media folder icon displays a yellow triangle with an exclamation mark.
2. If it does, expand the Media folder and for each affected channel, navigate to the lowest-level subfolder that displays the warning icon.

3. Click the context menu arrow of the subfolder with the warning icon, and then click **Resource Location**. Your computer's file browser opens the folder.
4. Examine the file names and the image formats of the files. All images need to be in the PNG format, and all files need to follow the naming conventions described in Naming Conventions for Images. If necessary, use a graphics application to save images that are using invalid file formats as PNG, JPEG, or GIF files.
5. Repeat the last step for any other subfolders in the channel, and then move on to any other affected channels.

   When all files have been corrected, the Media folder icon returns to normal.

Manage Android Image Folders

This section covers the following subjects:

Particulars for managing Android image folders

Range of screen supported in Android

Nine Patch Background Image

Particulars for managing Android image folders

Following are some particulars for managing Android image folders:

- You can create a folder in which resources are added based on the specific scenarios:
  - **MCC and MNC**: Creates a folder based on the Mobile Country Code and Mobile Network code. Add images or resources in the folder based on the MCC and MNC. You can add an image when you right-click the newly created folder and select Add Resources.
  - **Screen Size**: Screen size for which the application is built. Add the images based on the screen sizes specified. You can add an image when you right-click the newly created folder and select Add Resources.

- **Screen Aspect**: Aspect for which the application is built. Add the images based on the screen aspect specified. You can add an image when you right-click the newly created folder and select Add Resources.
- **Screen Orientation**: Can be set as portrait or landscape. A folder is created accordingly under android. You can add an image when you right-click the folder and select Add Resources. Ensure that the image name specified and added is the same as the image name specified in Splash Screen Properties.
- **Dock mode**: Creates a folder based on the dock mode specified. The dock mode can be car or normal. Add images in the respective folder based on the dock mode specified. You can add an image when you right-click the newly created folder and select Add Resources.
- **Night mode**: Mode in which the application is built. Add the images based on the mode specified. You can add an image when you right-click the newly created folder and select Add Resources.
- **Screen pixel density**: Creates a folder based on the dpi of the device. Add the images based on the dpi specified.You can add an image when you right-click the newly created folder and select Add Resources.
- **Touchscreen type**: Creates a folder based on the dpi of the device. Add the images based on the touch screen specified.You can add an image when you right-click the newly created folder and select Add Resources.
- **Keyboard availability**: Creates a folder based on the keyboard availability set. Add the images based on the keyboard mode specified.You can add an image when you right-click the newly created folder and select Add Resources.
- **Primary text input method**: Creates a folder based on the input method set . Add the images based on the method specified.You can add an image when you right-click the newly created folder and select Add Resources.

- **Navigation key availability**: Creates a folder based on the navigation key availability. Add the images based on the availability specified.You can add an image when you right-click the newly created folder and select Add Resources.
- **Primary non-touch navigation method**: Creates a folder based on the navigation set . Add the images based on the method specified.You can add an image when you right-click the newly created folder and select Add Resources.
- **Platform Version**: Creates a folder based on the Android Platform version. Add the images based on the method specified.You can add an image when you right-click the newly created folder and select Add Resources.
- **drawable-<language>-r<Region>** when you have resources specific to locales only. For example, if you want to add a resource for en_US locale, then create a drawableen-rUS folder and place the resource in drawable-en-rUS folder.
- **drawable-<screensize>** when you have resources specific to screen size only. Possible screen sizes are small, normal, large, and xlarge. For example, if you want to add a resource for normal screen size, then create a drawable-normal folder and place the resource in drawable-normal folder.
- **drawable-<orientation>** when you have resources specific to orientation only. Possible values for orientation are land and port. For example, if you want to add a resource for landscape orientation, then create a drawable-land folder and place the resource in drawable-land folder.
- **drawable-<screen-pixel-density>** when you have resources specific to resolution or screen-pixel-density only. Possible values for screen-pixel-density are ldpi, mdpi, hdpi, xhdpi, and nodpi. For example, if you want to add a resource for hdpi pixel density, then create a drawable-hdpi folder and place the resource in drawable-hdpi folder.

If you have resources specific to a combination of the above folder, you must follow precedence rules and name the folders appropriately. The precedence rules are as follows:

- Locales or language and region
- Screen size

- Orientation
- Screen pixel density

The following table explains a few examples:

Folder	Description
drawable-en-rUS-land	Resources specific to US-English devices in landscape mode
drawable-port-hdpi	Resources specific to devices in portrait mode with hdpi pixel-density
drawable-en-rUSlarge-land	Resources specific to US-English large screen devices in landscape mode
drawable-en-rUSlarge-land-hdpi	Resources specific to US-English large screen devices in landscape mode with hdpi density

Range of screens supported in Android

A set of four generalized sizes: small, normal, large, and xlarge

> ***Note:*** Beginning with Android 3.2 (API level 13), these size groups are deprecated in favor of a new technique for managing screen sizes based on the available screen width. If you're developing for Android 3.2 and greater, see Declaring Tablet Layouts for Android 3.2 for more information.

A set of six generalized densities:

- ldpi (low) ~120dpi
- mdpi (medium) ~160dpi

- hdpi (high) ~240dpi
- xhdpi (extra-high) ~320dpi
- xxhdpi (extra-extra-high) ~480dpi
- xxxhdpi (extra-extra-extra-high) ~640dpi

> ***Note:*** For more information on images, see Supporting Multiple Screens on the Android Developer site.

Nine Patch Background Image

Kony Visualizer supports Android's NinePatch image class. A NinePatchDrawable graphic is a resizeable bitmap image that Android automatically resizes to accommodate the contents of the view in which you have placed it as the background. Benefit of the NinePatch image is the ability to define what can be stretched. You can define what parts can be stretched and what remains as it was originally designed. This way you can have content appear without distortion on different screen size devices.

Use Case

- Used as background image for a segment, textbox, popup and other widgets whose content increases dynamically. This image does not look stretched or loose proportions on different screen sizes.
- Another advantage is memory, same small size image can be reused for different screen size devices.
- Used as splash screen images to occupy complete width and height on different screen sizes.

> ***Note:*** For detailed information on NinePatch image, see http://developer.android.com/guide/topics/graphics/2d-graphics.html#nine-patch.

On Windows, a draw9patch.bat file exists in the Android SDK folder within the tools folder. On Mac, a draw9patch executable file exists in the Android SDK folder within the tools folder. Launch this tool to create 9-patch images. This tool is developed by Android.

To launch the NinePatch tool, do the following:

1. Navigate to Android SDK or tools folder.
2. Run the file draw9patch.bat to launch the Draw 9-patch tool.
3. Drag the PNG image that you want to edit into the Draw 9-patch window.
4. The image appears with a blank one-pixel border around it. Click within the 1-pixel perimeter to draw the lines that define the stretchable patches.
5. Remove any mistakes by holding the Shift key and clicking on a marked pixel.
6. Once the editing is done, on the **File** menu, click **Save 9-patch**.You must save it with the file name format `<imagename>.9.png`, saving it to the following folder: `\resources\mobilerichclient\android`

> ***Note:*** For more information about using the 9-patch tool, see http://developer.android.com/guide/developing/tools/draw9patch.html.

Add a Favorite Icon (Favicon) for Use in a Browser

A favicon file is a .ico file containing one or more small icons for display in web browsers, and most commonly appear next to a web site's URL, on a browser tab, and can also be associated with shortcuts and bookmarks.

To add a favorite icon for use in a browser, do the following:

1. On the **File** menu (the **Project** menu in Kony Visualizer), and then click **Settings**.
2. Click the **Desktop Web** tab.
3. For the **Web Browser (favicon.ico)** field, click its corresponding **Browse** button (the **Edit** button in Kony Visualizer).
4. Navigate to the file you want to use, select it, and then click **Open** (**OK** in Kony Visualizer).

> ***Note:*** Kony Visualizer Classic uses .ico files. Kony Visualizer uses .png files.

5. Click **Finish** (**Apply** in Kony Visualizer).

Add a Touch Icon for Apple for Use in a Browser

The Apple Touch Icon or apple-touch-icon.png is a file used for a web page icon on the Apple iPhone, iPod Touch, and iPad. When someone bookmarks your web page or adds your web page to his or her home screen, the Apple Touch icon is used. If this file is not found these Apple products will use the screenshot of the web page, which often looks like no more than a white square.

This file should be saved as a .png, have dimensions of 58 x 58, and be stored in your resources common directory.

To add a touch icon for Apple for use in a browser, do the following:

1. On the **File** menu (the **Project** menu in Kony Visualizer), and then click **Settings**.
2. Click the **Mobile Web** tab.
3. For the **iPhone Shortcut** field, click its corresponding **Browse** button (the **Edit** button in Kony Visualizer).
4. Navigate to the file you want to use, select it, and then click **Open** (**OK** in Kony Visualizer).
5. Click **Finish** (**Apply** in Kony Visualizer).

Android App Icons - Mipmap Drawables

Android guidelines recommend to use Mipmap drawables for launcher icons. The advantage of using mipmap icons is described in detailed below.

When you target your application to support a specific set of device densities, (by specifying android.splits.abi block in build.gradle), Android build eliminates other resources from the drawable folders that are not relevant to a user's device density from apk. When a launcher app wants to upscale a low-resolution icon, the icon might look blurred. A user can use the Mipmap drawable folder to store app launcher icons for all densities. Android system never removes files in the Mipmaps folder that is packed into apk. Packing Mipmaps folder into the apk ensure that right icon is picked up by launcher apps for best resolution.

- App uses Mipmap/ folders instead of the drawable/ folders for launcher icons.
- If you are targeting a specific density deice, ensure to add higher resolution versions of the icons to enhance the visual experience of the icons on higher resolution devices. For example, if you target xxhdpi, ensure to pack xxxhdpi launcher mipmap icons.

Adding Mipmap Resource

To add a Mipmap resource folder,

1. In your platform folders, create a mipmap folder. For example, navigate to **Resources** > **Channel** > **Native** > **Platform**
2. Create a folder, for example **mipmap-<screen-pixel-density>**.
   Some possible values for screen-pixel-density are **ldpi** (low), **mdpi** (medium), **hdpi** (high), **xhdpi** (extra-high), **xxhdpi** (extra-extra-high), and **xxxhdpi**(extra-extra-extra-high).
3. Alternatively, you can also add mipmap resources in the following manner.
   Copy **mipmap> <screen-pixel-density>** folders for the following:
   For mobile - **<WorkSpace>\<Application>\resources\mobile\native\android**
   For tablet - **<WorkSpace>\<Application>\resources\tablet\native\androidtab**

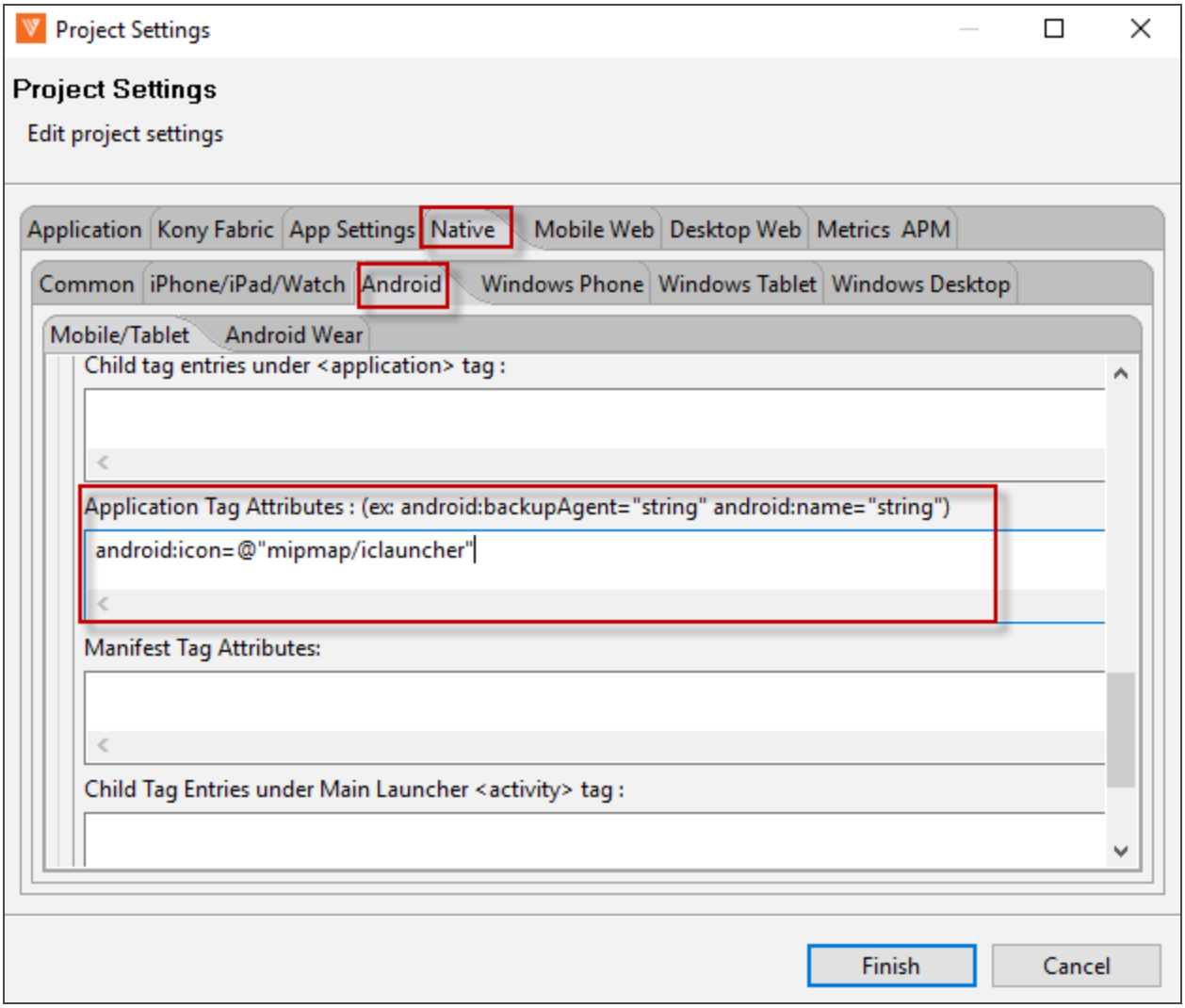
Configure Mipmap in Project Settings

To ensure that Mipmaps feature works, you must do the following.

1. Navigate to **Settings** > **Native** > **Android** > **Application-Tag-Attributes**
2. Enter the following code in it.
   `android:icon="@mipmap/<iconImageName without extension>"`

For example, if the image name is launcher.png

```
android:icon="@mipmap/launcher"
```



Configure Splash Screens

A splash screen is the initial screen that appears when you launch an application.

This topic covers the following procedures:

Configure the Splash Screen for the Mobile Channel

Configure the Splash Screen for the Tablet Channel

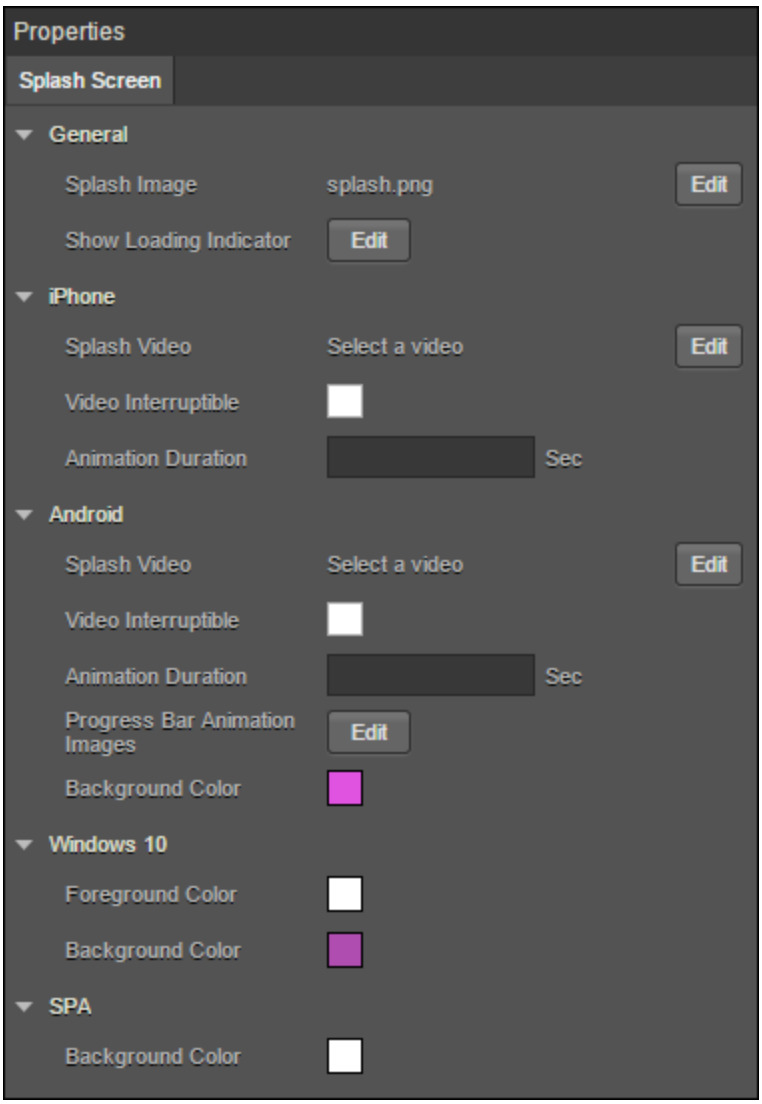
Configure the Splash Screen for the Desktop Channel

Clear the Splash Screen of a Channel

Configure the Splash Screen for the Mobile Channel

To configure the splash screen for the mobile channel, do the following:

1. In the Project Explorer, on the **Properties** tab, expand the Mobile channel, and then click **Splash Screen**. Its properties display in the Properties pane.



Note: If you want to enable a Native iOS experience for the splash image in iOS devices, select the **useLaunchScreenStoryBoard** option. When this property is enabled, Xcode's Storyboard tool is used to display the app's launch screen instead of the splash screen. You must add scale-specific images that are to be displayed as the launch image in the following format for iPhones:
"splashiphone@1x.png"
"splashiphone@2x.png"
"splashiphone@3x.png"

2. In the General section, click **Edit** to select a Splash Image. The Splash Image dialog appears.
    - If you want a default splash image for all the platforms, select default option under platform and assign a desired image.
    - If you want to set image for selected platforms, you need to select required platform and an image respectively.

    > ***Note:*** Splash screen Image names should not have spaces, uppercase letters, special characters. Supported formats for the image are PNG, JPEG, and GIF.

3. In the General section, click **Edit** to configure a Loading Indicator. This property is applicable only for iPhone, Android, Windows platform.
    - The Show Loading Indicator dialog appears in which you need to select required platform and set value to true in order to show loading indicator.
    - If you want to set same value for all platforms, select default option under platform and set its value to true or false.

4. Click **Edit** to select Splash Video. Choose Splash Video dialog appears in which you need to select a video (Make sure the video is placed in common folder under Resources). This property is applicable only for iPhone and Android platforms.

    > ***Note:*** Splash video occupies the time between start of the application till the initialization of the application. Once the application gets initialized, it ends the video (even if it is not complete) and launches the application. The preferable video formats are MP4 and M4V.

    > ***Note:*** For iOS apps, to mix the audio of the splash video with other music apps (running in the background) you must add a new **mix audio** attribute in the **infoplist_configuration.json** file of the app in the **resources** file of the app project.
    >
    > `"mixAudio":true`

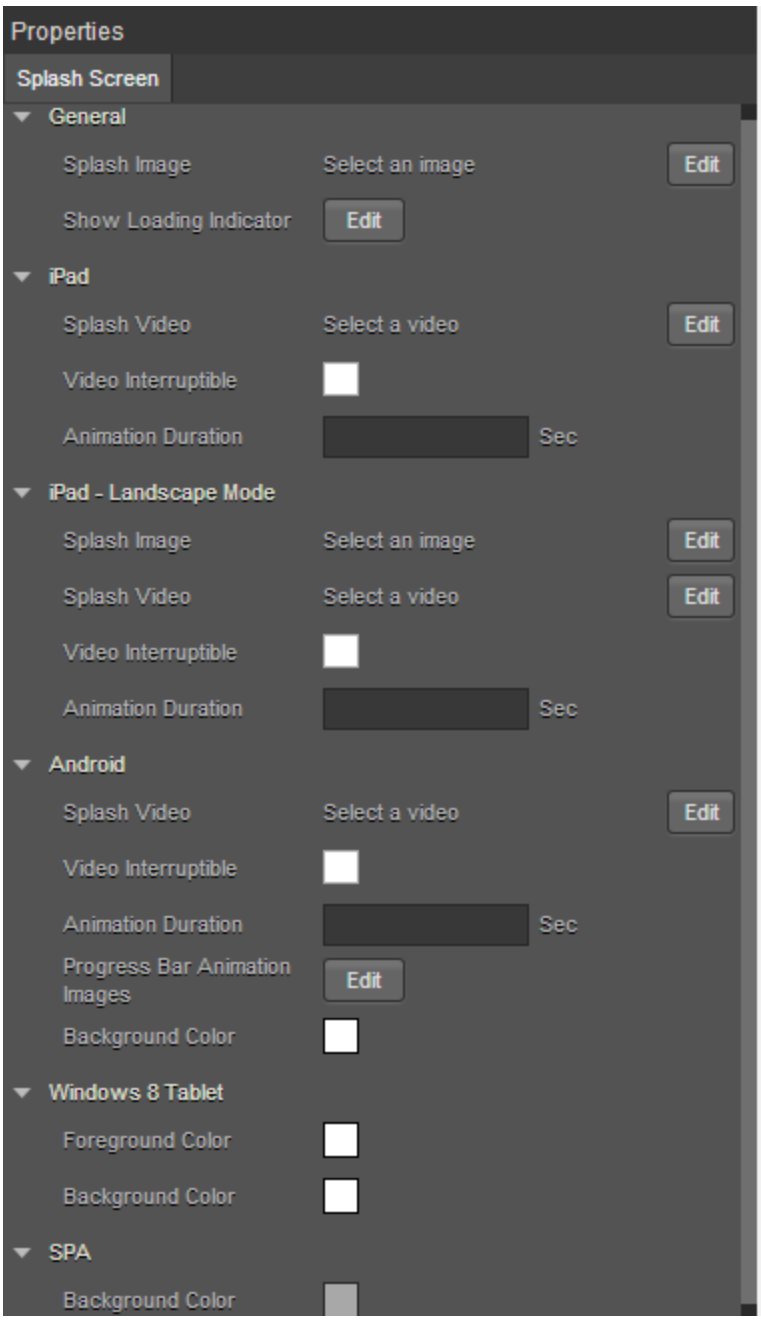
5. Select if the video can be interrupted using Video Interpretable check box.

6. Specify the Animation Duration in seconds. Each progress bar image you specify appears on the screen for the specified number of seconds.
7. Specify the Progress Bar Animation Images by clicking the **Edit** button. Browse and select to add each progress bar animation image. This property is applicable only for Android.
8. Select the Background Color
    - Under Android and Windows, the Color picker dialog appears in which you need to choose a desired background color.
    - Under SPA, the Color picker dialog appears in which you need to choose a desired background color.

Configure the Splash Screen for the Tablet Channel

To configure the splash screen for the tablet channel, do the following:

1. In the Project Explorer, on the **Properties** tab, expand the Tablet channel, and then click **Splash Screen**. Its properties display in the Properties editor.



> ***Note:*** If you want to enable a Native iOS experience for the splash image in iOS devices, select the **useLaunchScreenStoryBoard** option. When this property is enabled, Xcode's Storyboard tool is used to display the app's launch screen instead of the splash screen. You must add scale-specific images that are to be displayed as the launch image in the following format for iPads:
> "splashipad@1x.png"
> "splashipad@2x.png"

2. In the General section, click **Edit** to select a Splash Image. The Splash Image dialog appears.
    - If you want a default splash image for all the platforms, select default option under platform and assign a desired image.
    - If you want to set image for selected platforms, you need to select required platform and an image respectively.
    - You need to set image for iPad-Landscape Mode separately.

    Note: Splash screen Image names should not have spaces, uppercase letters, special characters. Supported formats for the image are PNG, JPEG, and GIF.
3. In the General section, click **Edit** to configure a Loading Indicator. This property is applicable only for iPad, Android, Windows platforms.
    - The Show Loading Indicator dialog appears in which you need to select required platform and set value to true in order to show loading indicator.
    - If you want same value for all platforms, select default option under platform and set its value to true or false.
4. Click **Edit** to select Splash Video. Choose Splash Video dialog appears in which you need to select a video (Make sure the video is placed in common folder under Resources). This property is applicable only for iPad and Android platforms.

Note: Splash video occupies the time between start of the application till the initialization of the application. Once the application gets initialized, it ends the video (even if it is not complete) and launches the application. The preferable video formats are MP4 and M4V.

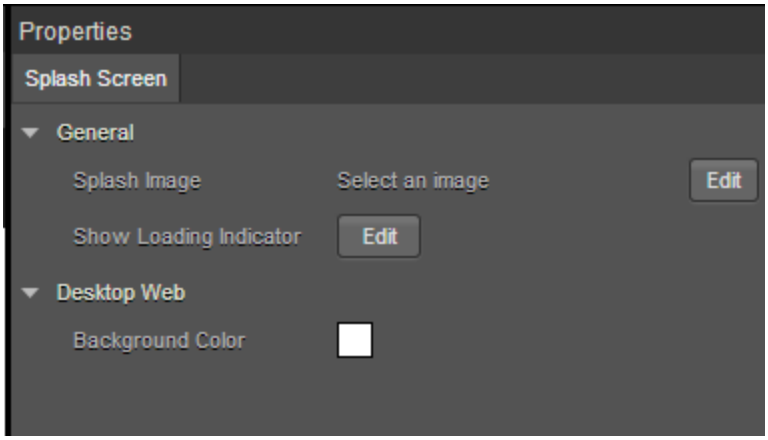
5. Select if the video can be interrupted using Video Interpretable check box.
6. Specify the Animation Duration in seconds. Each progress bar image you specify, appears on the screen for the specified number of seconds.
7. Specify the Progress Bar Animation Images by clicking the **Edit** button. Browse and select to add each progress bar animation image. This property is applicable only for Android.
8. Select the Background Color.
    - Under Android, the color picker dialog appears, in which you need to choose a desired background color.
    - Under SPA, the color picker dialog appears in which you need to choose a desired background color.

Configure the Splash Screen for the Desktop Channel

To configure the splash screen for the Desktop channel, do the following:

1. On the Project Explorer, on the **Properties** tab, expand the Desktop channel, and then click **Splash Screen**.
2. In the General section, click the **Edit** button that corresponds with **Select an image**. The Splash Image dialog displays.
3. appears in which you need to select Desktop Web platform and an image.

    Note:Splash screen Image names should not have spaces, uppercase letters, special characters. Supported format for the image is PNG. Supported formats for the image are PNG, JPEG, and GIF.



4. In the General section, click **Edit** to configure a Loading Indicator. The Show Loading Indicator dialog appears in which you need to select Desktop Web platform and set value to true in order to enable this property.
5. In Desktop Web section, select Background Color. The color picker dialog appears in which you need to choose a desired background color.

Clear the Splash Screen of a Channel

You can clear the splash screen configuration so that you can either restart the configuration or do without a splash screen.

To clear the splash screen configuration of a channel, do the following:

1. In the Project Explorer, on the **Project** tab, expand the channel for which you want to clear the splash screen.
2. Click **Splash Screen**.
3. In the Properties Editor, on the **Splash Screen** tab, click the **Edit** button associated with the Splash Image property.
4. Ensure that the check box for the platform you want to clear the splash screen from is checked, then click that platform's Value cell.

5. Click once any image file that is listed so that it is selected in the list, then click it again so that no image in the list is selected.
6. Click **OK** to close the Select Image dialog box.
7. Repeat steps 4 through 6 for any other platforms that you want to clear the splash screen from.
8. Click **OK** to close the Splash Image dialog box.

- In the Project Explorer, on the **Project** tab, click the context menu arrow for the desired channel (Mobile, Tablet, or Desktop), and then click **Clear Splash Screen**.

Refresh the Project

In the course of working on your project, you may have added resources or made changes that are not yet visually reflected in the various panes and tabs of Kony Visualizer. You can update the project to reflect these changes by refreshing it.

To refresh the project, do the following:

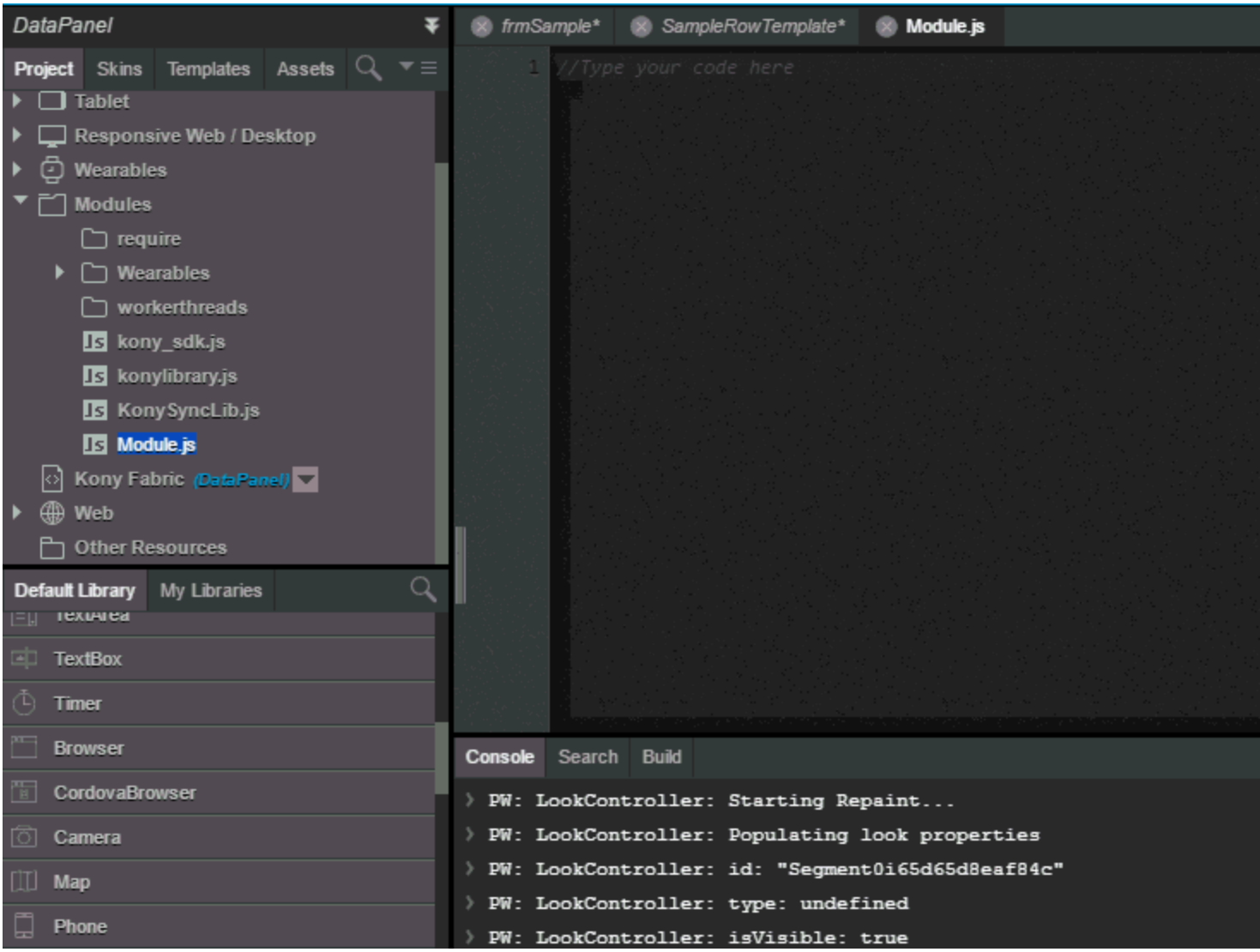
1. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **Refresh**.
2. If your project has any unsaved files and you want to save those changes before refreshing, click **Yes** if prompted to save your changes. If you click **No**, the project will refresh and your changes will be lost.

File Update Notification

In Visualizer, you can edit some files using Visualizer code editor. These files can also be modified externally from their location through an external code editor. When a file is open in Visualizer canvas, if that file is modified externally, then you will receive a notification about the modification.

If the file is not active (open on canvas but not in focus), when you view that file, you will get the notification. You can choose how to handle these modifications.

- When you select **OK**, you override the current file version with the version in the file system.
- When you select **Cancel**, you retain the version that is on Visualizer.



Open an External File

As you create an app and add functionality to it, you may want to incorporate code or other data that resides in a file located outside the project you're working on. With Kony Visualizer, you can open such files so that you can copy the data you need and add it to your app.

To open a file that is external to your current project, do the following:

1. On the **File** menu (the **Project** menu in *Kony Visualizer*), click **Open File**.
2. Navigate to the file you want to open, and then click **Open**. The file opens on the Visualizer Canvas.
3. Select the portion of the file that you want, and then from the **Edit** menu, click **Copy**.
4. Open the asset where you want to paste the data you copied, place the insertion point where you want to paste, and then from the **Edit** menu, click **Paste**.

Disable Resizing an Application

Applies to Kony Visualizer Classic.

With Kony Visualizer, you can provide an option to disable resizing of application in Windows Desktop platform.

To disable Resizing of application for Windows Desktop, do the following:

1. Create a **win7build.properties** file in the project directory of kony Visualizer application.
2. Open the File and add below statement in it.
   `windesktop_disableapplicationresize = true`
   Set the property to True for disabling application resize.
3. Build the project normally.

Open Console Logs

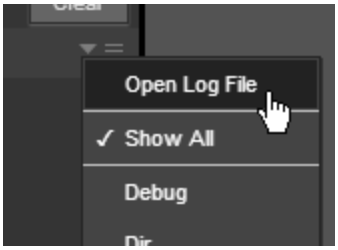
For performance reasons, only the 50 most recent log statements are displayed on the Logs tab of the Console. However, if you want to see earlier log activity, you can open the complete log file. The log file contains statements tracking all Kony Visualizer activity and user actions. When a log file reaches 5.2 Mb in size, it is archived and a new log file is created, picking up where the previous log file left off.

Log files are located at the following path:

```
C:\Users\<user>\Kony Visualizer\vizdata\logs
```

To open the Console log file, do the following:

1. On the Console, if it's not already the front-most tab, click the Logs tab.
2. In the upper right corner of the Console, click the context menu arrow, and then select **Open Log File**. The log file opens in your computer's default text editor.



Forking

Forking allows you to customize properties uniquely for a platform. You can fork:

- A Form
- A Widget Property
- A Skin

Forking a Form

Forking forms enables you to customize the position and availability of widgets on different platforms.

You fork a form to accomplish the following:

- Position widgets differently on different platforms
- Make widgets available on one platform but not on another platform
- Replace a widget on one platform with a widget on another platform

Important Considerations

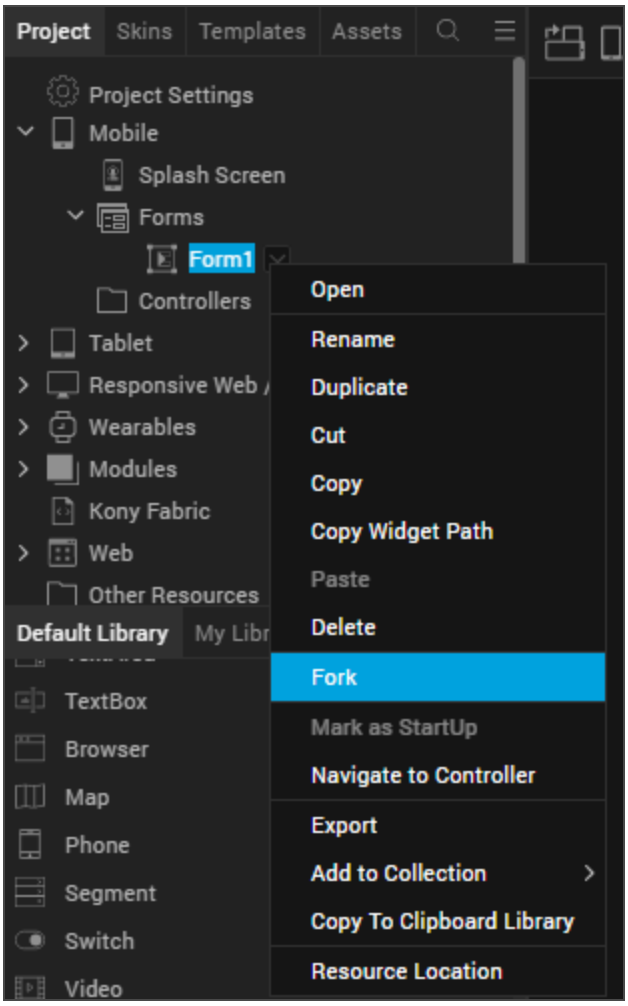
- You can fork Flex forms.
- All the forked forms appear below their source forms in the Project Explorer.
- Renaming a source form renames all its forked forms.
- Deleting a source form deletes all its forked forms.
- A forked form only lists common (i.e. General) properties and platform-specific properties. For example, a form forked for Android Native lists only common properties and Android native specific properties. In the following example, the General properties are not expanded, but are still present.
- Default skins and user-defined skins are also forked during form forking.
- Notes have only those comments made in the forked form's platform. For example, a form forked for Android Native will have only the notes written on an Android device.
- During forking, any unsupported widgets are ignored. For example, Camera Widget is not supported on iOS web. If a form containing a Camera Widget is forked for iOS web, the form will be forked, but the Camera Widget is not forked.

How to Fork a Form

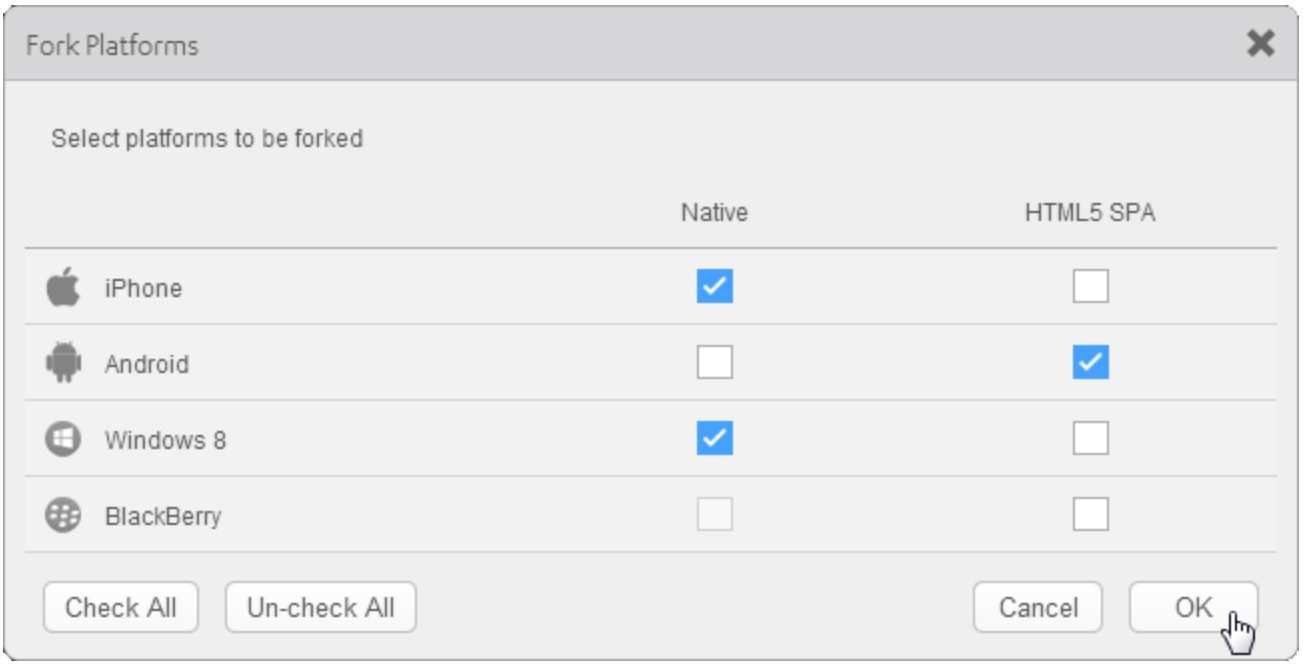
Forking allows you to customize a form specific to a platform. For example, you can fork a form to appear on an iOS (native) platform with green background and have the same form forked to appear on an Android (web) platform with yellow background.

To fork a form, do the following:

1. In the Project Explorer, on the **Project** tab, expand the channel with the form you want to fork, either Mobile, Tablet, or Desktop.
2. Expand Forms, click the context menu arrow of the form you want to fork, and then click **Fork**.The Fork Platforms dialog box displays.



3. Select the platforms for which you want to create a fork, and then click **OK**.



The form is forked for the platforms you selected, which is reflected on the Project tab.

Fork a Widget Property

Forking allows you to provide different values for the same property or skin, across the platforms. You can fork the flex layout and fork all the look properties associated with the flex container.

Only certain widget properties can be forked. You can identify whether a property can be forked by the presence of an icon( ) to the left of the property's name. Two types of forking are available for you:

1. Simple Forking: Allows you to fork one platform at a time.
2. Complex Forking: Allows you to fork multiple platforms at the same time.

Simple Forking

This example illustrates how to do a simple forking, which forks just one platform at a time. In it, the **View Type** property of a **Calendar** widget is forked, but the same principle is applicable for all simple forking of widget properties.

To fork an individual property for a particular platform, do the following:

1. On the Visualizer Canvas, select the widget whose property you want to fork.
2. Select the desired platform from the Platform list–for example, Android : Native.
3. In the Properties Editor, on either the Look tab or the properties tab for the specific widget (which, since our example is a Calendar widget, the properties tab is the Calendar tab), configure the value of the property you want to fork. In this example, we'll select **Popup Grid** from the **View Type** drop-down list.
4. Click the fork icon located to the left of the property you changed, The icon changes to , indicating that the value is forked specifically for the platform you chose, which in this example is the Android : Native platform.

   When you navigate to a different platform, the value of the forked property is replaced either with:

   - A default value of the property.
   - A forked property value, if the property is forked for that specific platform.

Complex Forking

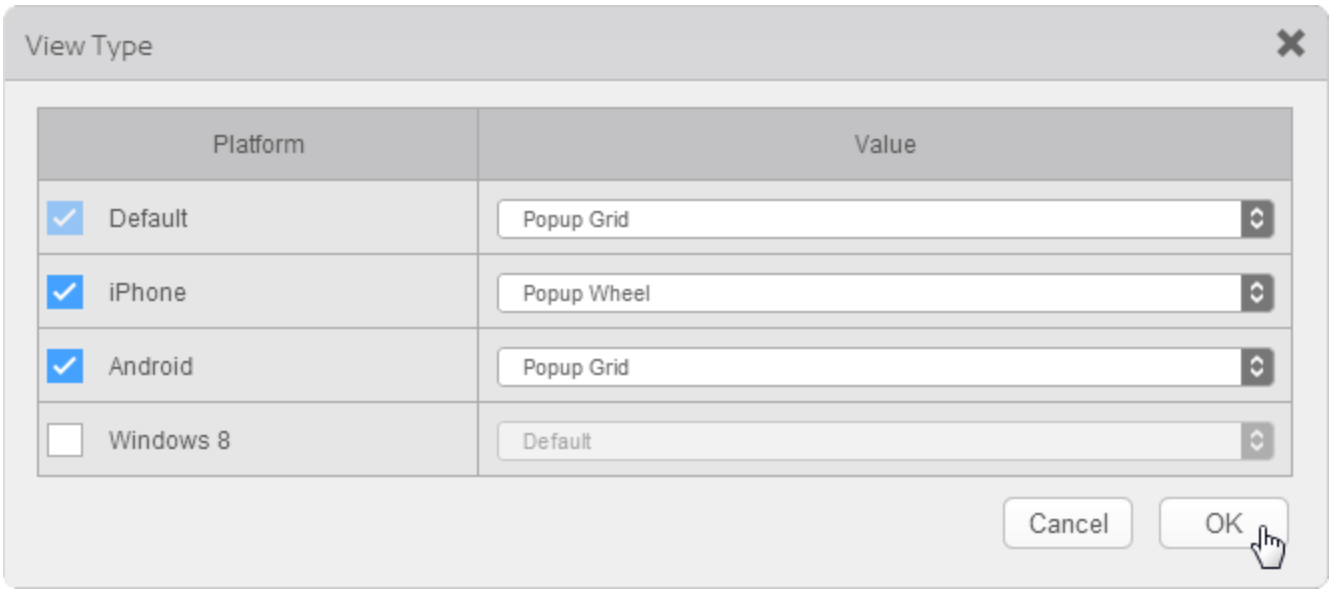
This example illustrates how to do a complex forking, which forks multiple platforms at the same time. In it, the **View Type** property of a **Calendar** widget is forked, but the same principle is applicable for all complex forking of widget properties.

Properties for which complex forking is possible have an ellipsis button located at the far right of the property.

In this section, using the **View Type** property of a **Calendar** widget, let us understand complex forking. The below steps, however, can be used for forking any of the widget properties.

To fork an individual property for multiple platforms, do the following:

1. On the Visualizer Canvas, select the widget whose property you want to fork.
2. In the Properties Editor, on either the Look tab or the properties tab for the specific widget (which, since our example is a Calendar widget, the properties tab is the Calendar tab), configure the value of the property you want to fork. In this example, we'll select **Popup Grid** from the **View Type** drop-down list.
3. Click the ellipsis button to the right of the property you want to fork for multiple platforms.
4. In the dialog box that opens, click the platforms for which you want to fork the property, and then for each platform, select the property value you want. Click **OK**.



The icon changes to , indicating that the value is forked specifically for the platforms you chose.

Configuring Project Settings

Project Settings are the various properties that are to be set for the application. The properties are listed below:

- Application Properties
- Kony Fabric Properties

- App Settings
- Splash Screen Properties
- Mobile Web Properties
- Desktop Web Properties
- Push Notifications
- Native Properties
- Metrics APM Properties

To learn more about various project properties in Kony Visualizer, refer Project Settings in Kony Visualizer.

> ***Note:*** Setting all these properties is a prerequisite for building the application you have developed.

Open Project Settings

Before you proceed with configuring various Project Settings, you must open the **Project Settings** dialog box.

To launch project properties window, follow these steps:

- In Kony Visualizer, click the **Project** menu, and then click **Settings**.

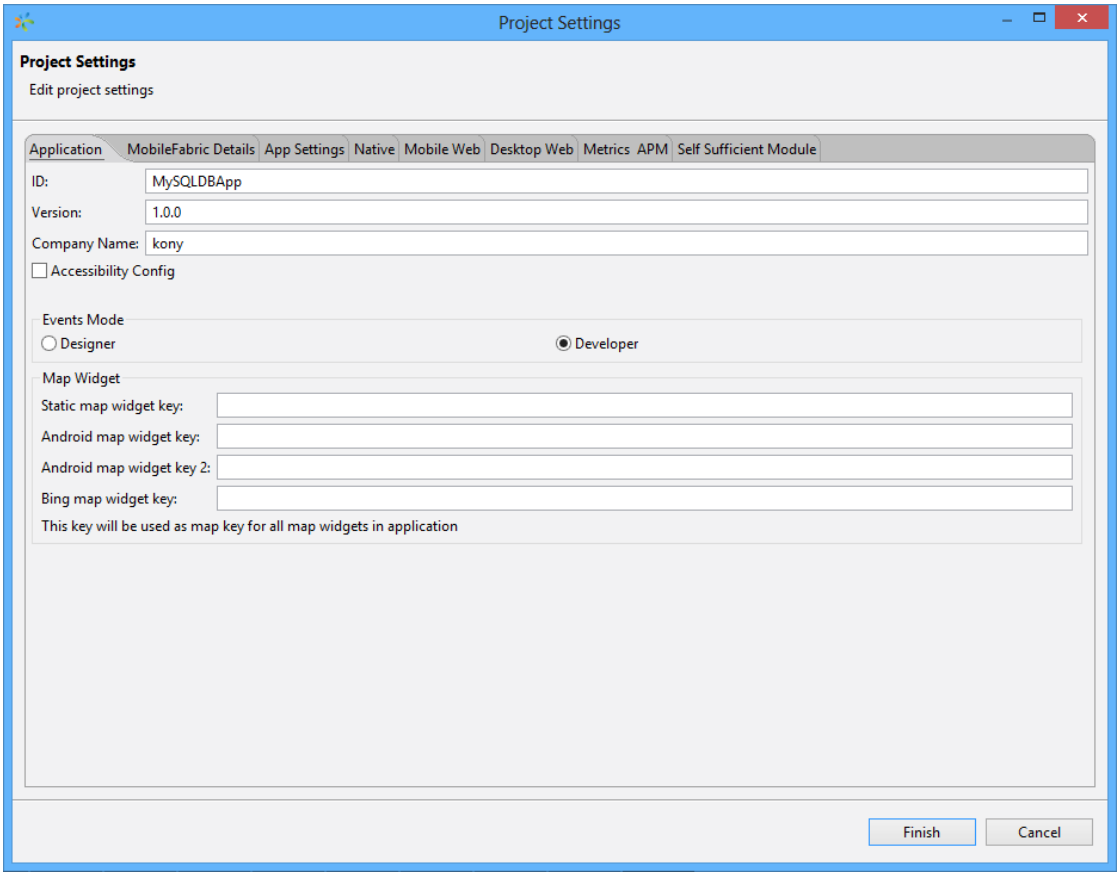
Application Properties

Application properties are specific to the application. These are the properties like *application ID*, *version*, *name*, and *company name*.

To set application properties, do the following:

1. Go to Project Properties window.
2. Click **Application** tab. **ID**, **Version**, **Company Name** fields are automatically populated based on the application you select. You can change these values if required.

> ***Important:*** Do not use `admin` as the app ID as this is a keyword.



3. Enter a value for a key in the all the **Map widget key** fields. This key needs to generated. For more information about map key and generating it, see Generating and Configuring Map APIs Keys.
4. Click **Finish**.

Set Application Settings Properties

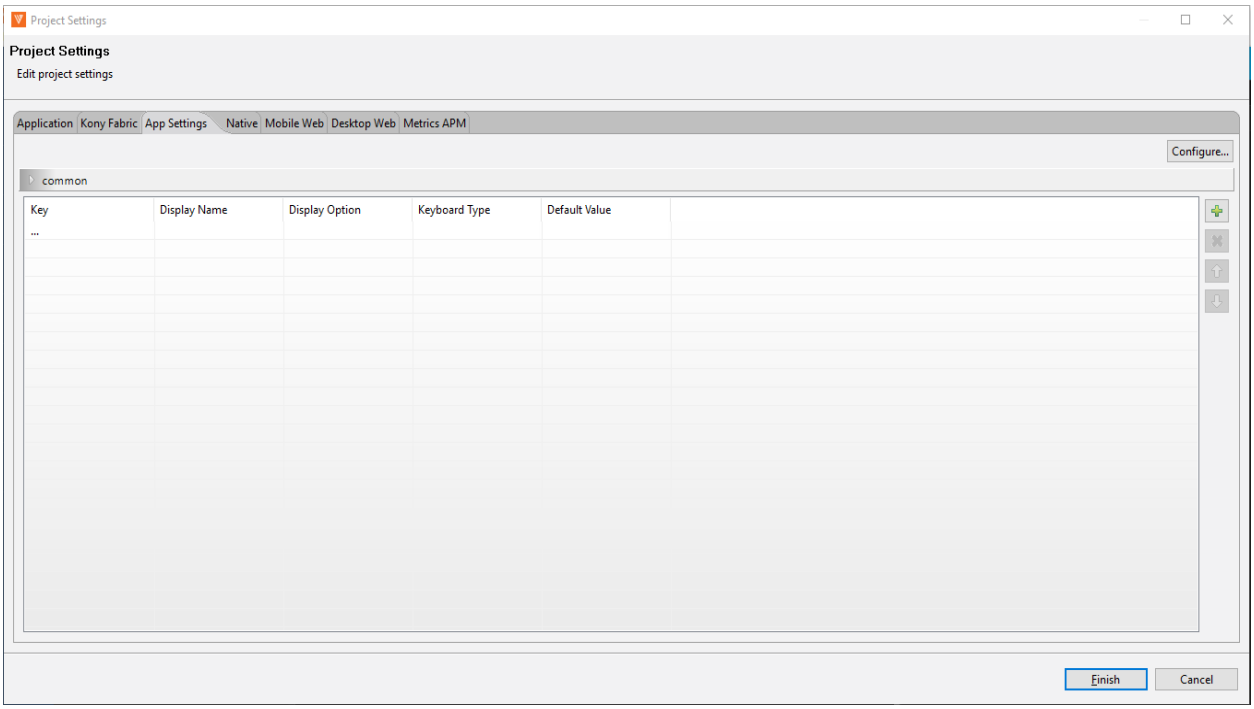
With the **App Settings** tab, you can expose application-level settings so that users of the app have the freedom to modify the properties you expose and change the app's behavior.

For instance, you might want to give users the ability can developers can choose to set the following for an application:

- Define views for widgets.
- Change dynamic properties for widgets.
- Enable and disable components of the application, such as videos.

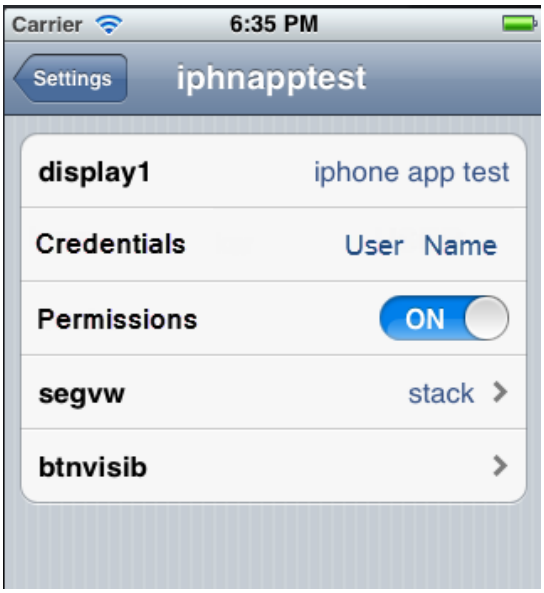
To set App Settings properties, do the following:

1. Go to Project Properties window.
2. Click the **App Settings** tab. The following dialog appears.



3. The settings under the ***common*** header are common across all platforms. You can create platform specific settings under a new header by clicking the **Configure** button and setting platforms on which the setting will be rendered on.
4. Select  to add a setting for an application that the user can configure.
5. Set the following:
    - **Key**: specifies a unique identifier for each setting.
    - **Display Name**: specifies the name to be displayed for a setting(key).
    - **Display Option**: specifies the way in which the settings are set. For example, if you set your display option as a textbox in the settings of the application, once the application is built and deployed , you can set the settings for that key as text in a textbox.
    - **Keyboard Type**: specifies a keyboard type if the **Display Option** set is textbox. For more information on Keyboard types see the Kony Visualizer Widget Programmer's Guide.
    - **Default Value**: Specifies the default value for each **key**.

      The following illustration depicts the settings built and deployed for iPhone that correspond to the Application settings configured in Kony Visualizer.



Use cases for different Display Options

Textbox

You can set a display option as a textbox and use the application settings APIs to read or write the values set to the application. In the figure above User Name is a value entered in a textbox. You can read this key and its corresponding value use it anywhere in the application. The values in this field are stored as a string in appsettings.

Label

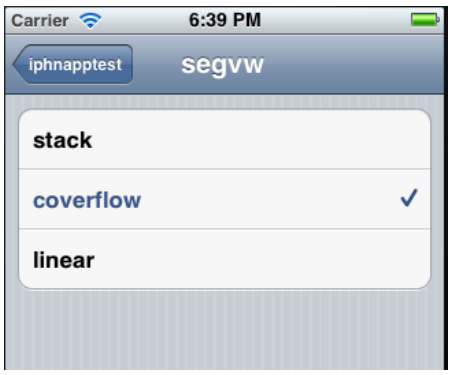
You can set a label in application settings for two purposes. First to read/write the value of the label to the application. Second you can use the label as a separator in the settings area itself to show different types of settings. The values in this field are stored as a string in appsettings.

Switch

You can set permissions for an application using the **Switch** display option. For example if you use the appsettings APIs to read the value of the switch and if the value is false you can disable some functionality of that application. The values in this field are boolean.

Single Select

If you set the Single Select display option, you provide the user with the option to select only one value from a list of given options. When you select your **Display Option** as ***singleselect***, in the **Default Value** column click ... to define multiple values and the default value set is the first value in the first row. Suppose you define multiple views for a segment in the appsettings area, the user can select any one of the listed views using the settings. Using the appsetting APIs, the value the user selects can be assigned to *frm.segment.view=""* and the view can be changed at runtime. The values in this field are stored as string. The figure below depicts how single select is viewed on application settings.



Multi Select

If you set the Multi select display option, you provide the user to select multiple values from a list of given options. When you select your Display Option as *multiselect*, in the **Default Value** column click ... to define multiple values and the default value set is the first value in the first row. Suppose the user sets multiple values for locations, the locations set are returned as a JS object and can be assigned to any widget using the application settings APIs. The figure below depicts how *multiselect* is rendered on iPhone.

Configure Splash Screens

A splash screen is the initial screen that appears when you launch an application.

This topic covers the following procedures:

Configure the Splash Screen for the Mobile Channel

Configure the Splash Screen for the Tablet Channel

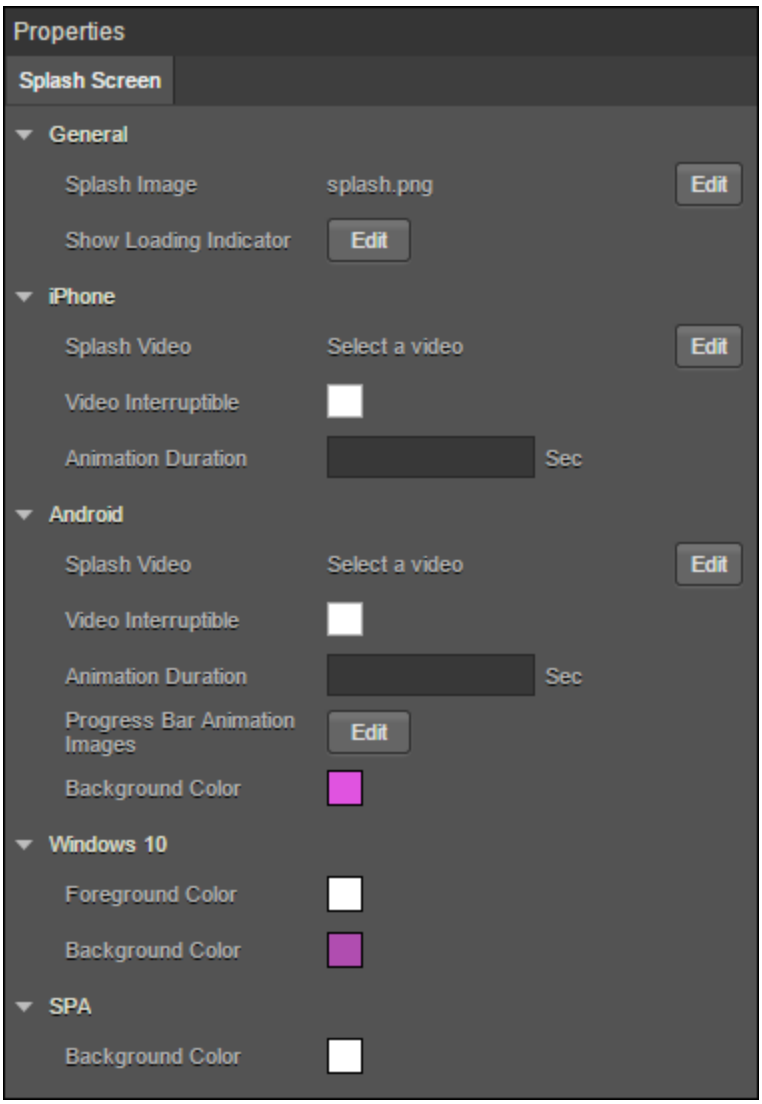
Configure the Splash Screen for the Desktop Channel

Clear the Splash Screen of a Channel

Configure the Splash Screen for the Mobile Channel

To configure the splash screen for the mobile channel, do the following:

1. In the Project Explorer, on the **Properties** tab, expand the Mobile channel, and then click **Splash Screen**. Its properties display in the Properties pane.



Note: If you want to enable a Native iOS experience for the splash image in iOS devices, select the **useLaunchScreenStoryBoard** option. When this property is enabled, Xcode's Storyboard tool is used to display the app's launch screen instead of the splash screen. You must add scale-specific images that are to be displayed as the launch image in the following format for iPhones:
"splashiphone@1x.png"
"splashiphone@2x.png"
"splashiphone@3x.png"

2. In the General section, click **Edit** to select a Splash Image. The Splash Image dialog appears.
    - If you want a default splash image for all the platforms, select default option under platform and assign a desired image.
    - If you want to set image for selected platforms, you need to select required platform and an image respectively.

    > ***Note:*** Splash screen Image names should not have spaces, uppercase letters, special characters. Supported formats for the image are PNG, JPEG, and GIF.

3. In the General section, click **Edit** to configure a Loading Indicator. This property is applicable only for iPhone, Android, Windows platform.
    - The Show Loading Indicator dialog appears in which you need to select required platform and set value to true in order to show loading indicator.
    - If you want to set same value for all platforms, select default option under platform and set its value to true or false.

4. Click **Edit** to select Splash Video. Choose Splash Video dialog appears in which you need to select a video (Make sure the video is placed in common folder under Resources). This property is applicable only for iPhone and Android platforms.

    > ***Note:*** Splash video occupies the time between start of the application till the initialization of the application. Once the application gets initialized, it ends the video (even if it is not complete) and launches the application. The preferable video formats are MP4 and M4V.

    > ***Note:*** For iOS apps, to mix the audio of the splash video with other music apps (running in the background) you must add a new **mix audio** attribute in the **infoplist_configuration.json** file of the app in the **resources** file of the app project.
    >
    > `"mixAudio":true`

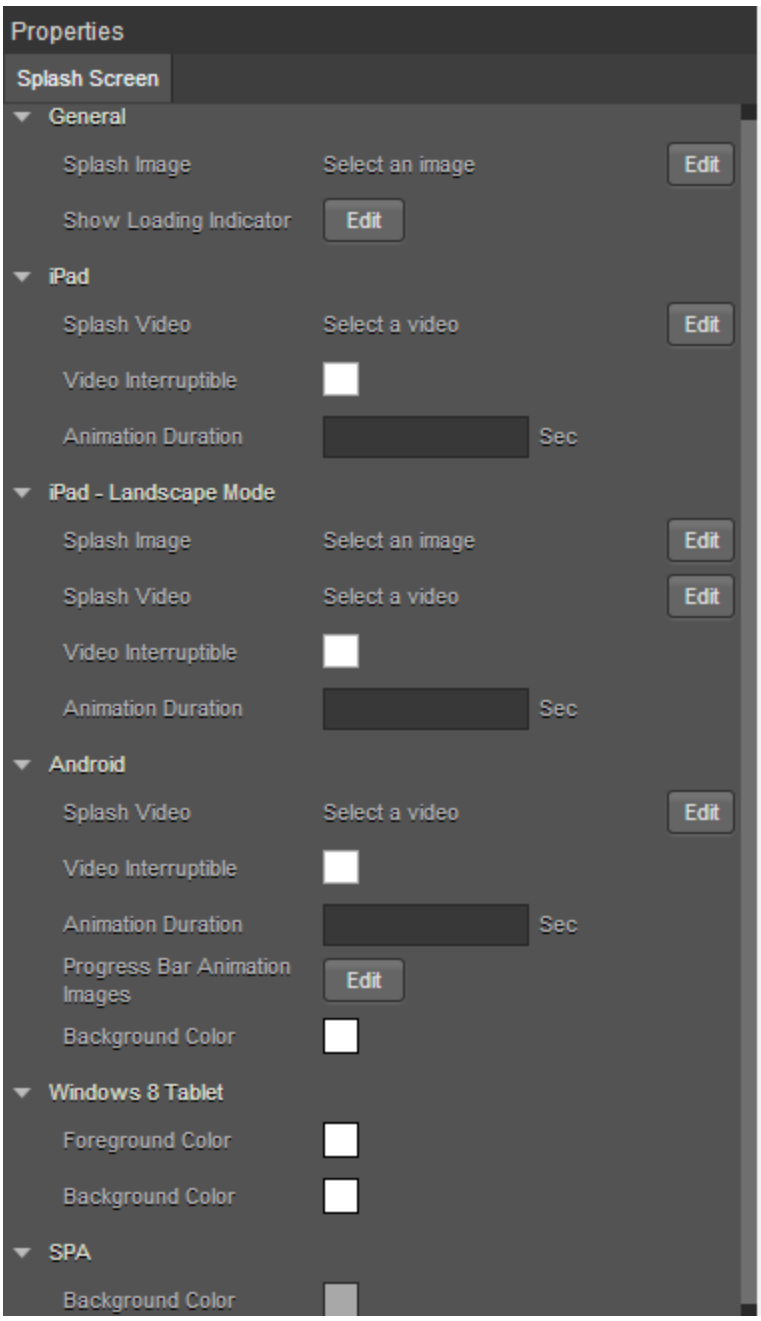
5. Select if the video can be interrupted using Video Interpretable check box.

6. Specify the Animation Duration in seconds. Each progress bar image you specify appears on the screen for the specified number of seconds.
7. Specify the Progress Bar Animation Images by clicking the **Edit** button. Browse and select to add each progress bar animation image. This property is applicable only for Android.
8. Select the Background Color
    - Under Android and Windows, the Color picker dialog appears in which you need to choose a desired background color.
    - Under SPA, the Color picker dialog appears in which you need to choose a desired background color.

Configure the Splash Screen for the Tablet Channel

To configure the splash screen for the tablet channel, do the following:

1. In the Project Explorer, on the **Properties** tab, expand the Tablet channel, and then click **Splash Screen**. Its properties display in the Properties editor.



> ***Note:*** If you want to enable a Native iOS experience for the splash image in iOS devices, select the **useLaunchScreenStoryBoard** option. When this property is enabled, Xcode's Storyboard tool is used to display the app's launch screen instead of the splash screen. You must add scale-specific images that are to be displayed as the launch image in the following format for iPads:
> "splashipad@1x.png"
> "splashipad@2x.png"

2. In the General section, click **Edit** to select a Splash Image. The Splash Image dialog appears.
    - If you want a default splash image for all the platforms, select default option under platform and assign a desired image.
    - If you want to set image for selected platforms, you need to select required platform and an image respectively.
    - You need to set image for iPad-Landscape Mode separately.

    Note: Splash screen Image names should not have spaces, uppercase letters, special characters. Supported formats for the image are PNG, JPEG, and GIF.
3. In the General section, click **Edit** to configure a Loading Indicator. This property is applicable only for iPad, Android, Windows platforms.
    - The Show Loading Indicator dialog appears in which you need to select required platform and set value to true in order to show loading indicator.
    - If you want same value for all platforms, select default option under platform and set its value to true or false.
4. Click **Edit** to select Splash Video. Choose Splash Video dialog appears in which you need to select a video (Make sure the video is placed in common folder under Resources). This property is applicable only for iPad and Android platforms.

Note: Splash video occupies the time between start of the application till the initialization of the application. Once the application gets initialized, it ends the video (even if it is not complete) and launches the application. The preferable video formats are MP4 and M4V.

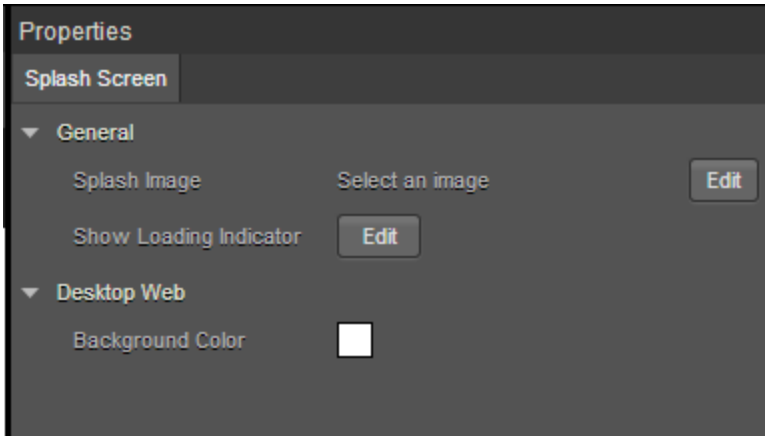
5. Select if the video can be interrupted using Video Interpretable check box.
6. Specify the Animation Duration in seconds. Each progress bar image you specify, appears on the screen for the specified number of seconds.
7. Specify the Progress Bar Animation Images by clicking the **Edit** button. Browse and select to add each progress bar animation image. This property is applicable only for Android.
8. Select the Background Color.
    - Under Android, the color picker dialog appears, in which you need to choose a desired background color.
    - Under SPA, the color picker dialog appears in which you need to choose a desired background color.

Configure the Splash Screen for the Desktop Channel

To configure the splash screen for the Desktop channel, do the following:

1. On the Project Explorer, on the **Properties** tab, expand the Desktop channel, and then click **Splash Screen**.
2. In the General section, click the **Edit** button that corresponds with **Select an image**. The Splash Image dialog displays.
3. appears in which you need to select Desktop Web platform and an image.

    Note:Splash screen Image names should not have spaces, uppercase letters, special characters. Supported format for the image is PNG. Supported formats for the image are PNG, JPEG, and GIF.



4. In the General section, click **Edit** to configure a Loading Indicator. The Show Loading Indicator dialog appears in which you need to select Desktop Web platform and set value to true in order to enable this property.
5. In Desktop Web section, select Background Color. The color picker dialog appears in which you need to choose a desired background color.

Clear the Splash Screen of a Channel

You can clear the splash screen configuration so that you can either restart the configuration or do without a splash screen.

To clear the splash screen configuration of a channel, do the following:

1. In the Project Explorer, on the **Project** tab, expand the channel for which you want to clear the splash screen.
2. Click **Splash Screen**.
3. In the Properties Editor, on the **Splash Screen** tab, click the **Edit** button associated with the Splash Image property.
4. Ensure that the check box for the platform you want to clear the splash screen from is checked, then click that platform's Value cell.

5. Click once any image file that is listed so that it is selected in the list, then click it again so that no image in the list is selected.
6. Click **OK** to close the Select Image dialog box.
7. Repeat steps 4 through 6 for any other platforms that you want to clear the splash screen from.
8. Click **OK** to close the Splash Image dialog box.

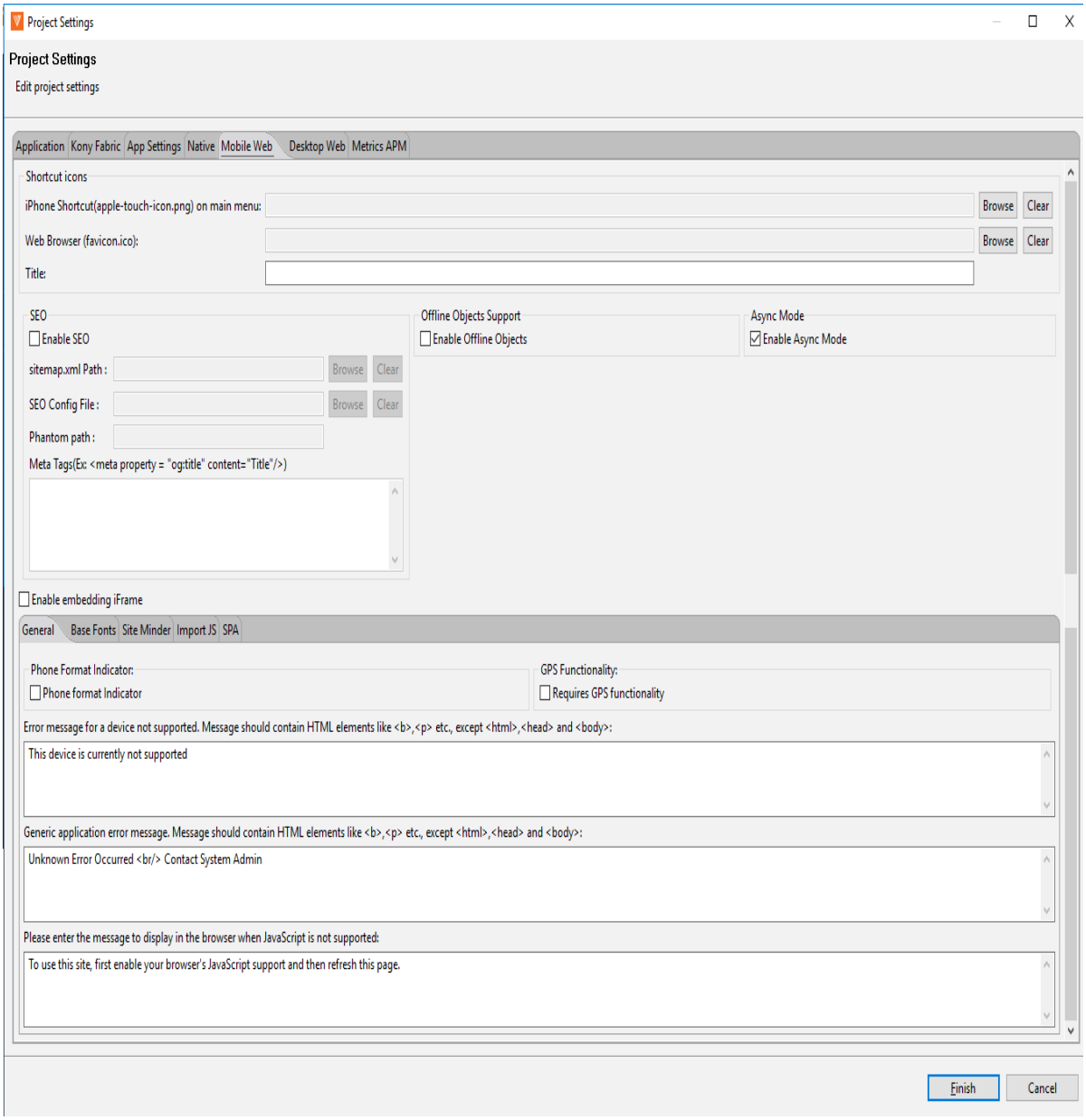
- In the Project Explorer, on the **Project** tab, click the context menu arrow for the desired channel (Mobile, Tablet, or Desktop), and then click **Clear Splash Screen**.

Mobile Web Properties

Mobile Web is the browser on the device. Mobile Web properties define the properties for the application on Mobile Web under various platforms.

To set Mobile Web properties, do the following:

1. On the **Edit** menu, click **Settings**, and then click the **Mobile Web** tab.

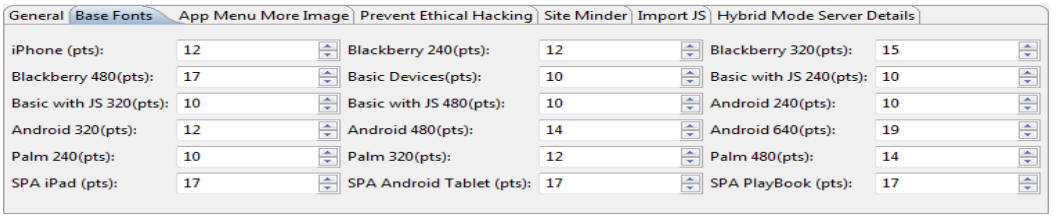


2. In the Shortcut icons section, click the **Browse** button that corresponds with the iPhone Shortcut property, and then navigate to and select the file you want to use as the icon that represents the app on the iPhone screen.

3. In the Shortcut icons section, click the **Browse** button that corresponds with the Web Browser property, and then navigate to and select the file you want to use. When you launch the application in a web browser you see the icon specified here as the application icon.
4. In the Title text box, enter the title that you want to appear in the in the web browser when you launch the application on Mobile Web.
5. Navigate to the **General** tab.
    i. Select the appropriate **Session Manager** in **Session Management**. Session management indicates how session data (for example, user name and password) is managed. *Kony Session Manager* indicates that the data is stored on the Kony Application Server and *Http Session Manager* indicates that the data is stored as part of the session object.
    ii. Select **Phone format Indicator** if you want to highlight a telephone number clearly in the browser.
    iii. Select **Requires GPS functionality** to enable the application to use the GPS functionality. For more information about GPS functionality, see the appendix GPS Functionality.
    iv. Select **Enable focus skin for HTML5 iPhone**, to enable focus skins for HTML5 iPhone.
    v. Select **Enable Server side Mobile Web in Build Screen**, to build applications for the Server side Mobile Web platform.
    vi. Under **No Java Script message**, enter a message to be displayed to the user if the browser does not support Java Script. the default message is *To use this site, first enable your browser's JavaScript support and then refresh this page.* If you want to display a different message, overwrite this message.
6. Navigate to **Base Fonts** tab.

i. Select appropriate **Base Font** sizes in points for **iPhone** and various categories of browsers.

> ***Note:*** When you specify the font size within the skin of a widget, the font size is calculated based on this **Base Font**.

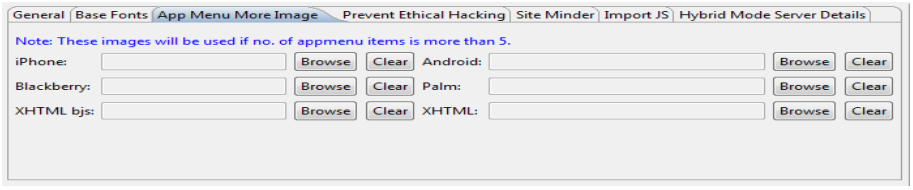


The following table explains the base fonts for each platform:

Platform	Base Font (in points)
iPhone	12
Basic Devices	10
Basic with JS 240	10
Basic with JS 320	10
Basic with JS 480	10
Android 240	10
Android 320	10
Android 480	14

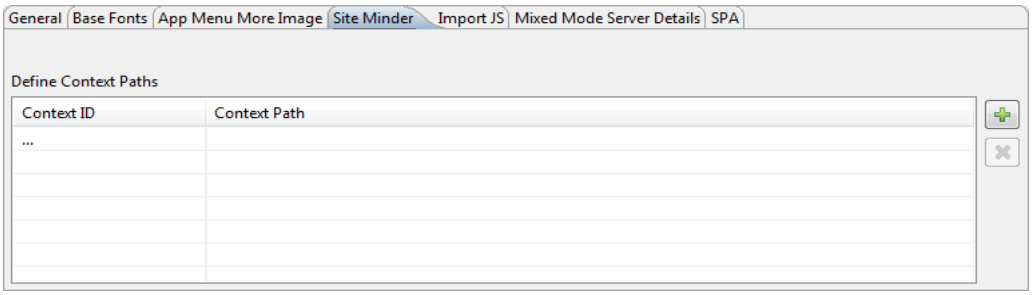
Platform	Base Font (in points)
Android 640	19
SPA iPad	17
SPA Android Tablet	17

7. Navigate to **App Menu More Image** tab.
    i. On platforms like iPhone, Android, XHTML BJS, and XHTML *More* button appears when the app menu has more than 5 app menu items. You can specify an image for the *More* button on these platforms. Browse and select the image if you an image to displayed instead of the standard *More* button.

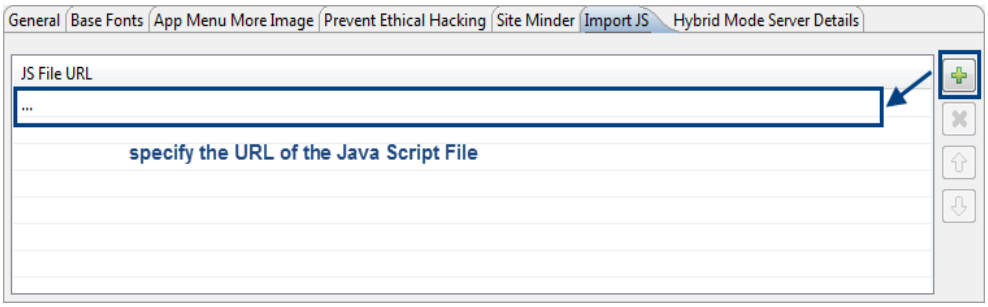


> ***Note:*** If you do not specify an image, the standard *More* button is displayed.

8. Navigate to **Site Minder** tab.
    i. Add a **Context ID**. Context ID is used to set the context path in the URL of an application.
    ii. Add a corresponding **Context Path**. Context Path appears in the application URL.



9. Navigate to **Import JS** tab.
    i. Click the **Add** button to add a row where you can specify the URL of the Java Script file that you want to be executed in the browser.



    ii. Repeat the above step if you want add more Java Script files.

10. Navigate to **Mixed Mode Server Details** tab. Provide the details of the server that must be used for a Mixed Mode application. Use this option to set default image while downloading the image if you do not have any image for image while loading property.

11. You can specify a default downloading image under the **SPA** tab which is displayed to the user instead of a blank page.

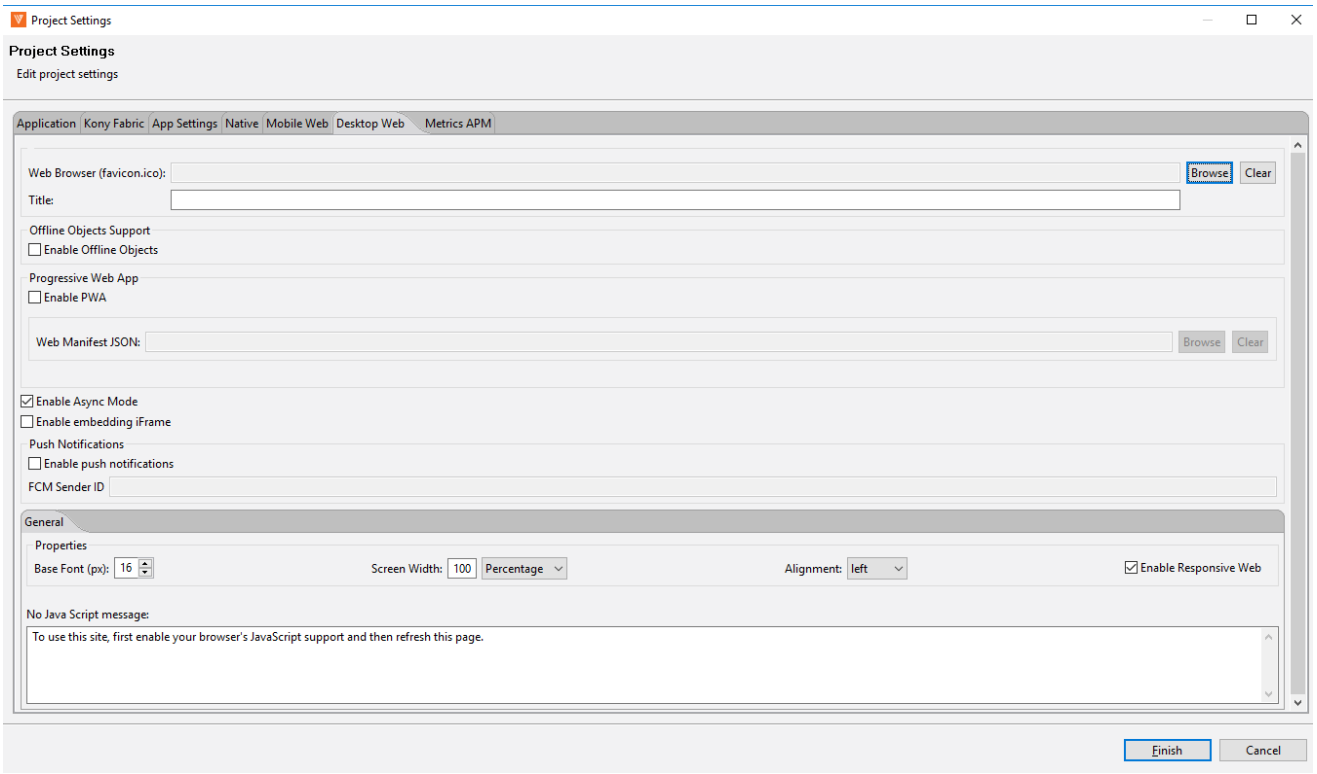
12. Click **Finish**.

Desktop Web Properties

Desktop Web is the browser of the desktop. Desktop Web properties specify the properties for the application on Desktop Web under various platforms.

To set Desktop Web properties, do the following:

1. On the **Edit** menu, click **Settings**, and then click the **Desktop Web** tab.



1. Browse and select the appropriate **fav icons** for *Desktop Web Browser*. When you launch the application on the Desktop Web Browser you see the icon specified here as the application icon.
2. Enter a **Title** for the application. This title appears on the web browser when you launch the application on Mobile Web.
3. If you want to enable support for offline objects for your desktop web application, in the **Offline Objects Support** section, select **Enable Offline Objects**.

4. If you want to build a Progressive Web App, in the **Progressive Web App** section, select **Enable PWA**.
   Once you select the option, the Web Manifest JSON option is enabled.

5. From Web Manifest JSON , select Browse. The file explorer opens.

6. Navigate to the folder where you have the JSON file and select your desktopweb manifest file. The Web Manifest JSON file contains information on the resources the Progressive Web App required. The information can be name of the app, app icons, etc. Click here for a sample.

7. Using the **Enable Embedding iFrame** feature, you can choose the SPA/Desktopweb application behavior in a subwindow (for example, iFrame). The default setting for this feature is not selected. To allow an application to open in a subwindow, select **Enable Embedding iFrame**. The application can be launched in any other main windows application.

   To restrict the application from opening in a subwindow, leave the default setting. The application will not open in any other main windows application.

8. Navigate to the **General** tab.

9. From the **Base Fonts** list, select the appropriate base font size in pixels.

   i. Select appropriate **Base Font** sizes in points for **iPhone** and various categories of browsers.

   > ***Note:*** When you specify the font size within the skin of a widget, the font size is calculated based on this **Base Font**.

10. In the Screen Width field, specify the width the application occupies in the Desktop Web browser. The screen width value can be percentage based or pixel based.

11. From the **Alignment** list, select how the application is aligned in the browser. The possible values are center, left, and right.

12. Select the check box of **Enable Responsive Web** if you want to enable Responsive Web design for your desktop application.

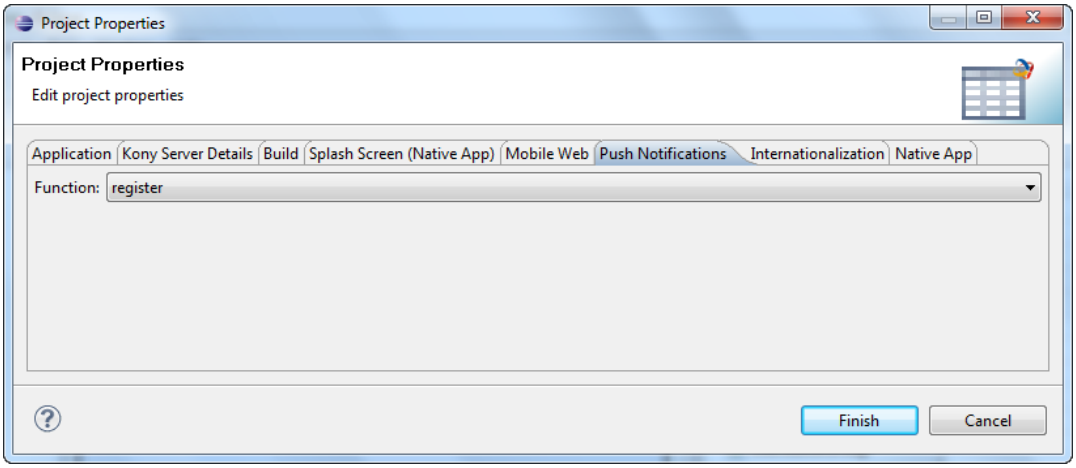
> ***Important:*** If you have enabled Progressive Web App in Project Settings, when you build a Responsive Web application, the build creates a new output in the Progressive Web App format. For more information about Responsive Web apps, click here. For more information about Progressive Web Apps, click here.

Push Properties

This tab allows you to specify a user-defined function that registers the application for Push Notifications.

To set push notification properties, do the following:

1. Go to Project Properties window.
2. Click **Push Notifications** tab.



3. Select an appropriate function that registers the application for Push Notifications and enables push notifications.
4. Click **Finish**.

Set Native App Properties

Native app properties are divided into two categories: those that are common to all platforms, and those that are platform-specific. These properties range from the logo image your app displays to the types of screens and SDKs the app supports, and how certificates are handled.

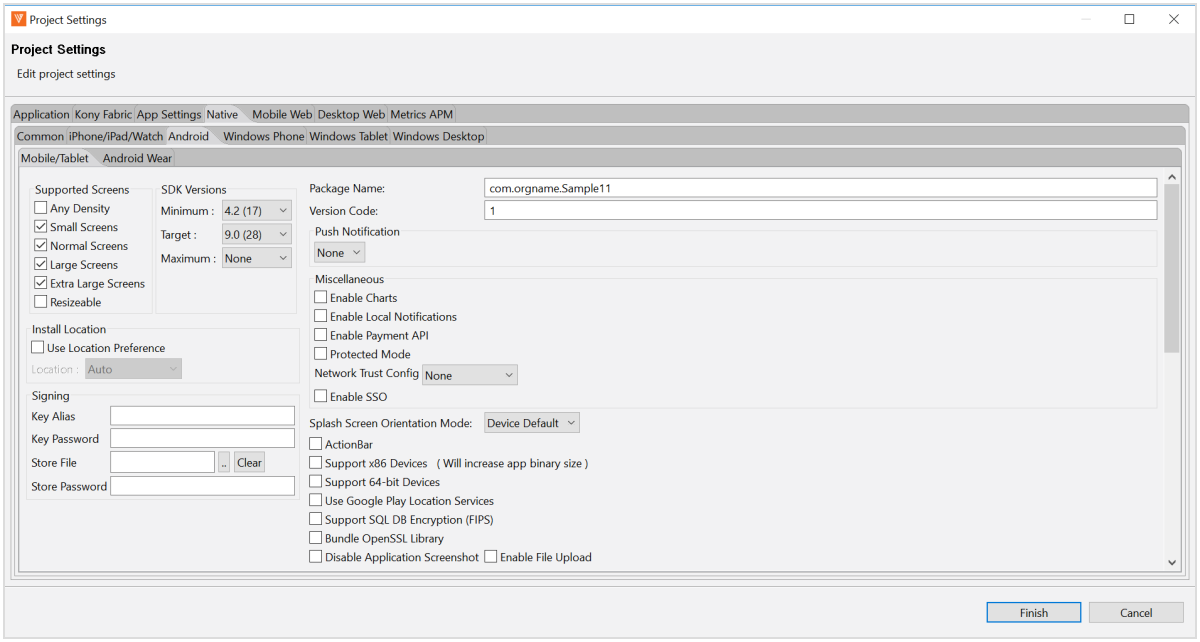
From Kony Visualizer V8 SP4, you can also enable certain Android features by manually adding the corresponding properties to the androidbuild.properties file.

To set Native app properties, follow these steps:

1. In Kony Visualizer, from the Project Explorer, click **Project Settings**. The Project Settings window appears.
2. Click the **Native** tab. A list of sub-tabs appears.
3. Click the **Common** sub-tab, and set the following properties:
    a. Enter a **Name** for the Native channel version of the application. If no name is specified, the Application ID specified under **Application Properties** is used.
    b. Browse and select an image file for the logo.

> *Important:*
> - The logo you specify here will be renamed to `icon.png` while building the application on the iPhone platform.
> - For Desktop, the icon dimension must be in multiples of 8. Minimum pixels can be 8 x 8 and maximum pixels can be 248 x 248.

4. Click the **Android** sub-tab, and set the following properties:



a. In the **Supported Screens** section:

- **Any Density** - If this option is enabled, the application can accommodate any screen density for a resource. You do not have to use this option if your app directly manipulates bitmaps. Generally this option must always be set to true.
- **Small Screens** - If this option is enabled, the application supports smaller screen form-factors. A small screen is one with a smaller aspect ratio than the normal (traditional HVGA) screen. An application that does not support small screens is not available for small screen devices from external services (such as Google Play).
- **Normal Screens** - If this option is enabled, the application supports normal screen form-factors. Traditionally, this is an HVGA medium density screen, but WQVGA low density and WVGA high density are also considered to be normal.
- **Large Screens** - If this option is enabled, the application supports larger screen form-factors. A large screen is defined as a screen that is significantly larger than a normal handset screen, and might require some special care on the application's

part to make good use of it. The application may rely on resizing by the system to fill the screen.

If this option is unchecked, it enables screen compatibility mode.

- **Extra Large Screens** - If this option is enabled, the application supports extra large screen form-factors. An extra large screen is defined as a screen that is significantly larger than a large screen, for example, tablet (or something larger) and may require special care on the application's part to make good use of it. The application may rely on resizing by the system to fill the screen. If the option is unchecked, this will generally enable screen compatibility mode.
- **Resizable** - If this option is enabled, the application is resizable for different screen sizes. This property is deprecated by Android SDK and is supported only for customers on the 2.6 plug-in. This property enables you to run an application in the compatibility mode. For more information, see Support Screen Elements for the Android App Manifest.

b. In the **SDK Versions** section:

- Select the **Minimum** SDK Version that needs to be supported for the application. The default minimum SDK value is 4.0.

> ***Notes:***
> - Kony Visualizer does not support SDK Versions less than 4.0.
> - You must keep the minimum SDK value between 4.0 and 4.4.
> - The SDK values of 5.0 and above results in a build error(technical limitation).
> - The application must be built with a minimum version matching the device SDK version. For example, a device with 5.0 version of SDK cannot run an application built on 4.0.

- Select the **Target** SDK Version that needs to be supported for the application.

  > ***Note:*** The Target SDK Version must be greater than or equal to the Minimum SDK Version.

- Select the **Maximum** SDK Version that needs to be supported for the application.

c. **Package Name**: Package name is a unique name to identify a specific application. It is generally in the format `domain.company.application`.

> ***Note:*** The name you specify for *Android Package* must contain at least two segments.

A segment is a valid Java package name. The following are a few examples of valid Android Package names:

- `com.konylabs.<ApplicationName>`
- `com.kony.<ApplicationName>`
- `com.konysolutions.<ApplicationName>`
- `com.kony.<ApplicationName>_Android`.

d. **Version Code**. This is an internal version number. This number is used to determine whether the application is a recent version. This version number is not shown to users. The value must be an integer. You can increase each version by one to indicate a newer version.

e. In **Push Notification** section:

- **GCM** - Select this option to enable Push Notifications for the application. This option copies the libraries required for push notification into the project during build time.

> ***Important:*** GCM (Google Cloud Messaging) is supported only for Android SDK Versions 2.3 and above.

- **Custom GCM Broadcast Receiver (Optional)** - If your application requires to override the default GCM broadcast receiver behavior, you can provide your own custom broadcast receiver. To customize the GCM receiver, see Customizing GCM Broadcast Receiver.

- **FCM** Select this option to enable Push Notifications for the application. This option copies the libraries required for push notification into the project during build time.
  - **Custom FCM Service (Optional)** - If your application requires to override the default FCM service, you can provide your own custom FCM service. To customize the FCM service, see Customizing FCM Service.

f. In the **Miscellaneous** section:

- **Protected Mode** - Selecting this option ensures that your app is not run on a rooted/jail-broken device.

> ***Note:*** This option works only if the application is built in Release mode. The Protected Mode option works only if the application is built in Release mode. To know more about protecting your application, see Applying Application Security.

- **Enable Local Notifications** - Select this option to enable notifications scheduled by an app and delivered on the same device. They are suited for the apps with time-based behaviors, such as calendar events.

- **Enable Payment API** - Selecting this option enables online transactions in applications.

- **Network Trust Config** - Using this option, you can control the certificates that are used.

- **None** - No certificates are allowed. This means that if the certificate is present in the Android Trust store, it will allow the N/W call to proceed; otherwise, it throws an exception. With this option, servers having non-trusted or self-signed certificates are not accessible via the app on the device.
- **All** - All types of certificates are allowed regardless of whether they are bundled. This option is useful during the development phase of an app, but not for publication. With this option, all servers are accessible regardless of the kind of certificate they hold (i.e. self-signed, non-trusted, trusted). Due to the lack of security inherent in this option, the Google Play store rejects such apps when they are submitted for publication.
- **Allow Bundled** - Only the certificates that are bundled along with the app are allowed. With this option, the app can communicate **only** with servers that have the certificate(s) that are bundled with the app.
  - To bundle the certificate in the application, copy the certificate under the following folder: For mobile, `<workspace>/<app>/resources/mobile/native/android/assets/certs`. For tablet, `<workspace>/<app>/resources/tablet/native/androidtab/assets/certs`

    > ***Note:*** If an *assets* folder does not exist, create an *assets* folder under respective locations as indicated above.
    > Create *certs* directory under the *assets* folder and add all the certificates into this folder.

> ***Important:*** Allow Bundled option will not work in Android 2.3.x OS versions due to certificate chaining issue (causes certificate exception). This is a known Android native issue. For more information, see Issue 25152 on the Android Open Source Project Issue Tracker. Among other topics, Issue 25152 documents that to make the bundled option work in Android 2.3.x devices, the root CA needs to be omitted from the server end.

- **Allow Pinned** - Only the certificates that are pinned or associated to the host. Pinning makes use of knowledge of the pre-existing relationship between the user and an organization or service to make the security-related decisions better.

g. In the **Install Location** section:

- **Use Location Preference**. This property defines the location where the application is deployed.
  - **Auto** - Indicates that the application is deployed on the device and can be moved to the SD Card later if required.
  - **Prefer SD Card** - Implies that the application is deployed on the storage card and cannot later be moved to the device memory.

> ***Note:*** This works only if the Minimum SDK is 2.3 or above.

h. **Signing** - Use this option to sign the android binary automatically during the build process.

- **Key Alias**- Use this option to enter the alias of the key.
- **Key Password** - Use this option to enter the password for the key.
- **Store File** - Use this option to locate and configure the store file.
- **Store Password** - Use this option to enter the password for the store.

i. **Splash Screen Orientation Mode** - When resource folders are created from IDE by **Add resource folders** option, then directories like drawable-port and drawable-land are created automatically inside the directory `<workspace>/<app>/resources/mobile/native/android`.

- **Portrait** - Use this option if splash screen support is required only for portrait mode, and copy it inside drawable-port folder.
- **Landscape** - Use this option if splash screen support is required only for landscape mode, and copy it inside drawable-land folder.
- **Both** - Use this option if splash screen support is required for both modes. If splash screen images are different images, then place the image in their respective directories. If you use same image for both modes, then copy the resources under `mobile > native > android` instead of drawable-port or drawable-land folder.

j. **ActionBar** - Enabled only if target SDK is 3.0 or above. Use this option to enable *Action Bar* feature.

k. **Support x86 Devices** - Select this option to support any Android-x86 devices.

l. **Support 32-bit Devices** - Select this option to build an Android APK with 32-bit support. Once you select this option, only 32-bit `.so` files (armeabi-v7a and x86) are packed and the application leaves out 64-bit `.so` files (arm64-v8a and x86_64). If you do not select the Support 32-bit devices option, 64-bit libraries / `.so` files (arm64-v8a and x86_64) are packed by default.

> ***Note:*** From Kony Visualizer V9 onwards, 64-bit APK's are generated, by default.

> ***Note:*** From August 1st 2019 onwards, all apps published on Google Play must support 64-bit architectures.

To support both 64-bit and 32-bit architectures in Google Play store, you must make sure that you perform the following actions:

- Build the 32-bit and 64-bit APKs with two different version codes, which are separated by at least 1000.
  For example, `64-bit version code = 32-bit version code + 1000`.
- Ensure that the third-party libraries (`AAR` files) contain the respective `.so` files in all supported architectures: lib/armeabi-v7a, lib/arm64-v8a, and lib/x86 lib/x86_64 for 32-bit and 64-bit architectures, respectively.

> ***Note:*** To enable your application to be built with both 32-bit and 64-bit native libraries, refer to Support for 32-bit and 64-bit Architectures in a Single APK.

m. **Support SQL DB Encryption (FIPS)**- In Android, if you select this option, Kony Visualizer automatically bundles Federal Information Processing Standard (FIPS) compliant SQL Cipher third-party library with the application. After the application is complied with this option selected, the APIs in Web SQL support database encryption. For more info, see **API Reference Guide > Offline Data Access APIs > Web SQL APIs.**

> ***Note:*** FIPS does not provide the mechanism to build the x86_64 architecture for Android. From V8 SP4 onwards, for the x86_64 architecture, Kony Visualizer automatically bundles libraries that are non-compliant to FIPS for the following items:
> SQLCipher
> OpenSSL

n. **Bundle OpenSSL Library**- In Android, if you select Bundle OpenSSL Library option, Kony Visualizer automatically bundles a third-party OpenSSL native library along with the application. The following APIs use this OpenSSL library to support additional hashing algorithms than the algorithms supported by Native Android SDK (Java Implementation). For information on supported algorithms in these APIs, see **API Reference Guide**

> Cryptography APIs.

i. kony.crypto.createPBKDF2Key

ii. kony.crypto.createHMacHash

> ***Note:*** FIPS does not provide the mechanism to build the x86_64 architecture for Android. From V8 SP4 onwards, for the x86_64 architecture, Kony Visualizer automatically bundles libraries that are non-compliant to FIPS for the following items:
> SQLCipher
> OpenSSL

o. **Disable Application Screenshot** - This option specifies whether the user can take a screenshot of your application.

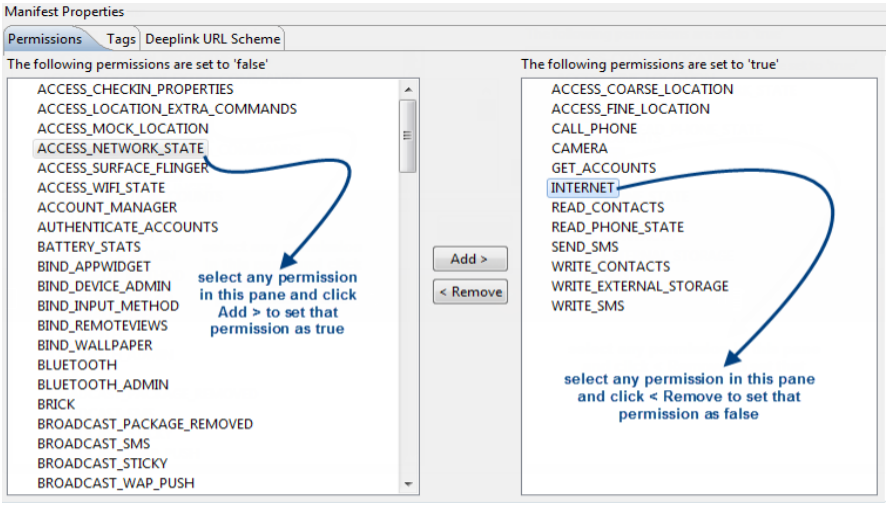
p. **Enable File Upload** - Enables you to upload files to a remote sever by using the HttpRequest API.

q. In **Manifest Properties** section:

- **Permissions** tab: Set the permissions to **true** or **false** based on the application requirements. Set the appropriate permissions for Android Manifest file. For more information, see The Android Manifest File.

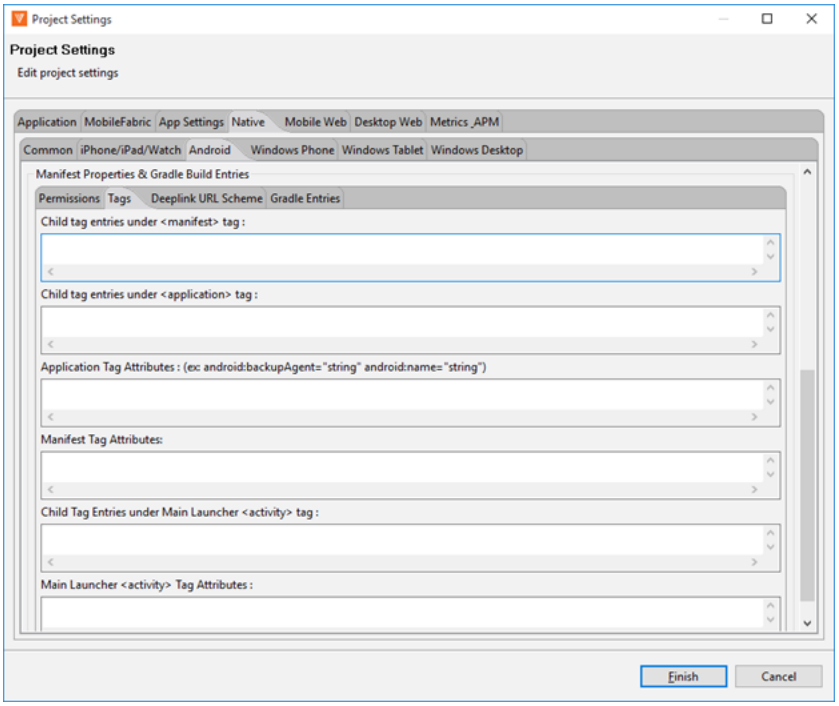
  i. To enable permissions, select the permissions from the left pane and click **Add >**.

  ii. To disable permissions, select the permissions from the right pane and click **< Remove**.



> ***Important:*** Add the `WRITE_EXTERNAL_STORAGE` setting if you need to save images in an external storage like SD Card.

- **Tags** tab: You can add tags to the Android manifest file directly from Kony Visualizer by specifying tag entries and attributes on the Tags tab. You can specify child tag entries and attributes for <manifest> and <application> tags, and the Main Launcher <activity> tag. For more information on the tags you can add with the manifest or application tags, see http://developer.android.com/guide/topics/manifest/manifest-intro.html.



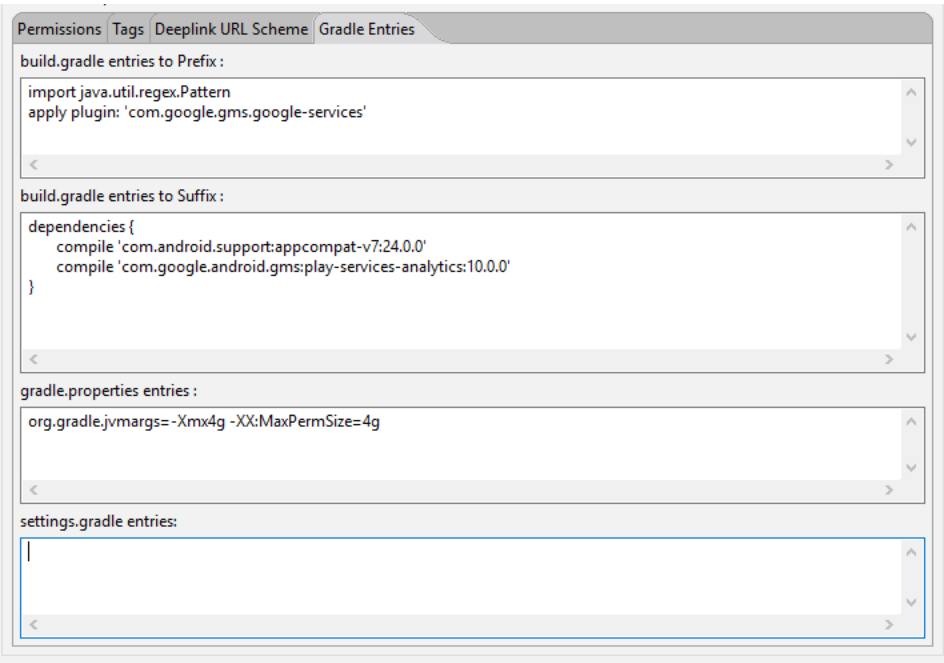
For more information on the tags you can add, see http://developer.android.com/guide/topics/manifest/manifest-intro.html.

- **Deeplink URL Scheme** tab: You can use the URL Scheme tab for Deeplinking. The values for Scheme/ port/ path/host/pathprefix/path pattern specified under this tab can be used to deep-link to a particular URL directly. For instance, if you set the masterdata for a browser widget to be a URL to deep-link, use the following format, `scheme://host:port/path`*or*`pathPrefix`*or*`pathPattern`. For more information on each of the values available under the URL Scheme tab, see http://developer.android.com/guide/topics/manifest/data-element.html#path.

  > ***Important:*** For a URL scheme in Android, please note that the scheme name should be in lowercase; otherwise the scheme name will not work in higher versions (Android 4.0 and above) or Android devices.

- **Gradle Entries** tab: You can use the Gradle Entries tab to import additional gradle packages, apply external plugins, or specify build-related configuration information,

build dependencies, or the location of any external repositories or modules used by your Android application.



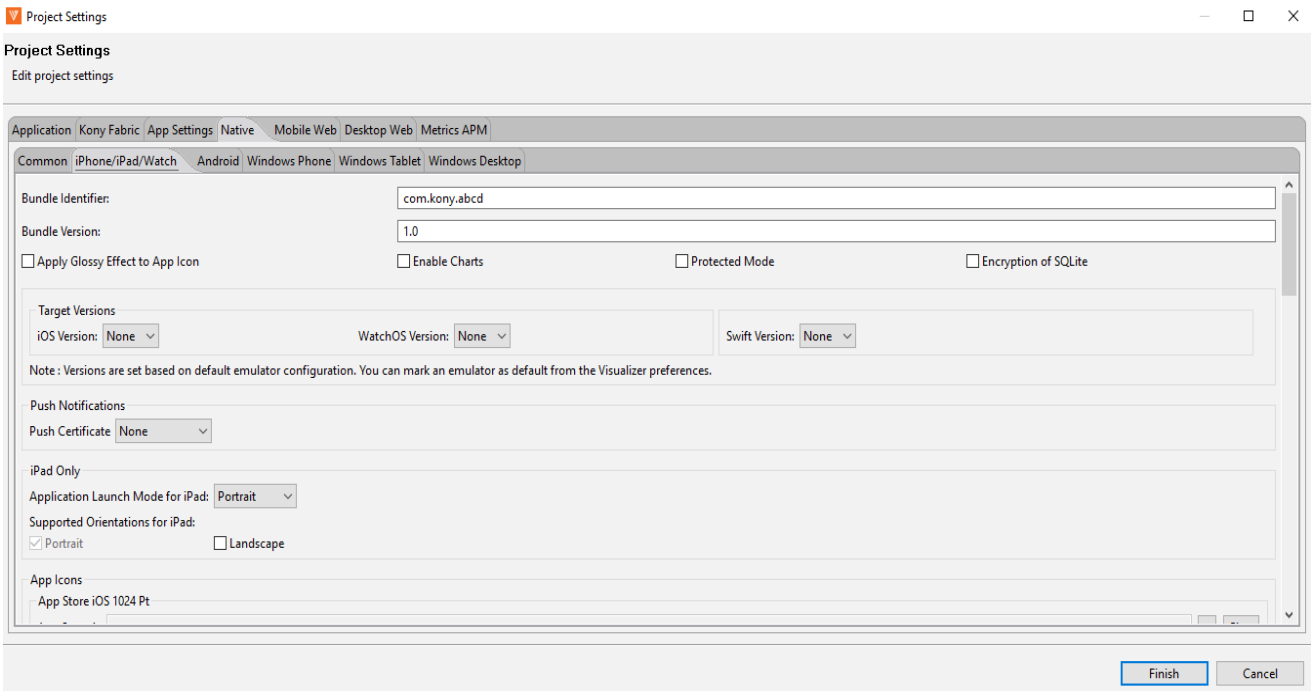
You can specify build.gradle entries as a prefix or suffix entries:

- Prefix entries are added just below any existing import statements in the build.gradle build script file. Use prefix entries to import additional gradle packages, or to specify external plugins to use in the build.
- Suffix entries are appended to the end of the generated build.gradle file. Use suffix entries to customize build logic; for example, to add compilation dependencies such as Google and Android support repositories, local library modules, or local and remote repository paths. For more information, see Organizing Build Logic.

Use the gradle.properties entry to configure project-wide Gradle settings, such as the Gradle daemon's maximum heap size or proxy settings. For more information, see The Build Environment.

Use the settings.gradle entry to specify external modules (Gradle-based third-party Android libraries) to include when building your application. For more information, see Configure Your Build.

5. Under **iPhone/iPad/Watch** tab, set the following properties:



a. **Bundle Identifier** - Provide an unique name that identifies the application bundle. This is usually in three parts and follows the convention of `com.kony.<appname>`.

b. **Bundle Version** - a number that identifies the version of the application bundle.

c. **Apply Glossy Effect to App Icon** - specifies if the glossy effect must be applied to the app icon. The default value is *false*.

d. **Enable Watch** - Select this option to enable wearable functionality in your application. If you select this option, wearable binaries are bundled with your application.

e. **Protected Mode** - Selecting this option ensures that your app is not run on a rooted/jail

broken device. To use this option, you have to patch Xcode with the Finalizer utility. For information on patching Xcode with Finalizer, see the Install Finalizer Package. This option works only if the application is built in Release mode. The Debug mode will generate unprotected binaries.

> ***Note:*** The Protected Mode option works only if the application is built in Release mode. To know more about protecting your application, refer Protect your app topic.

f. **Push Certificate** - Choose the required option from the list to either enable push notifications in different types of environments. You can also disable receiving push notifications for any environment. This field contains the following options:

- **development**: Select this option to receive push notifications when in the developer environment.
- **production**: Select this option to receive push notifications when in the production environment.
- **None**: Select this option to disable push notifications in any environment.

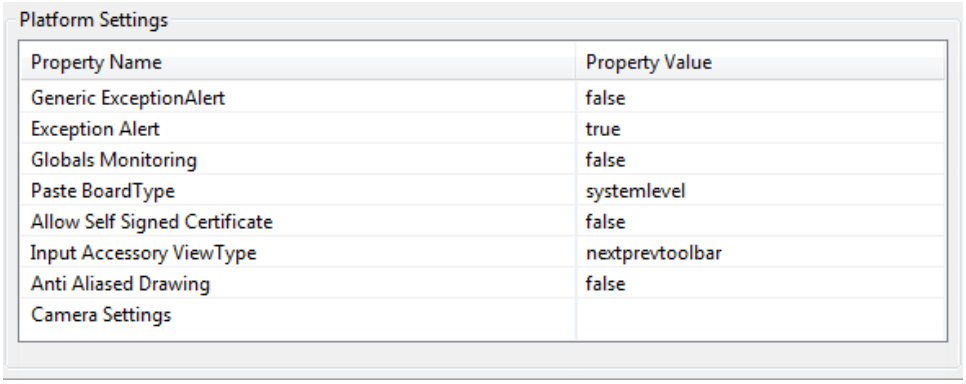
> ***Note:*** This feature is available from Kony Visualizer V8 SP4 FP19 onwards, and is available in both [[[Undefined variable MyVariables.ProdNameVizEnterprise]]] and [[[Undefined variable MyVariables.Kony QuantumViz]]].

g. **Application Launch Mode for iPad** - specifies the default mode of launching the application on iPad. *Portrait* is the default value.

h. **Supported Orientations for iPad** - specifies the supported orientations for the iPad. This depends on the launch mode. The different orientations for a form and at application level are listed at

i. **Deeplink URL Scheme**: specifies a url to which the application will deep-link to. If the application name is southwest then the url scheme that the other applications can use to launch the southwest application is *southwest://*. For more information about deep-linking, see Appendix E: the App Service Event.

- **Platform Settings**: Using the Platform Settings Area, you can set certain default properties for an application for iPhone.


Platform Settings

Property Name	Property Value
Generic ExceptionAlert	false
Exception Alert	true
Globals Monitoring	false
Paste BoardType	systemlevel
Allow Self Signed Certificate	false
Input Accessory ViewType	nextprevtoolbar
Anti Aliased Drawing	false
Camera Settings	

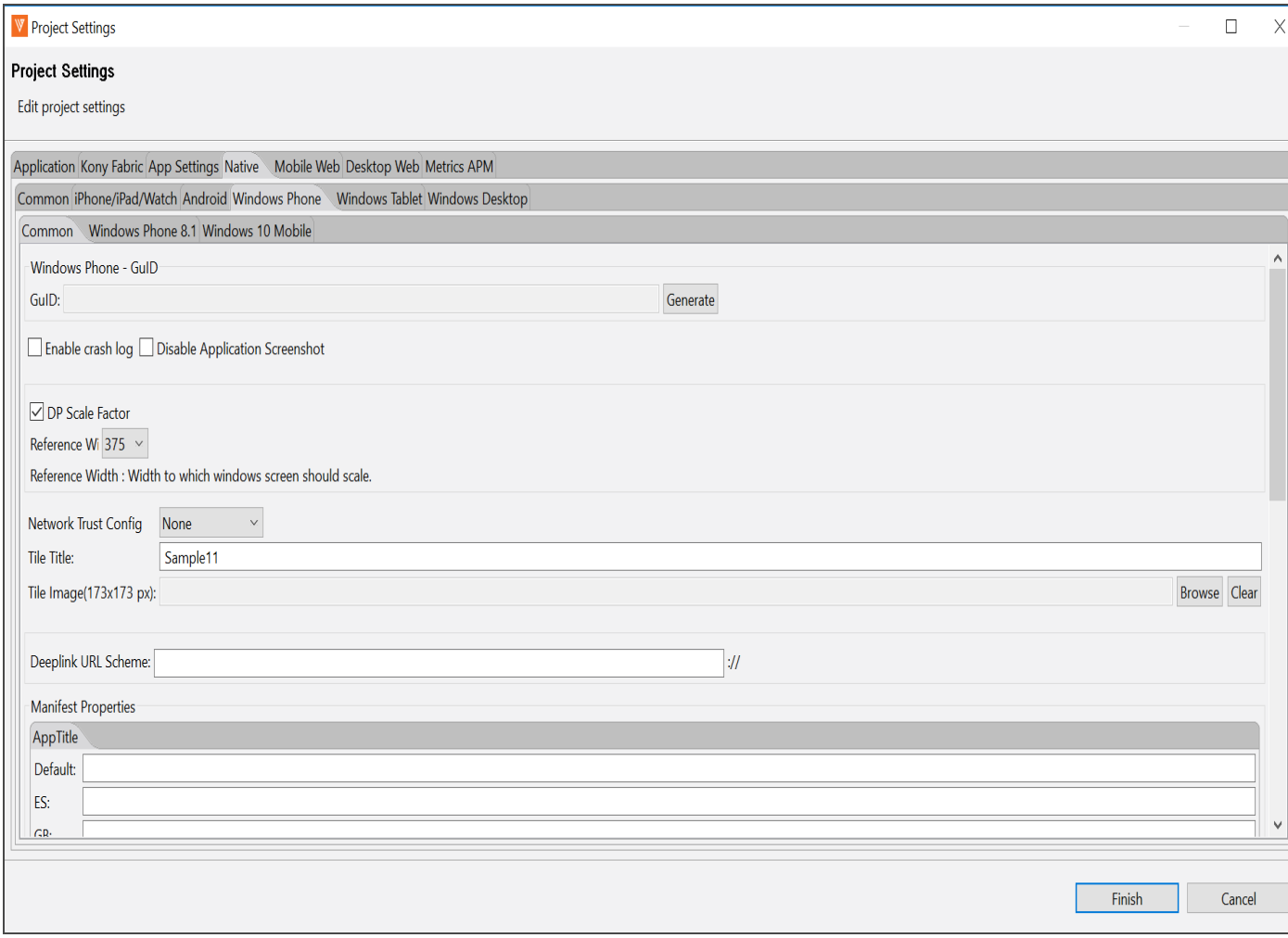
  - Generic exception alert: When true, generic exception alerts like "system error" are fired and when false detailed exception message is shown as alert. Best practice is to be use true for release mode and false for debug mode.
  - Exception alert: When true, system throughs the exception alerts otherwise (with false) app would crash in case of exception instead of alert. Best practice is to be use true for release mode and true for debug mode.
  - Globals monitoring: If set to true, the information like number of variables of the given type used in the app. Possible types are Strings, tables, numbers, closures, forms, other objects is printed in the logs.
  - Paste Board Type: It will allow the user to copy paste content from the app to external writable area like message etc. system level - it will allow to copy paste into other applications. Applevelpersistent - it will allow to copy paste within the app and the messages are persistent will be available across the app restarts. Applevelnonpersistent - it will allow to copy paste within the app and the messages are Not persistent and will not be available across the app restarts. Nopasteboard - it will not allow paste anywhere.
  - Allow Self Signed Certificate: By default it is false, if true, it allows self

signed certificate for development.

> ***Note:*** Self Signed Certificate option is only applicable if you use Network APIs in your application.

- Input Accessory View Type: The input accessory view type for widgets like text box, calendar, grouped widgets etc where you have next previous cancel buttons. This can be overridden by form level Input Accessory View Type.
- Anti Aliased Drawing: If set to true, allows smoother widgets and layout without any jagged edges.
- Camera Settings: Allows to set images to the icons which appear on the camera such as cancel icon, settings icon, tapanywhere.
- Backward_compatibility_mode: By default it is false, if true it will allow the application feature to behave as it would have behaved on earlier version (if there is any behavioral change in the latest version)

6. Click the **Windows Phone** sub-tab, and set the following properties:



   a. Under the **Common** sub-tab:

   b. In the **Capabilities** tab, set the permissions to true or false based on the application's requirements. For more information about each of the permissions refer Capabilities and requirements.

      - To enable permissions, select the permissions that are currently *false* and click **Add >**.

- To disable permissions, select the permissions that are currently *true* and click **< Remove**.

c. In the **Packaging** tab, set the following properties:

- **Package name:** A unique name to identify a specific application. It is generally in the format `domain.company.application`.
- **Package display name:** The name with which the application is submitted to Google play. This name is used to search the application in Google play.
- **Publisher name:** The publisher of the app. This default value is the name of the project. This attribute is required for certain types of apps, such as push-enabled apps.
- **Publisher display name:** Display name of the publisher.
- **Version:** Internal version number of the package.
- **Publisher ID:** Kony is the default Publisher ID of the project.

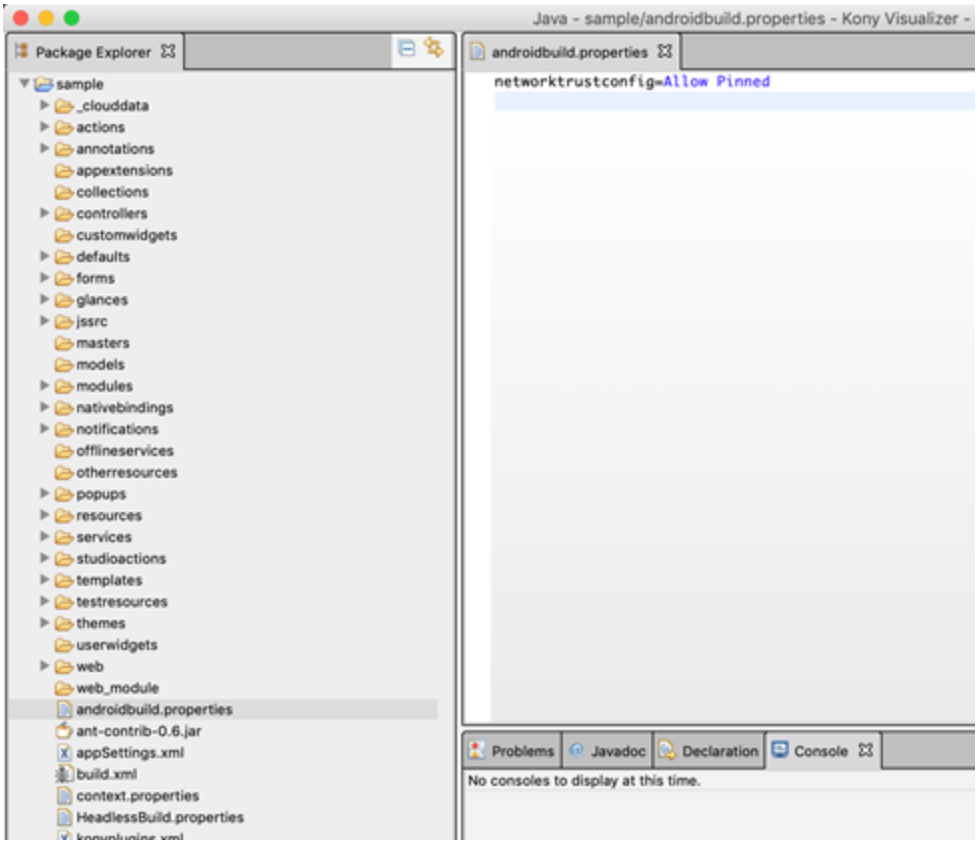
7. Click the **Windows Tablet** and **Windows Desktop** sub-tabs to set their respective properties.

   In Windows Desktop, if you select **Disable Application Minimize** feature, the application will not be allowed to minimize.

8. Click **Finish**.

Add Android Properties to androidbuild.properties File

From V8 SP4 onwards, you can enable or disable certain Android features by adding the corresponding properties to the `androidbuild.properties` file. You must first create the `androidbuild.properties` file in your Kony Visualizer project workspace (**workspaceLocation\\project**) as shown in the image.



> ***Note:*** If the same property is passed from Kony Visualizer in future plugins, the Kony Visualizer property takes precedence.

Enable Android Features

You can enable the following Android features by adding these properties in the androidbuild.properties file:

Public Key Pinning

You can enable the Public Key Pinning feature by specifying the following property in the androidbuild.properties file as shown.

This feature is enabled only if the `networktrustconfig` property is set as `Allow Pinned` in the androidbuild.properties file.

```
networktrustconfig = Allow Pinned
```

React Native

You must set both the following properties in the androidbuild.properties file as shown, to enable React Native feature support:

```
enableReactNative = true
```

Add the enableReactNative flag to enable ReactNative support.

```
reactNativeAppsList = "<reactNativeApp1>,<reactNativeApp2>..."
```

Specify the list of ReactNative apps (root folder name of ReactNative apps that are placed in the ReactNativeProjects folder) to be embedded into your Kony app.

APK Tamper Protection

The APK Tamper Protection feature helps you to verify if an APK has been tampered with (modified from its original version). If a tamper is detected, the application safely exits during the bootup process. This is an optional feature that is supported only in Release and Protected modes. APK Tamper Protection is available from V8 SP4 onwards. From V8 SP4 Fixpack 20 onwards, support for Google Play App Signing has been added. Google Play App Signing is a mandatory signup for Android App Bundle support.

This feature is enabled only if the `addAPKTamperProtection` property is set as `true` in the androidbuild.properties file.

```
addAPKTamperProtection = true
```

For this feature to work, you must provide either of the information as follows:

- Add KeyStore entries from the Signing section in Visualizer: **Project Settings > Native > Android > Mobile/Tablet > Signing**.
- Alternatively, you can add the developerSigningKeyHash key in the androidbuild.properties file. An example of a typical developerSigningKeyHash is shown here.

```
developerSigningKeyHash:2otpMeAC68KCm7Q+F48tzTFtzmU=
```

The developerSigningKeyHash key helps you to utilize the APK Tamper Protection feature in the following scenarios:

- Google Play App Signing, where the key used to sign the APK that is being uploaded is different from the final APK delivered to customers from Google Play. For more information on how the Google Play App Signing process works, click here.
- CI/Cloud build environment, without actually revealing the original developer signing KeyStore information.

Generate the developerSigningKeyHash Key for Google Play App Signing

1. When customers enroll into the Google Play App Signing process , the **SHA-256** or **SHA-1** hash of the public key can be obtained. They can obtain the hash by signing in to Google Play Store Console, navigating to **Release Management** > **App Signing**, and then copying the **SHA-256/SHA-1 certificate fingerprint**.
2. Go to `<WorkSpace>\temp\<AppID>\build\luaandroid\extres`.
3. Locate and open the `PrintApkSignatureHash.jar` file.
4. Run the following command to generate the developerSigningKeyHash:
   `java -jar PrintApkSignatureHash.jar --fingerprint "<Hash-Algorithm>: <certificate fingerprint in Hexadecimal>"`
   Here, `Hash-Algorithm` can either be `SHA-1` or `SHA-256`.
   For example:

```
java - jar PrintApkSignatureHash.jar--fingerprint" SHA256: EB: 71: 4E:
90: 3D: 2A: 7E: 14: 4B: D1: 73: 47: 3A: EA: 3D: 06: C5: F2: 69: B5:
DC: BB: 28: 44: A0: 8D: AC: 17: E7: F2: 7F: 8F"
```

5. Here is a sample output that is generated.

```
developerSigningKeyHash : xxxxxxxxxxxxxxxxx
```

6. Copy and paste the output value in the `androidbuild.properties` file.

Important Points

a. If you specify KeyStore entries through Kony Visualizer, the KeyStore entries will take precedence over the developerSigningKeyHash key until Kony Visualizer V8 SP4 Fixpack 19.

b. From Kony Visualizer V8 SP4 Fixpack 20 onwards, if you specify Visualizer KeyStore entries and developerSigningKeyHash, both items are respected simultaneously and the app is launched if any one of these items matches. This enhancement helps you to test the APK locally before uploading it to Google Play.

c. If you do not want to upload the Upload Signing key to the CI cloud, use the `uploadSigningKeyHash` property in the **additionalbuild.properties** file as an alternative to specifying KeyStore entries.

d. You must provide either the KeyStore or uploadSigningKeyHash to test Google Play App Signing locally for the enrolled APK, which is tamper-protected. If you do not provide any of those values, the test APK that is generated will be signed by the debug key and the APK will not boot as the hash validation process fails at run time.

Generate the developerSigningKeyHash Key by using the KeyStore File

To manually sign the application by using your own keystore file, follow these steps:

1. Go to `<WorkSpace>\temp\<AppID>\build\luaandroid\extres`.
2. Locate and open the `PrintApkSignatureHash.jar` file.
3. Run either of the following commands to generate the developerSigningKeyHash:
   - For apps built in Kony Visualizer V8 SP4 Fixpack 19 or earlier, use the following command. This command generates the hash with SHA-1 algorithm.
     ```
     java -jar PrintApkSignatureHash.jar keyStorePath keyStorePassword keyAlias
     ```

- For apps built in Kony Visualizer V8 SP4 Fixpack 20 or later, use the following command. This command generates the hash with either SHA-1 or SHA-256 algorithm, depending on the `-- algorithm` input parameter.
  ```
  java -jar PrintApkSignatureHash.jar --storepath keyStorePath --storepass keyStorePassword --alias keyAlias --algorithm hash-logo
  ```

Here, the items are as follows:

- keyStorePath: Path to your actual developer signing key, which is used to upload your app's APK to the Google Play Store.
- keyStorePassword: Password of your developer KeyStore.
- keyAlias: Signing key alias of your developer KeyStore.
- hash-algo: Hashing algorithm that is used to generate the signing key hash. It can either be `SHA-1` or `SHA-256`.

Note: If the hash is generated with SHA-256 algorithm, the hash will not work for apps built in Kony Visualizer V8 SP4 Fixpack 19 or earlier. However, if the hash is generated with SHA-1 or SHA-256 algorithm, the hash will work for apps built in Kony Visualizer V8 SP4 Fixpack 20 or later.

4. Here is a sample output that is generated.

```
developerSigningKeyHash : xxxxxxxxxxxxxxxxxx
```

Note: If you are generating the hash of the upload signing key to support Google Play App Signing, use `uploadSigningKeyHash` as the key instead of `developerSigningKeyHash`.

5. Copy and paste the output value in the `androidbuild.properties` file.

Support for 32-bit and 64-bit Architectures in a Single APK

From V9 onwards, when you enable the `support64bit` property in the `androidbuild.properties` file and select the **Support 32-bit Devices** check box from **Project Settings > Native > Android**, the Kony Android build generates a `Fat` application. This `Fat` application supports all architectures: armeabi-v7a, x86, arm64-v8a, and x86_64.

```
support64bit = true
```

Note: Kony recommends that you use a `Fat` binary for testing purposes only, and Kony does not recommend you to upload a `Fat` binary to Google Play. Use either the Split APK feature or the Android App Bundle feature to reduce the size of the binary that is downloaded to customers' devices.

Split APKs based on Supported Architecture

Bundling all architectures into a single fat APK increases the APK size of the app that is delivered to customers. This APK Splitting feature, which has been introduced from V8 SP4 Fixpack 12 GA onwards, helps you to decrease the APK size that is downloaded based on the target platform architecture. For more information on the splitting of APKs, click here.

You must enable the `splitapks` property as `true` in the androidbuild.properties file. This action generates the architecture-specific individual .apk files and universal .apk file that supports all the architectures. This approach reduces the size of the .apk file.

```
splitapks = true
```

Note: Based on the specified value of the support64bit property in the androidbuild.properties file as well as on the selection of the Support x86 Devices and Support 32-bit Devices check boxes in the Kony Visualizer Project Settings, the Kony Android build generates a set of APKs, each with a single supported architecture.

- Architecture-specific .apk files are generated with this naming convention: `<appid>-<architecture>-<buildtype>.apk`
  Consider an app with appid as **KonySample**, for which the 64-bit ARM APK names for Debug

and Release modes are as follows:

- Debug mode: **KonySample-arm64-v8a-debug.apk**
- Release Unsigned mode: **KonySample-arm64-v8a-release-unsigned.apk**
- Release Signed mode: **KonySample-arm64-v8a-release-signed.apk**

- Universal .apk files with all selected architectures are generated with this naming convention: `<appid>-universal-<buildtype>.apk`
  Consider an app with appid as **KonySample**, for which the APK names for Debug and Release modes are as follows:
  - Debug mode: **KonySample-universal-debug.apk**
  - Release Unsigned mode: **KonySample-universal-release-unsigned.apk**
  - Release Signed mode: **KonySample-universal-release-signed.apk**

The APKs are generated under the following paths:

- For Mobile: **<workspace>\temp\<appid folder>\build\luaandroid\dist\<appid folder>\build\outputs\apk** folder.
- For Tablet: **<workspace>\temp\<appid folder>\build\luatabandroid\dist\<appid folder>\build\outputs\apk** folder.

> ***Note:*** For architecture-specific .apk files, the version code of the individual .apk file must be unique. This is because since both 32-bit and 64-bit APKs are supported in 64-bit devices, there will be a conflict while choosing the APK file. Specifying a higher version code for the 64-bit APK file results in a greater precedence, and thus the 64-bit APK file would be chosen for a 64-bit device. This process helps to leverage the higher performance of a 64-bit APK in a 64-bit device.

The Kony Android `build.gradle` file automatically handles the use cases related to APK versioning in the following manner:

1. The Kony Android `build.gradle` file assigns **architecture codes** in the following order of priority:

'armeabi-v7a':1 < 'x86':2 < 'arm64-v8a':3 < 'x86_64':4

2. The final Google Play version code of the individual .apk file is: **{(architecture code) * 1000} + (version code from project settings)**
   For example, if the version code in the Kony Visualizer Project Settings is 3. Then, the Google Play version codes of each architecture APK is as follows:
   - armeabi-v7a: **1003** {(1 * 1000) +3 }
   - x86 : **2003** {(2 * 1000) +3 }
   - arm64-v8a : **3003** {(3 * 1000) +3 }
   - x86_64 : **4003** {(4 * 1000) +3 }

> ***Note:*** If you want to split APKs based on density along with architecture, customize the build by adding the appropriate `build.gradle` entries in the Gradle Entries tab > build.gradle entries to Suffix section.

Generate Android App Bundle

Google Play's Dynamic Delivery feature uses your Android App Bundle to build and serve APKs that are optimized for each device configuration. This results in a smaller app download for customers by removing unused code and resources needed for other devices. The support for Android App Bundle generation has been added from V8 SP4 Fixpack 12 GA onwards. For more information about Android App Bundle, click here.

To enable this feature, you must set the `generateAppBundle` property as `true` in the androidbuild.properties file. The Kony Android build then generates the binary in App Bundle (aab) format. The AAB file is a Google Play upload format file, and is not an installable file.

> ***Note:*** Kony Visualizer and CI builds in Kony Visualizer do not provide support for the generation of binaries in the Android App Bundle (aab) format. You must configure Kony Visualizer to support the generation of 32-bit and 64-bit architectures in a single APK for generating the Android App Bundle.

```
generateAppBundle = true
```

The AAB file is generated in the following paths after the build:

- For Mobile : **<workspace>\temp\<appid folder>\build\luaandroid\dist\<appid folder>\build\outputs\bundle** folder.
- For Tablet : **<workspace>\temp\<appid folder>\build\luatabandroid\dist\<appid folder>\build\outputs\bundle** folder.

> ***Note:*** The Kony Android build generates an AAB file with all the selected architectures. The selected architectures are based on the specified value of the support64bit property in the androidbuild.properties file as well as on the selection of the Support x86 Devices and Support 32-bit Devices check boxes from the Kony Visualizer Project Settings.

To test how Google Play uses the AAB file to generate APKs and how those APKs behave when deployed to a device, follow these steps:

1. Download bundletool.

   > ***Note:*** Android provides a tool, named **bundletool**, to extract APKs from the AAB file and test them. For more information on the usage of bundletool, click here.

2. Extract the **APKs set** file from the aab file by using this command:
   ```
   java -jar bundletool.jar build-apks --bundle=<app-id>.aab --output=<app-id>.apks --overwrite
   ```
   Here, **APKs set** is an archive file with the extension as **.apks**.

3. The following command fetches the configuration details of the connected device and installs the device-compatible set of APKs:
   ```
   java -jar bundletool.jar install-apks --apks=<app-id>.apks --device-id=serial-id
   ```
   Here, **--device-id**: Optional parameter that helps to install the device-compatible set of APKs to a specific device (identified by **serial-id**), when multiple devices are connected.
   serial-id: Device identifier that the **adb devices** command returns.

> ***Note:*** If you want to estimate the download size of the APKs from Google Play, generate the device-specific set of APKs by following the procedure specified here.

As part of the V8 SP4 Fixpack 12 GA release, the Dynamic Delivery feature (on-demand delivery of modules when an app developer requests) is not supported for Kony Framework libraries. You can, however, implement the Dynamic Delivery feature for third-party libraries by using FFI code.

To sign the App Bundle file and ensure that the file is ready to be uploaded to Google Play, follow these steps

1. It is mandatory that you enroll into the Google Play App Signing process for you to be able to upload your App Bundle to the Play Console. Otherwise, you cannot upload the App Bundle to Google Play. For more information on how the Google Play app signing process works, click here.
2. The Release build generates a signed App Bundle with the Release Key value, which is specified in Kony Visualizer. This Release Key is usually either the Upload Key (if you have already enrolled for the Google Play App Signing process) or the actual Release Key (if you are opting for the Google Play App Signing process for the first time to submit an update in the form of an App Bundle to the existing app in Google Play). The signed App Bundle can then be uploaded to Google Play. While delivering optimized APKs to the device, Google Play signs the APKs with its own app signing key.
3. If the Release Key value is not specified in Kony Visualizer, the Release build generates an unsigned App Bundle. The **jarsigner** command can be used to sign the aab file. For further information on jarsigner, click here.

   ```
   jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
   <keystorefile>
   <app bundle file> alias_name
   ```

   > ***Note:*** apksigner is not supported to sign the App Bundle.

4. In Debug build, Gradle automatically signs the App Bundle with the Android SDK Debug Key.

Bundle a Customized Cordova-Generated Android Project

To bundle the manually customized version of your Cordova-generated Android project, you must set the `cordovabuildmode` property as `incremental` in the androidbuild.properties file. This feature is available from Kony Visualizer V8 SP4 Fixpack 47.

```
cordovabuildmode = incremental
```

For more information about how to manually customize the Cordova-generated Android project, click here.

Project Settings in Kony Visualizer

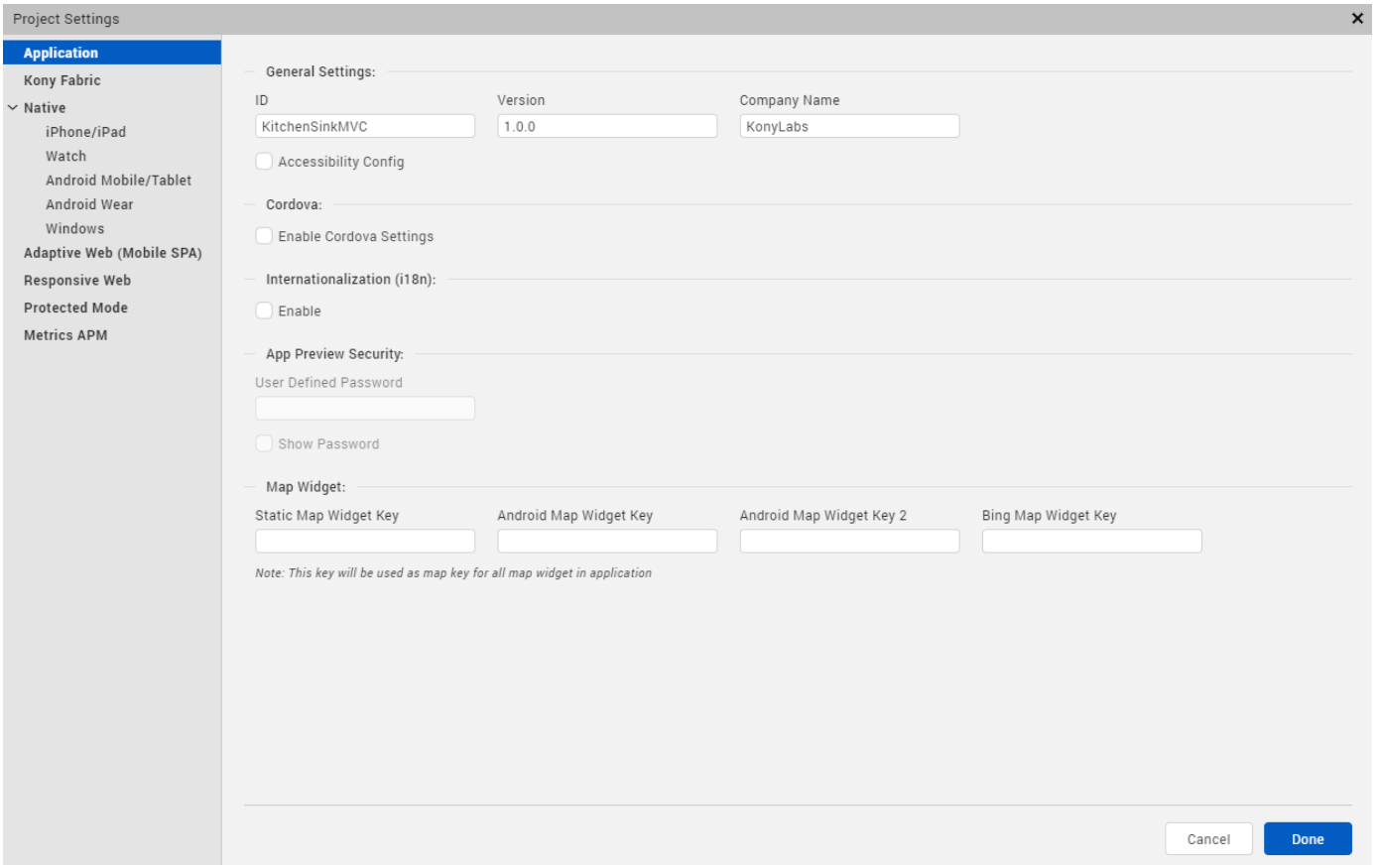
The project settings in Kony Visualizer are categorized into the following broad sections:

- Application
- Kony Fabric
- Native
    - General
    - Iphone/iPad
    - Watch
    - Android Mobile/Tablet
    - Android Wear
    - Windows
- Adaptive Web (Mobile Web)
- Responsive Web
- Protected Mode
- Metrics APM

Application Settings

Application properties are specific to the application. Using application settings, you can configure details about the Application ID, Version, Company name; set Accessibility configuration, Internationalization; configure Cordova settings, App preview security, and Map widget key fields.

Click to view the image



The following table describes all the fields in Application settings.

Section Name	Field Name	Description
General Settings	ID	Name of the application. It is auto populated.

	Version	A number that represents version of the application. It is auto-populated.
	Company Name	Name of the company is auto-populated.
	Accessibility Config	Enables assistive technologies such as TalkBack and VoiceOver. It assists visually-impaired users to navigate through various UI controls.
Cordova	Enable Cordova Settings	Configures the settings related to Cordova version.

	Version	Sets the Cordova version.
	Use globally installed Cordova version	Enables the globally-installed Cordova version.
Internationalization (i18n)	Enable	Sets various locales to the Visualizer project.
	Enable i18n Layout Config	Configures layout properties.
	Default Locale	Sets a default locale to the Visualizer project.

App Preview Security	User Defined Password	Enhances app preview security. While performing a cloud publish, you can set a password. To preview the app, you must enter the same password.
	Show Password	Unmasks the password.

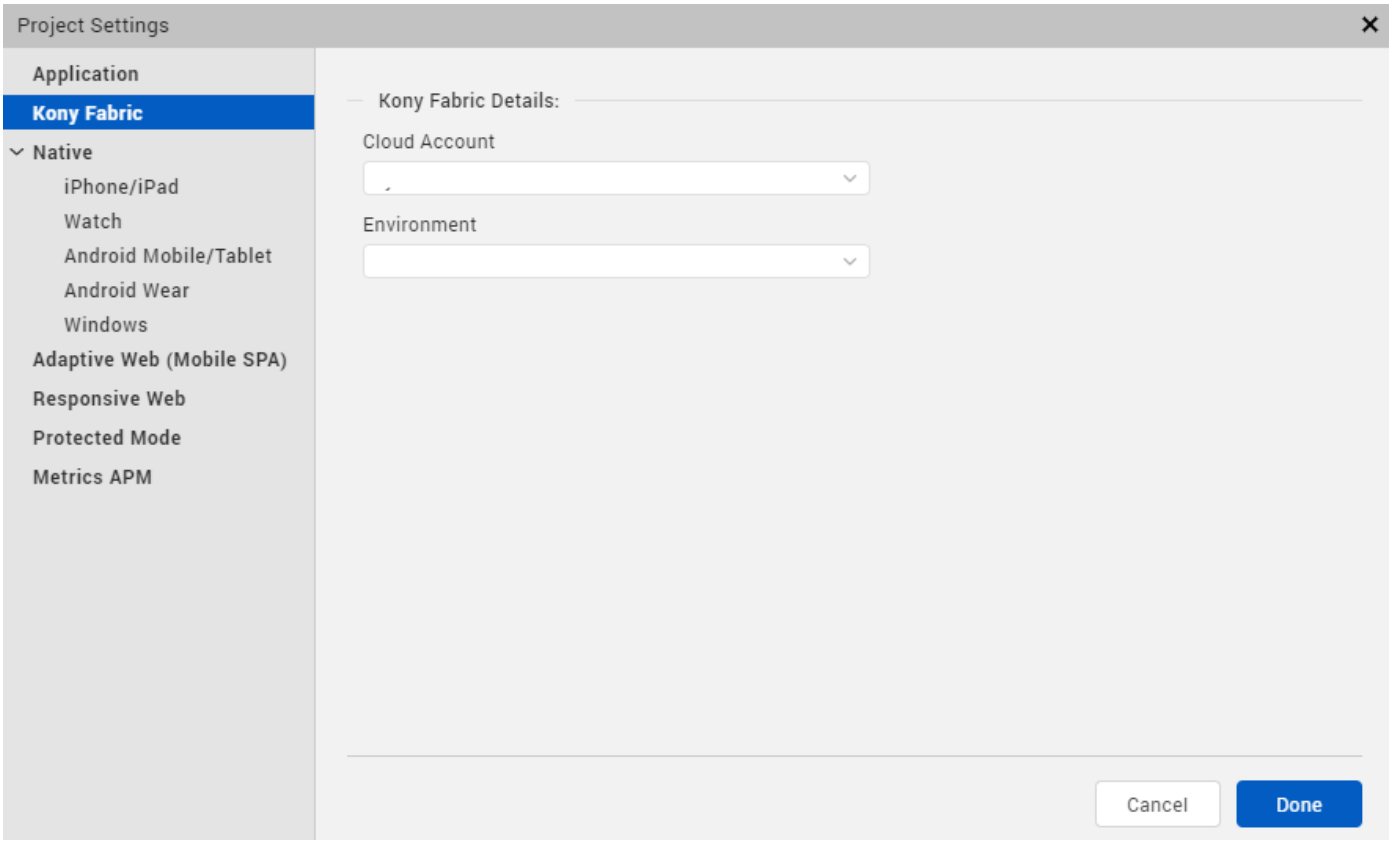
Map Widget	Static Map Widget Key	Map key enables application to display Google maps through the map widgets within the applications. Generate and enter the Static Map Widget key. For more information on generating Map API keys, refer Generating and Configuring Map API keys.

	Android Map Widget Key	For the applications using Version 1 of Google Maps API, enter the generated Android Map Widget key.
	Android Map Widget Key 2	For the applications using version 2 of Google Maps API, enter the generated Android Map Widget key.
	Bing Map Widget Key	Enter Bing Map Widget key.

Kony FabricSettings

Using Kony Fabric settings, you can configure the Cloud Account and Environment details.

Click to view the image



The following table describes all the fields in Kony Fabric settings.

Section Name	Field Name	Description
Kony Fabric Details	Cloud Account	Configures a cloud account.
	Environment	Configures an environment in the selected cloud.

Native Settings

Native app properties are divided into two categories: those that are common to all platforms and those that are platform-specific. These properties include: the logo image that your app displays, the types of screens and SDKs that the app supports, and how certificates are handled.

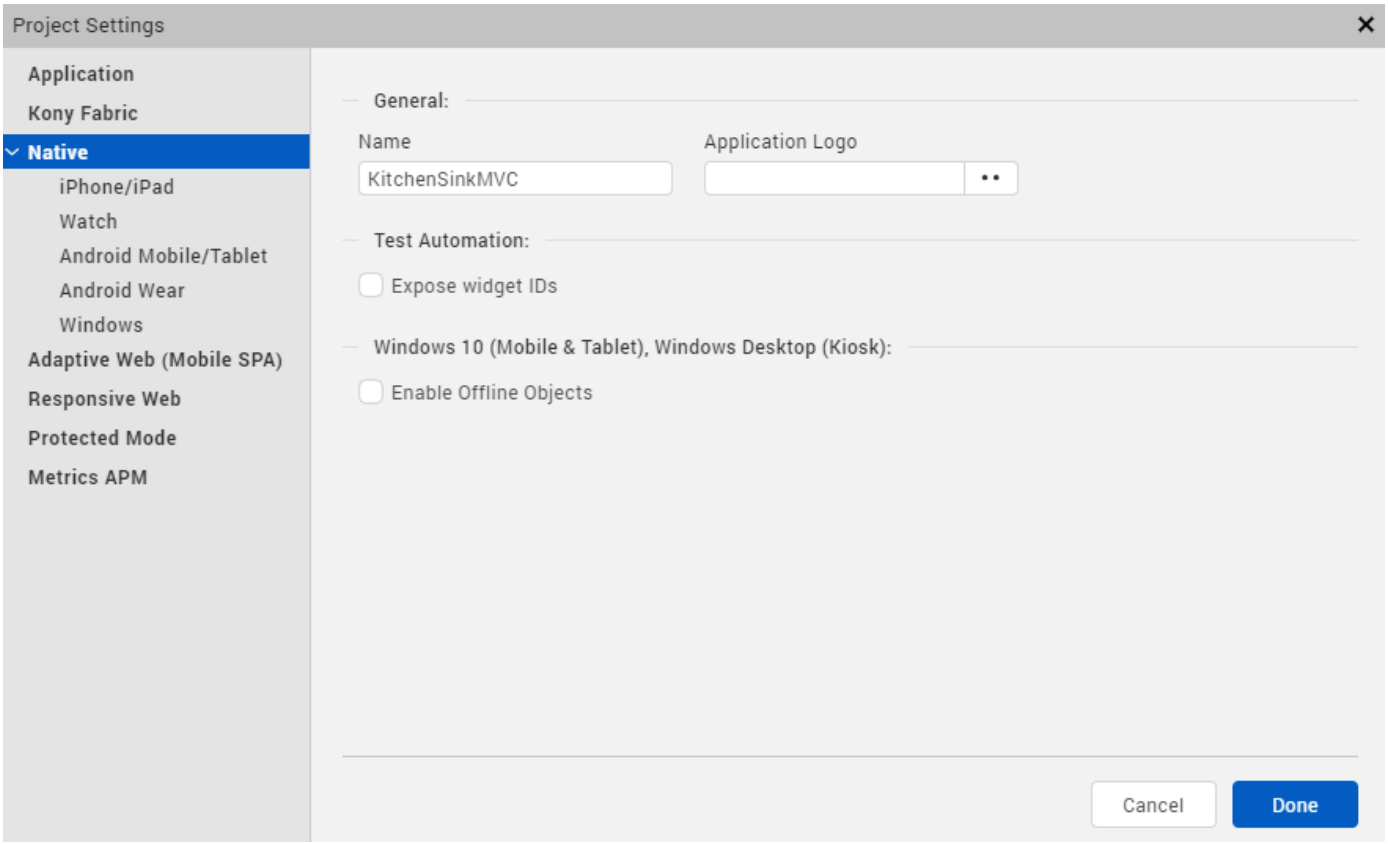
The following platforms are categorised based on their platform-specific properties:

- iPhone / iPad
- Watch
- Android Mobile / Tablet
- Android Wear
- Windows

General Settings

General Settings contain properties that are common to all platforms. Using General Settings, you can configure Application name, logo, Test automation as well as set offline object for Windows 10 mobile, tablet, and Windows Desktop.

Click to view the image



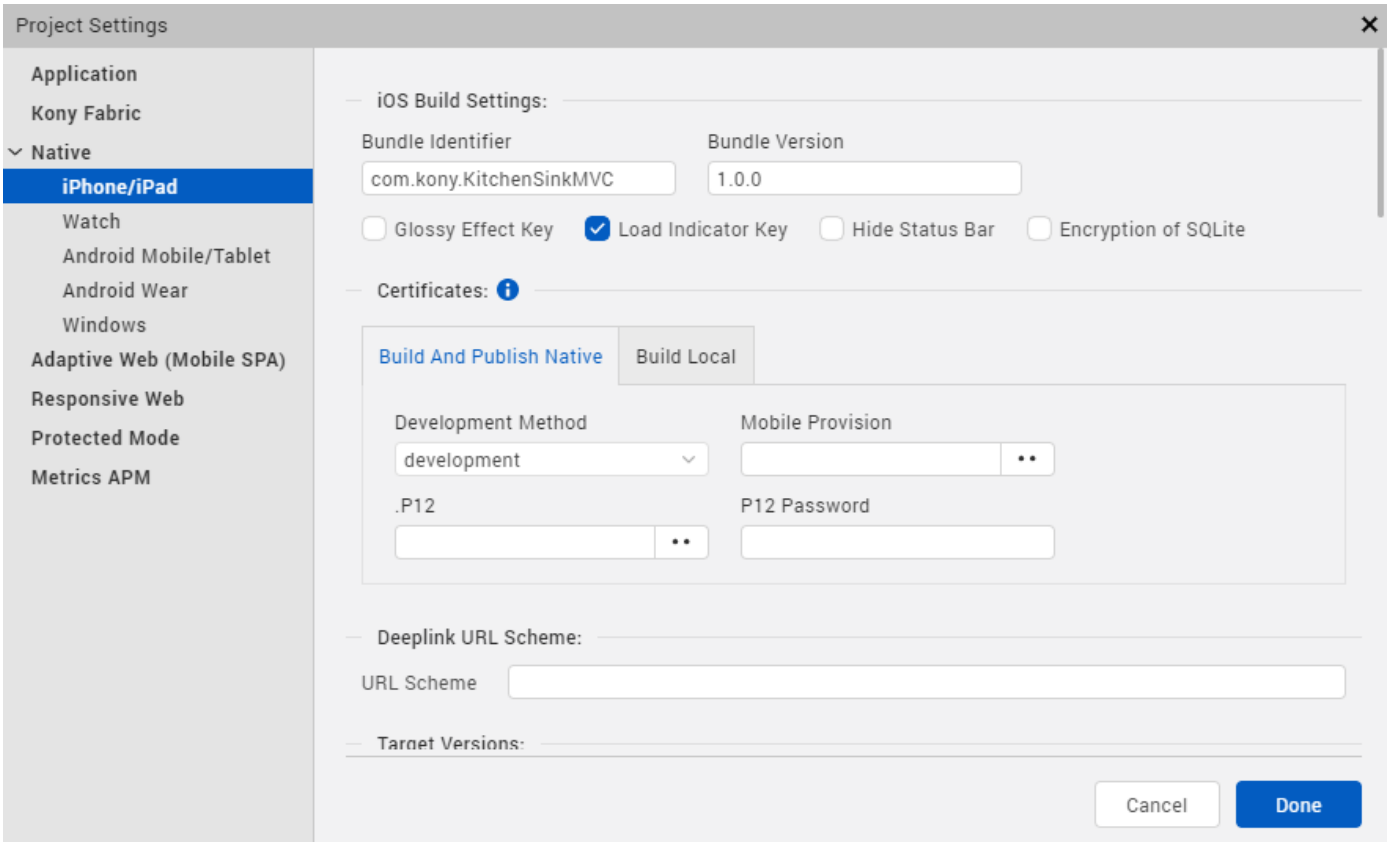
The following table describes all the fields in General settings.

Section Name	Field Name	Description
General	Name	Name of the Native channel version of the application. If no name is specified, the name specified under Application Properties is used.
	Application Logo	Sets an image as the application logo.
Test Automation	Expose widget IDs	Exposes the widget IDs.
Windows 10 (Mobile & Tablet), Windows Desktop (Kiosk)	Enable Offline Objects	Configures Offline objects for Windows 10 Mobile and Tablet, Windows Desktop (kiosk).

iPhone/ iPad

Using iPhone/ iPad settings, you can configure iOS Build Settings, Certificates; set Deep link URL Scheme; configure Target Versions, iPad Settings, and App Icons.

Click to view the image



The following table describes about all the fields in iPhone/ iPad settings.

Section Name	Field Name	Description
iOS Build Settings	Bundle Identifier	A Unique name that identifies the application bundle. It usually consists of three parts and follows the convention of com.kony.<appname>
	Bundle Version	A number that identifies the version of the application bundle.

	Glossy Effect key	Specifies if the glossy effect must be applied to the app icon. The default value is false.
	Load indicator key	Configures the load indicator in an application.
	Hide status bar	Hides the status bar in an application.
	Protected Mode	Ensures that your app is not run on a rooted/jail-broken device. To use this option, you must patch Xcode with the Finalizer utility. For information on patching Xcode with Finalizer, refer Install Finalizer Package. ***Note:*** Note: This option works only if the application is built in Release mode.
Certificates	Development Method	Refer the article to know about Development Method.
	Mobile Provision	Refer the article to know about Mobile Provision.
	.P12	Refer the article to know about .P12
	P12 Password	Refer to the article to know about P12 Password.

Deeplink URL Scheme	URL Scheme	Specifies a URL to which the application will deep-link to. For more information about deep-linking, Appendix E: the App Service Event.
Target Versions	iOS Version	Configures your iOS version.
iPad Settings	Application Launch Mode	Specifies the default mode of launching the application on iPad. Portrait is the default value.
	Supported Orientations	Specifies the supported orientations for the iPad. This depends on the launch mode.
App Icons	App Store 1x (iOS 1024 pt)	Each app in the App store can have an icon. Provide the image that you want displayed next to your app in the App Store. The default size of the icon should be 1024 pt.
	Notification iOS 7-11 20pt (iPhone and iPad)	Apps that support notifications should provide a small icon to display in notifications. Provide an image that you want displayed in notifications for iOS 7-11.

	Spot Light-Ios 5,6 Settings-iOS 5-11 29 Pt (iPhone)	Every app should provide a small icon that iOS can display when the app name matches the name in a Spotlight search. Provide an image that you want displayed during spotlight search for iPhone 5,6.
	Spot Light-Ios 7-11 40 Pt (iPhone and iPad)	Provide an image that you want to display during spotlight search for iOS 7-11.
	App-iOS 7-11 60 Pt (iPhone)	Provide an image that represents your app for iOS 7-11.
	Settings-Ios 5-11 29 Pt	Apps with settings should provide a small icon to display in the built-in Settings app. Provide an image to display your app for iOS 5-11.
	App-iOS 7-11 76 Pt	Provide an image that represents your app for iOS 7-11.
Platform Settings	Generic exception alert	When enabled, generic exception alerts like "system error" are fired and when disabled, detailed exception messages appear as alerts. The best practice is to keep the setting enabled for the release mode and disabled for the debug mode.

	Exception alert	When enabled, the system displays an exception alert. When disabled, the app crashes without displaying any exception or alert. The best practice to follow is to enable the feature for both release and debug modes.
	Paste Board Type	It will allow the user to copy paste content from the app to external writable area like message etc. • **system level** - it will allow to copy paste into other applications. • **Applevelpersistent** - it will allow to copy paste within the app and the messages are persistent will be available across the app restarts. • **Applevelnonpersistent** - it will allow to copy paste within the app and the messages are Not persistent and will not be available across the app restarts. • **Nopasteboard** - it will not allow paste anywhere.

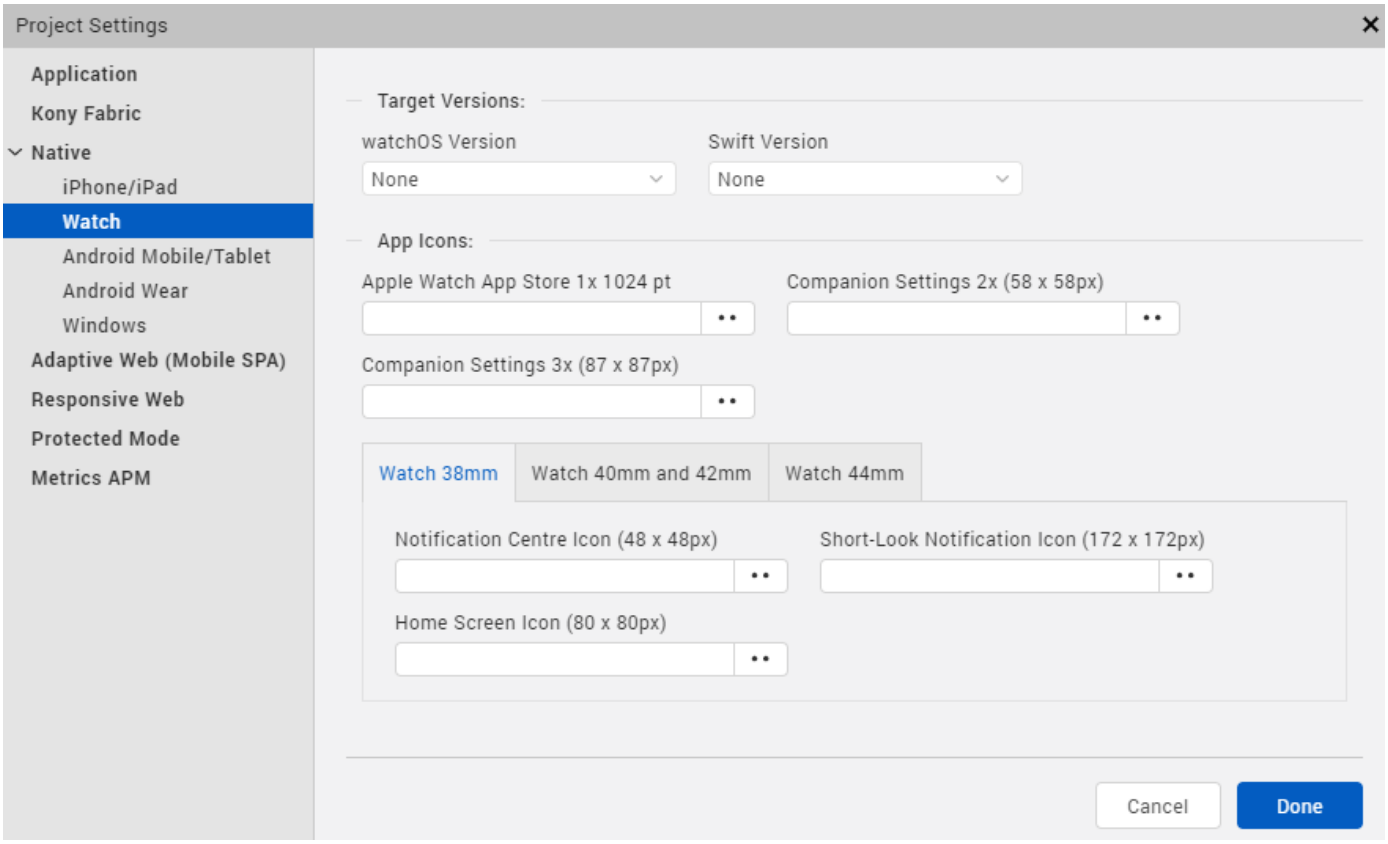
	Allow Self Signed Certificate	By default it is false, if true, it allows self signed certificate for development. ***Note:*** Self Signed Certificate option is only applicable if you use Network APIs in your application.
	Input Accessory View Type	The input accessory view type for widgets like text box, calendar, grouped widgets etc where you have next previous cancel buttons. This can be overridden by form level Input Accessory View Type.
	Anti Aliased Drawing	If enabled, allows smoother widgets and layout without any jagged edges.
	Backward_compatibility_mode	It is disabled, by default, If enabled, it will allow the application feature to behave as it would have behaved on earlier version (if there is any behavioral change in the latest version)
	extendTop	
	extendBottom	

	statusBarHidden	Lets you show or hide the status bar.
	statusBarStyle	Lets you set a style for the status bar.
	backgroundmodes	
	Camera Cancel Icon	Allows to set images to the icons which appear on the camera cancel icon, tap anywhere.
	Camera Settings Icon	Allows to set images to the icons which appear on the camera settings icon, tap anywhere.

Watch

Using Watch settings, you can configure Target Versions, App Icons, and Notification icons for 38mm and 42 mm Apple Watches.

Click to view the image



The following table describes all the fields in Watch settings.

Section Name	Field Name	Description
Target Versions	Watch OS Version	Configures the OS version number.
	Swift Version	Swift version number that the app is compatible with.
App Icons	Apple Watch App Store 1x 1024 pt	Each app in the Apple watch App store can have an icon. Provide the image that you want displayed next to your app in the App Store.

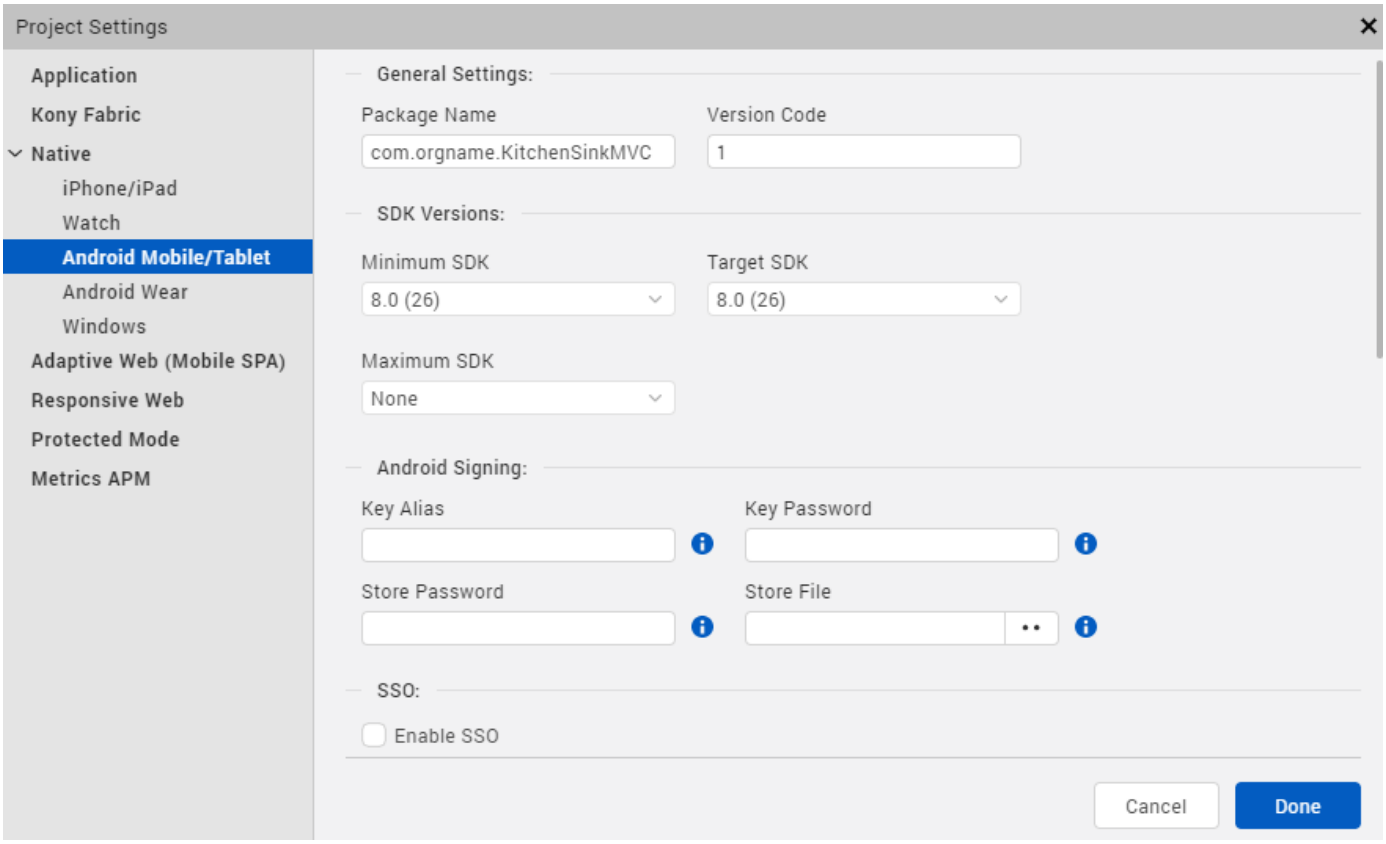
	Home Screen (All) (40 x 40)	Provide an image that represents your app on home screen.
	Companion Settings 2x	Configures companion app settings 2x.
	Companion Settings 3x	Configures companion app settings 3x.
Watch 38 mm	Notification Centre Icon	Apps that support notifications should provide a small icon to display in notifications for smart watches. Provide an image that you want displayed in notifications for watch 38 mm.
	Short- Look Notification Icon	Short-Look icon appears briefly, giving the wearer just enough time to scan a notification. Provide an image for the app icon in short look notifications for watch 38 mm.
Watch 40 mm and 42 mm	Notification Center Icon	Provide an image for app icon to display notifications in the center.

	Long-Look Notification Icon	Long-Look notifications display more information on screen from an app, such as message text and action buttons. Provide an image for app icon in long look notifications for 42 mm watch.
	Short-Look Notification Icon	Short-Look icon appears briefly, giving the wearer just enough time to scan a notification. Provide an image for app icon in short look notification for 42 mm watch.
Watch 44mm	Short-Look Notification Icon	Short-Look icon appears briefly, giving the wearer just enough time to scan a notification. Provide an image for app icon in short look notification for 42 mm watch.

Android Mobile/ Tablet

Using Android Mobile/ Tablet Settings you can configure General Settings, SDK versions, and Android signing.

Click to view the image



The following settings describe all the fields in Android Mobile/ Tablet settings.

Section Name	Field Name	Description

General Settings	Package Name	Package Name is the name used to search for an application in Google Play. Google Play is an online software store developed by Google for Android mobile devices. A software program called Market is pre-installed on most Android mobile devices. This software allows the users to browse and download third-party applications. ***Note:*** The name you specify for *Android Package* must contain at least two segments. A segment is a valid Java package name. The following are a few examples of valid Android Package names: • `com.konylabs.<ApplicationName>` • `com.kony.<ApplicationName>`
	Version Code	An internal version number, which is used to determine whether the application is a recent version. This version number is not shown to users. The value must be an integer. You can increase each version by one to indicate a newer version.
	Protected Mode	Ensures that your app is not run on a rooted/jail-broken device. ***Note:*** This option works only if the application is built in Release mode. The Protected Mode option works only if the application is built in Release mode. To know more about protecting your application, refer Applying Application Security.

SDK Versions	Minimum SDK	Select the **Minimum** SDK Version that needs to be supported for the application. The default minimum SDK value is 4.0. ***Notes:*** • Kony Visualizer does not support SDK Versions earlier than 4.0. • You must keep the minimum SDK value between 4.0 and 4.4. • The SDK values of 5.0 and above results in a build error(technical limitation). • The application must be built with a minimum version matching the device SDK version. For example, a device with 5.0 version of SDK cannot run an application built on 4.0.
	Target SDK	Select the **Target** SDK Version that needs to be supported for the application. ***Note:*** The Target SDK Version must be greater than or equal to the Minimum SDK Version.
	Maximum SDK	Select the **Maximum** SDK Version that needs to be supported for the application.
Android Signing	Key Alias	Used to sign the android binary automatically during the build process. Enter the alias of the key.

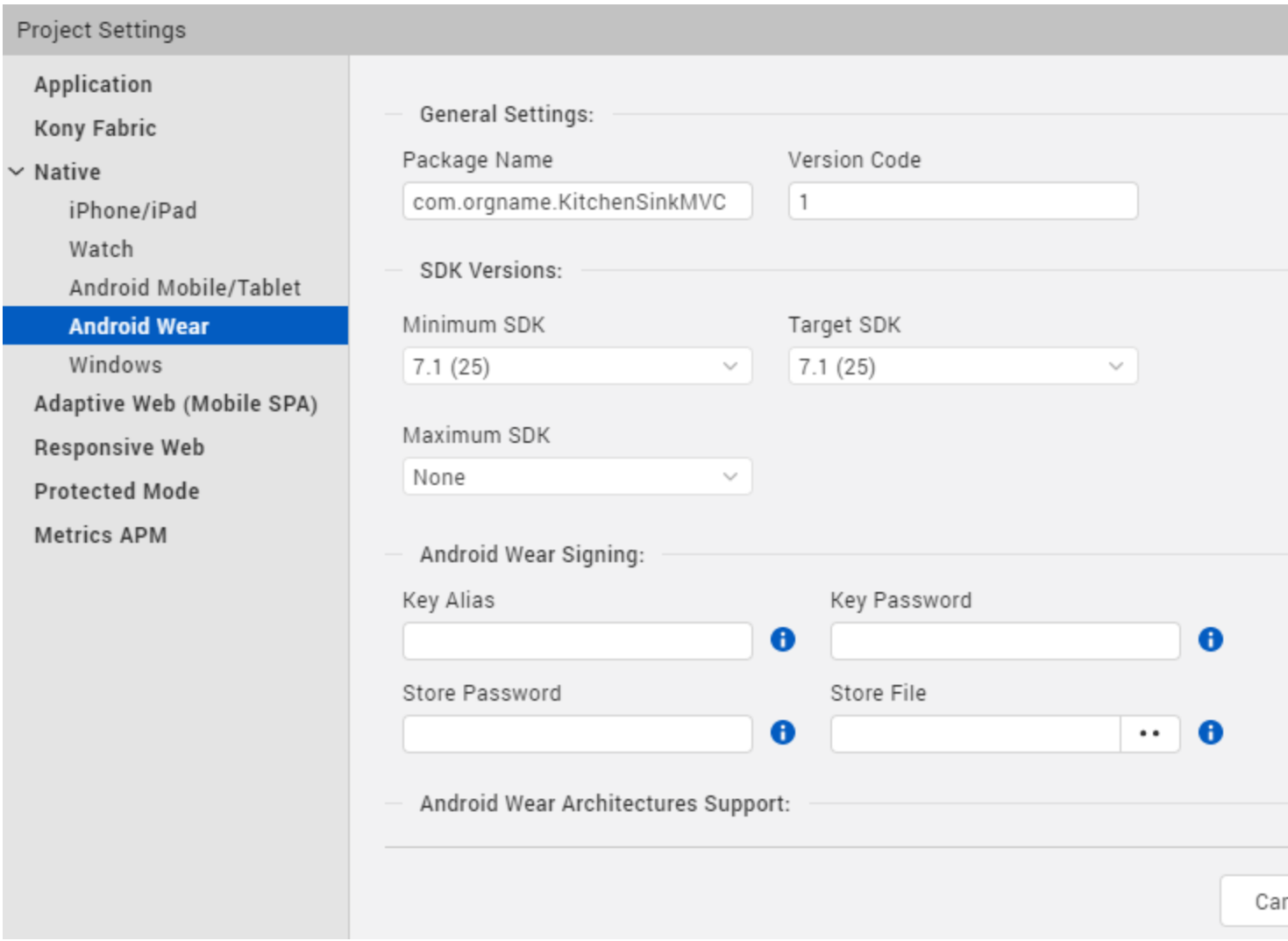
	Key Password	Enter the password for the key.
	Store Password	Enter the password for the store.
	Store File	Locate and configure the store file.
Support for Margin in Pixels	True	Supports margin in Pixels.
	False	Disables the support for margin in pixels.
Android Architectures Support	Support x86 Devices (Will increase app binary size)	Select this option to support any Android-x86 devices. Enabling this option increases the size of the binary that is generated.
	Support 32-bit Devices	Select this option to build an Android APK with 32-bit support.
Manifest Permissions, Tags and Gradle Build Entries	Permissions	Sets the permissions to true or false based on the application requirements. Set the appropriate permissions for Android Manifest file. For more information, refer The Android Manifest File. • To enable permissions, select the permissions from the left pane and click **Add >**. • To disable permissions, select the permissions from the right pane and click **< Remove**. ***Important:*** Add the `WRITE_EXTERNAL_STORAGE` setting if you need to save images in an external storage such as an SD Card.

	Tags	Adds tags to the Android manifest file directly from Kony Visualizer by specifying tag entries and attributes on the **Tags** tab. You can specify child tag entries and attributes for <manifest> and <application> tags and the Main Launcher <activity> tag. For more information on the tags you can add with the manifest or application tags, refer http://developer.android.com/guide/topics/manifest/manifest-intro.html.
	Gradle Entries	Imports additional gradle packages, applies external plugins, or specifies build-related configuration information, build dependencies, or the location of any external repositories or modules used by your Android application. **build.gradle entries to Prefix** imports additional Gradle packages, or specifies external plugins to use in the build. **build.gradle entries to Suffix** customizes build logic. For example, you can add compilation dependencies such as Google and Android support repositories, local library modules, or local and remote repository paths. For more information, see Organizing Build Logic. **gradle.properties entries** configures project-wide Gradle settings, such as the Gradle daemon's maximum heap size or proxy settings. For more information, see The Build Environment. **settings.gradle entries** specifies external modules (Gradle-based third-party Android libraries) to include when building your application. For more information, see Configure Your Build.

Android Wear

Using Android Wear Settings you can configure General Settings, SDK versions, and Android signing.

Click to view the image



The following settings describe all the fields in Android Wear settings.

Section Name	Field Name	Description

General Settings	Package Name	Package Name is the name used to search for an application in Google Play. Google Play is an online software store developed by Google for Android mobile devices. A software program called Market is pre-installed on most Android mobile devices. This software allows the users to browse and download third-party applications. ***Note:*** The name you specify for *Android Package* must contain at least two segments. A segment is a valid Java package name. The following are a few examples of valid Android Package names: • `com.konylabs.<ApplicationName>` • `com.kony.<ApplicationName>`
	Version Code	An internal version number, which is used to determine whether the application is a recent version. This version number is not shown to users. The value must be an integer. You can increase each version by one to indicate a newer version.

SDK Versions	Minimum SDK	Select the **Minimum** SDK Version that needs to be supported for the application. The default minimum SDK value is 7.1 (25). ***Notes:*** • Kony Visualizer does not support SDK Versions earlier than 4.0. • You must keep the minimum SDK value between 4.0 and 4.4. • The SDK values of 5.0 and above results in a build error(technical limitation). • The application must be built with a minimum version matching the device SDK version. For example, a device with 5.0 version of SDK cannot run an application built on 4.0.
	Target SDK	Select the **Target** SDK Version that needs to be supported for the application. ***Note:*** The Target SDK Version must be greater than or equal to the Minimum SDK Version.
	Maximum SDK	Select the **Maximum** SDK Version that needs to be supported for the application. The maximum SDK value is 9.
Android Signing	Key Alias	Used to sign the android binary automatically during the build process. Enter the alias of the key.

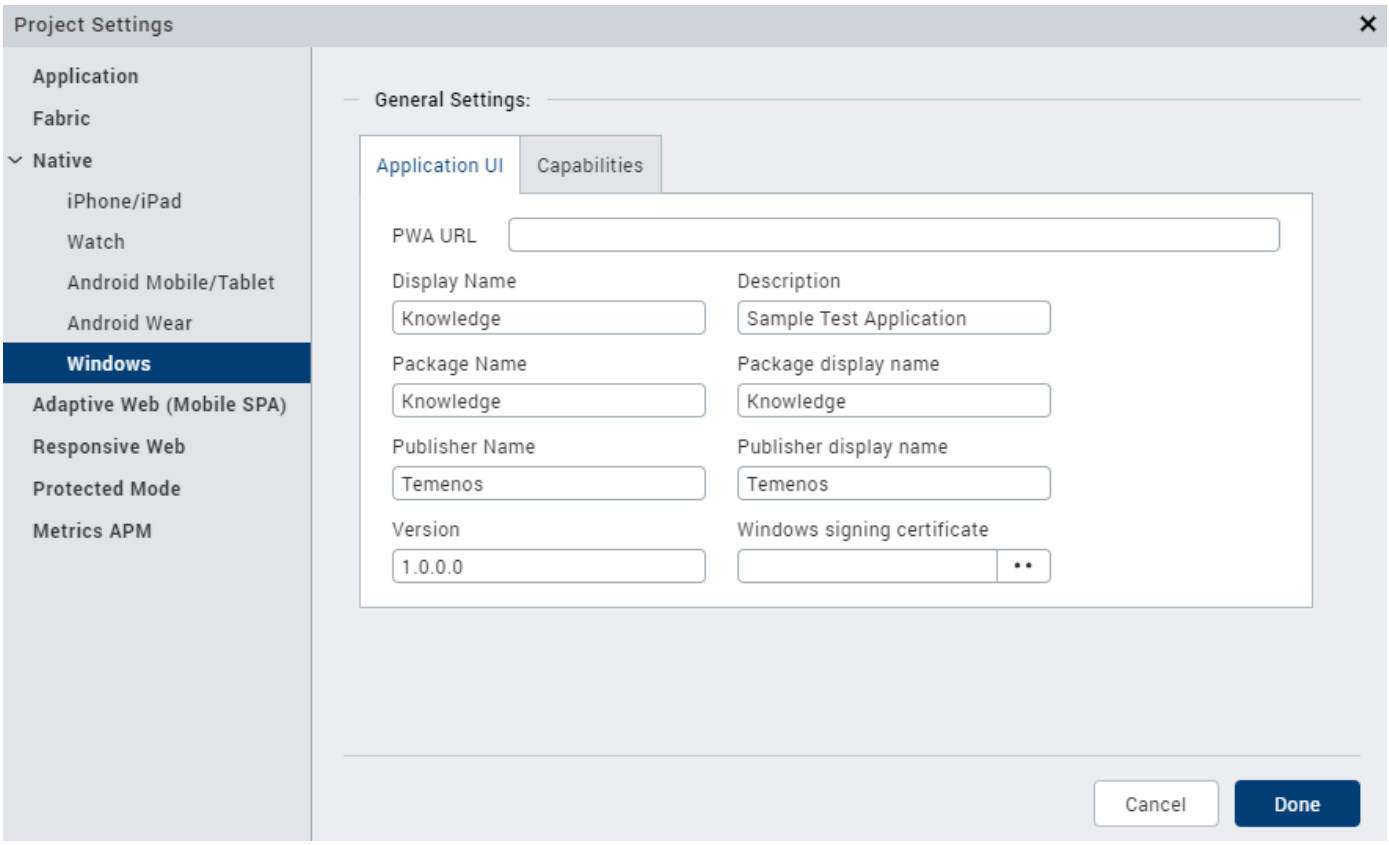
	Key Password	Enter the password for the key.
	Store Password	Enter the password for the store.
	Store File	Locate and configure the store file.
Android Wear Architectures Support	Support x86 Devices (Will increase app binary size)	Select this option to support any Android-x86 devices. Enabling this option increases the size of the binary that is generated.
	Support 32-bit Devices	Select this option to build an Android APK with 32-bit support.
Manifest Permissions, Tags and Gradle Build Entries	Permissions	Sets the permissions to true or false based on the application requirements. Set the appropriate permissions for Android Manifest file. For more information, refer The Android Manifest File. • To enable permissions, select the permissions from the left pane and click **Add >**. • To disable permissions, select the permissions from the right pane and click **< Remove**. ***Important:*** Add the `WRITE_EXTERNAL_STORAGE` setting if you need to save images in an external storage such as an SD Card.

	Tags	Adds tags to the Android manifest file directly from Kony Visualizer by specifying tag entries and attributes on the **Tags** tab. You can specify child tag entries and attributes for <manifest> and <application> tags and the Main Launcher <activity> tag. For more information on the tags you can add with the manifest or application tags, refer http://developer.android.com/guide/topics/manifest/manifest-intro.html.
	Gradle Entries	Imports additional gradle packages, applies external plugins, or specifies build-related configuration information, build dependencies, or the location of any external repositories or modules used by your Android application. **build.gradle entries to Prefix** imports additional Gradle packages, or specifies external plugins to use in the build. **build.gradle entries to Suffix** customizes build logic. For example, you can add compilation dependencies such as Google and Android support repositories, local library modules, or local and remote repository paths. For more information, see Organizing Build Logic. **gradle.properties entries** configures project-wide Gradle settings, such as the Gradle daemon's maximum heap size or proxy settings. For more information, see The Build Environment. **settings.gradle entries** specifies external modules (Gradle-based third-party Android libraries) to include when building your application. For more information, see Configure Your Build.

Windows

Using Windows Settings, you can configure configure General Application UI Settings and Capabilities for a Windows app.

Click to view the image



The following table describes all the fields in Windows Phone Settings.

Section Name	Field Name	Field Name
Application UI	PWA URL	If you want to generate a windows native application from a PWA app, specify the URL of the published Progressive Web App.

	Display name	Specifies the display name of the application visible in the applications list.
	Description	Specifies the description to be displayed on the tile. For example a tile on finance can contain a generic description about stocks.
	Package name	A unique name to identify a specific application. It is generally in the format `domain.company.application`
	Package display name	The name with which the application is submitted to Google play. This name is used to search the application in Google play.
	Publisher name	The publisher of the app. This default value is the name of the project. This attribute is required for certain types of apps, such as push-enabled apps.
	Publisher display name	Display name of the publisher.
	Version	Internal version number of the package.
	Windows signing certificate	If you have a certificate to publish an app to the Microsoft store, you can upload the certificate in this field.

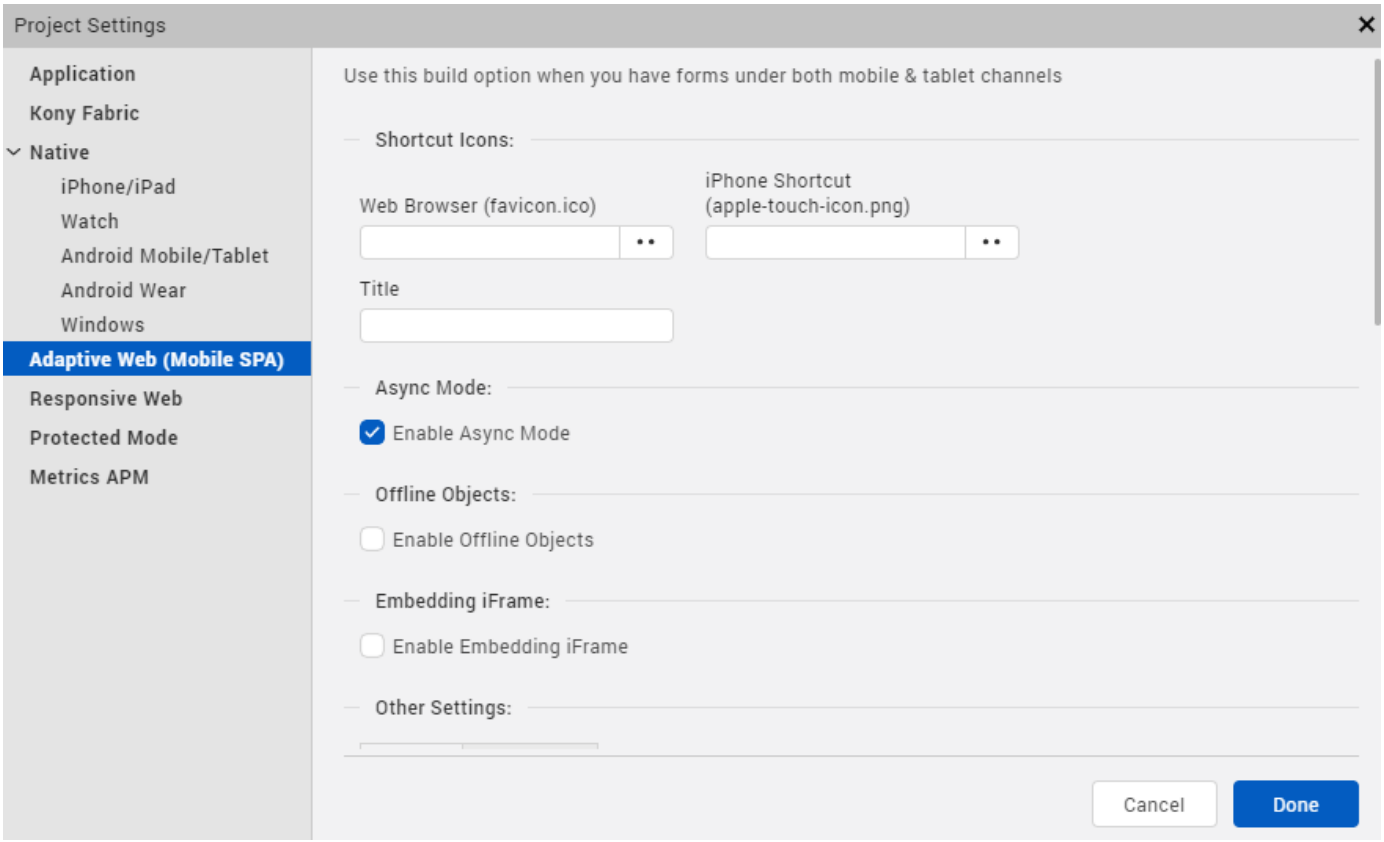
In the **Capabilities** tab, set the permissions to true or false based on the application's requirements. For more information about each of the permissions refer Capabilities and requirements.

- To enable permissions, select the permissions that are currently *false* and click **Add >**.
- To disable permissions, select the permissions that are currently *true* and click **< Remove**.

Adaptive Web (Mobile SPA)

Adaptive Web (Mobile SPA) is the browser on the device. Adaptive Web (Mobile SPA) properties define the properties of the application on Adaptive Web (Mobile SPA) for various platforms. You can set the Shortcut icons for the application, configure Async mode, and define Base fonts.

Click to view the image



The following table describes all the fields in Mobile Web settings.

Section Name	Field Name	Description

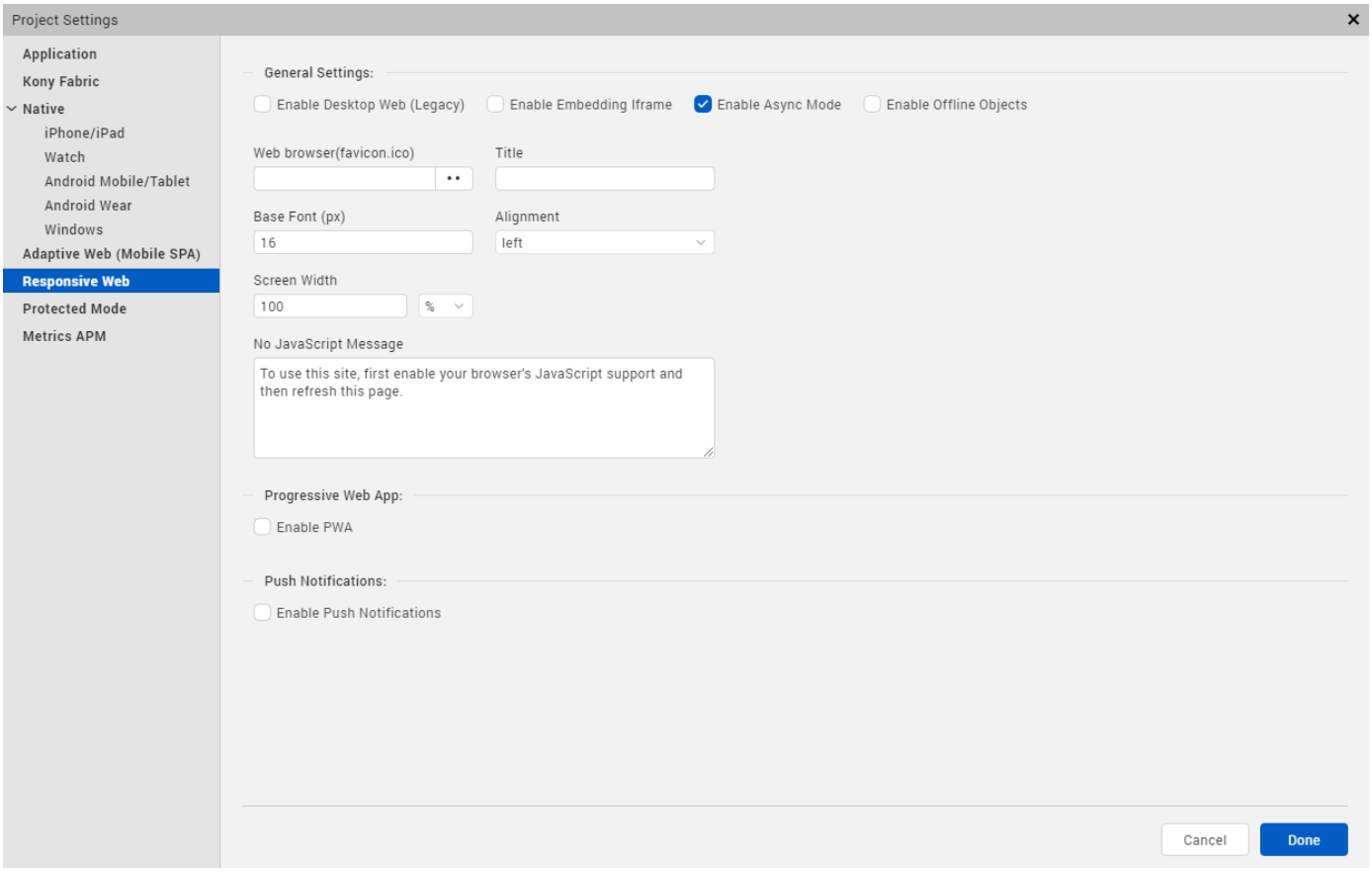
Shortcut Icons	Web Browser	Configures an icon to represent the app in a web browser.
	iPhone Shortcut	Configures an icon to represent the app on an iPhone screen.
	Title	Configures title of the application in a web browser.
Async Mode	Enable Async Mode	All the network calls will be in asynchronous mode when enabled.
General	Default image while loading	Set the default image to display while the app is loading.
	Phone format Indicator	Highlights a telephone number clearly in the browser.
	Requires GPS functionality	Enables the application to use the GPS functionality.
	Error Messages	The message to be displayed to the user in case of an error. Error messages are pre-populated. If you want to display a different message, overwrite this message.
Base Fonts	iPhone (px)	Configures the base font of iPhone in pixel.

	Android	Configures the base font of Android 240, Android 320, Android 360, Android 400, Android 440, Android 480, and Android 640 in pixel.
	SPA iPad	Configures the base font of SPA iPad in pixel.
	SPA Android Tablet	Configures the base font of SPA Android Tablet 800, Android Tablet 1024, Android Tablet 1280 in pixel.
	SPA Windows Tablet	Configures the base font of SPA Windows tablet in pixel.

Responsive Web

Responsive Web properties specify the properties of the application on Responsive Web for various platforms. You can configure Embedding Iframe, and Async Mode; set the Web Browser icon, application title, BaseFont, Alignment, Screen width; and enable Progressive Web App.

Click to view the image



The following table describes all the fields in Desktop Web settings.

Section Name	Field Name	Description
General Settings	Enable Responsive Web	Activates the Responsive Web Design for your desktop application.
	Enable Embedding Iframe	Sets the SPA or Desktop Web application behavior in a sub window. Enabling Embedding Iframe allows an application to open in a sub window.

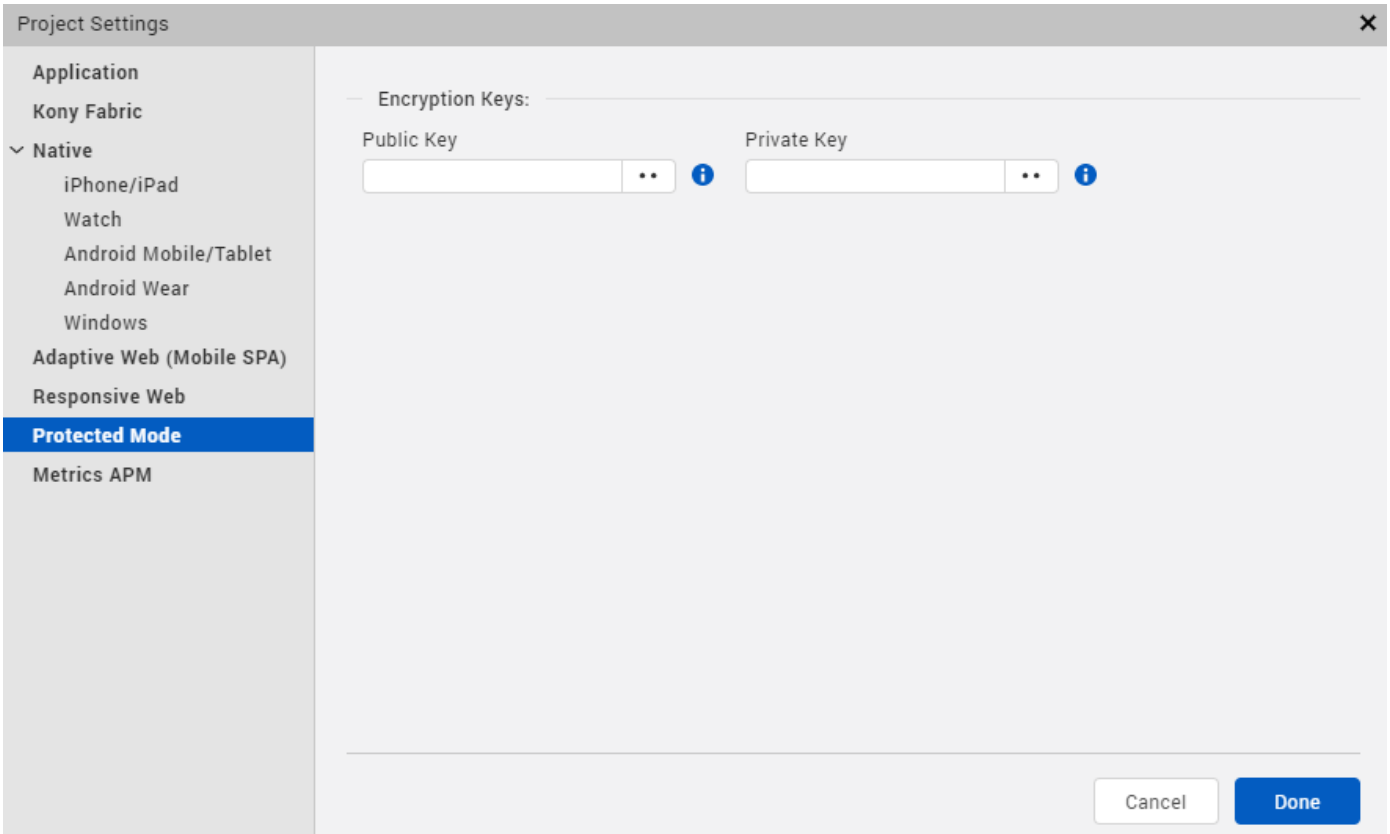
	Enable Async Mode	All the network calls will be in asynchronous mode when enabled.
	Web browser (favicon.ico)	Configures appropriate icon for an application in the Desktop Web Browser.
	Title	Sets the title of an application in the Desktop Web Browser.
	Base Font (px)	Configures appropriate base font size in pixel.
	Alignment	Defines the alignment of the application in the Desktop Web browser. The possible values are center, left, and right.
	Screen Width	Specifies the width that the application occupies in the Desktop Web browser. The Screen width value can be in percentage or pixel.
	No JavaScript Message	The message to be displayed when your browser does not support JavaScript.
Progressive Web App	Enable PWA	Builds a Progressive Web App.
Push Notifications	Enable Push Notifications	Enables Push Notifications for the app.

Protected Mode

Protected mode is a Kony Fabric token validation method. A successful validation occurs when the public key matches with the private key. It ensures that your app is not run on a rooted/ jail-broken device. Using Protected mode settings, you can configure the public and private encryption keys.

> ***Note:*** This option works only if the application is built in Release mode. The Protected Mode option works only if the application is built in Release mode. To know more about protecting your application, refer Applying Application Security.

Click to view the image



The following table describes all the fields in Protected Mode settings.

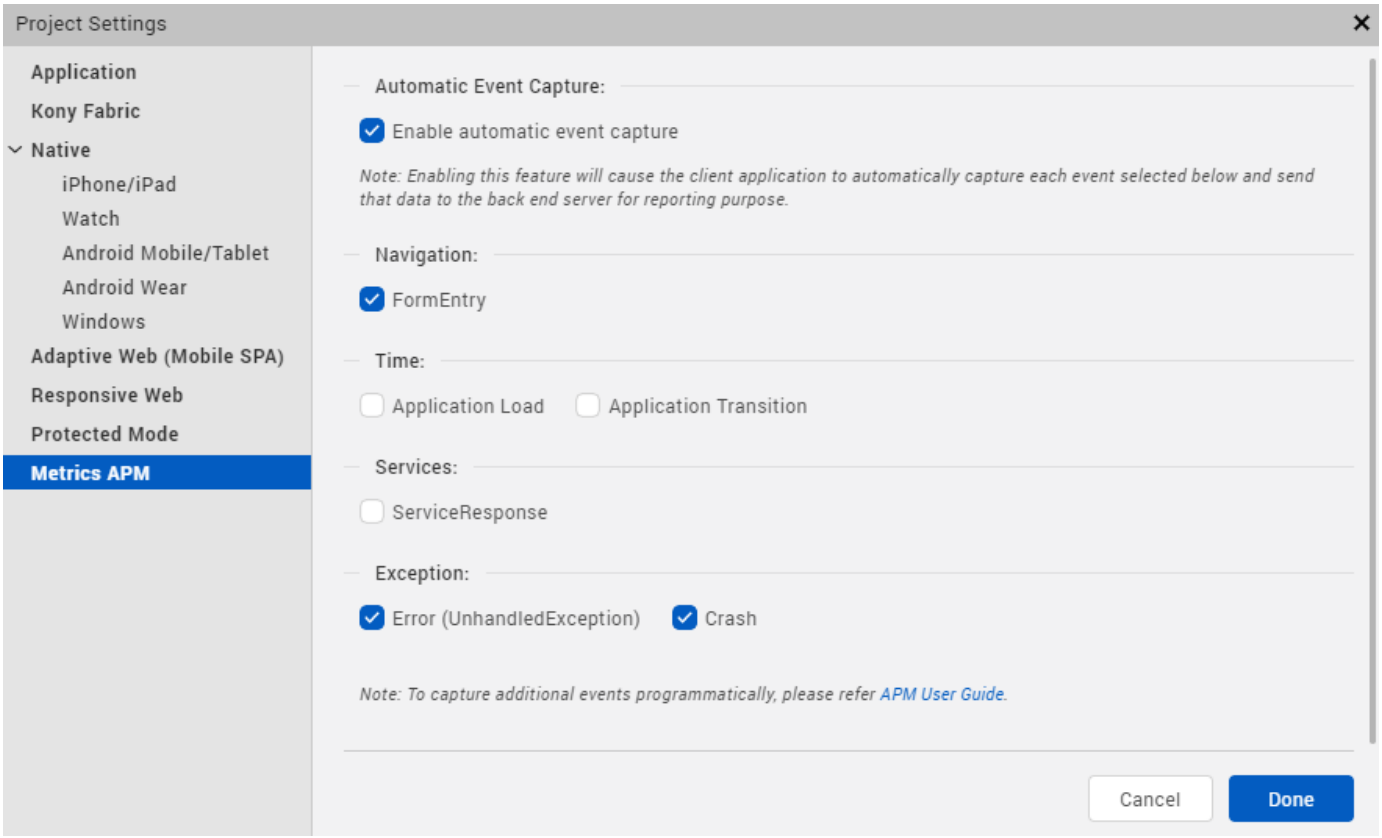
Section Name	Field Name	Description
Encryption Keys	Public Key	Configures the public key.

	Private Key	Configures the private key.

Metrics APM

Metrics Application Performance Monitoring deals with the set of metrics acquired by tracing the events of user's interaction with the application.

Click to view the image



The following table describes all the fields in Metrics APM settings.

Section Name	Field Name	Description
Automatic Event Capture	Enable automatic event capture	Captures each event and send the data to the backend server for reporting purpose.

Navigation	FormEntry	Allows automatic tracking for iOS native and android apps when a form in an application is opened.
Time	Application load	This event is invoked when the application is loaded and the first form is shown.
	Application Transition	This event is invoked when the application transits either from foreground to background and vice versa.
Services	ServiceResponse	Allows automatic tracking when an HTTP response is received from the service request or network call in the application.
Exception	Error	This event is invoked when application ends up in an unhandled exception in JavaScript code that the global exception handler catches up.
	Crash	This event is invoked when application crashes and resumes after crash.

Set Native App Properties

Native app properties are divided into two categories: those that are common to all platforms, and those that are platform-specific. These properties range from the logo image your app displays to the types of screens and SDKs the app supports, and how certificates are handled.

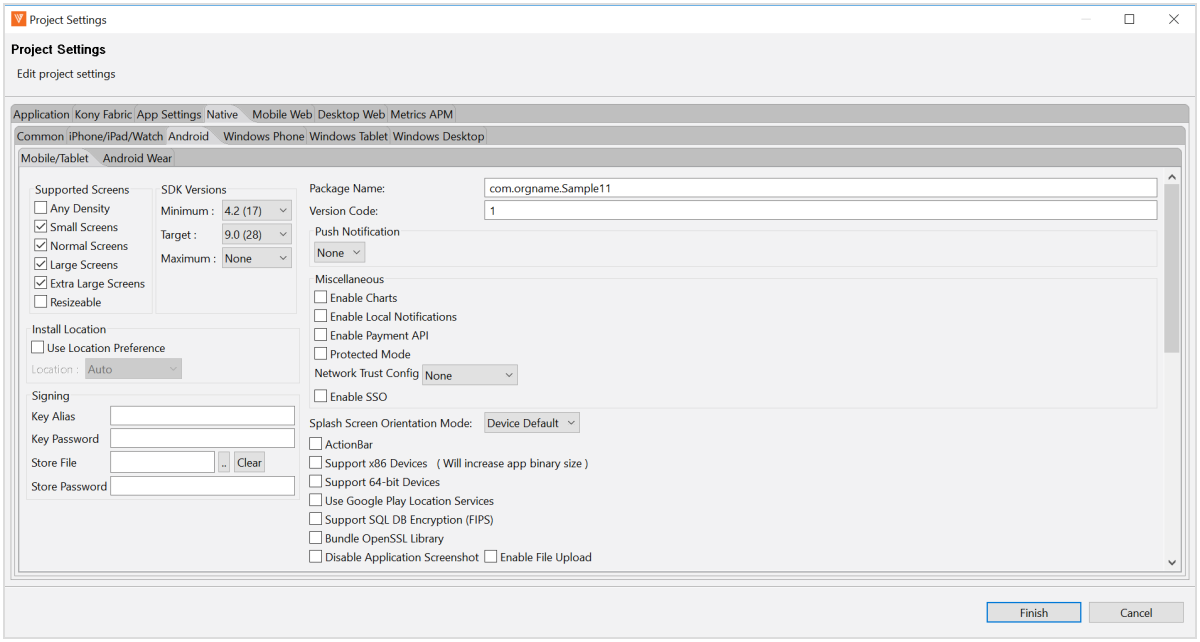
From Kony Visualizer V8 SP4, you can also enable certain Android features by manually adding the corresponding properties to the androidbuild.properties file.

To set Native app properties, follow these steps:

1. In Kony Visualizer, from the Project Explorer, click **Project Settings**. The Project Settings window appears.
2. Click the **Native** tab. A list of sub-tabs appears.
3. Click the **Common** sub-tab, and set the following properties:
    a. Enter a **Name** for the Native channel version of the application. If no name is specified, the Application ID specified under **Application Properties** is used.
    b. Browse and select an image file for the logo.

> *Important:*
> - The logo you specify here will be renamed to `icon.png` while building the application on the iPhone platform.
> - For Desktop, the icon dimension must be in multiples of 8. Minimum pixels can be 8 x 8 and maximum pixels can be 248 x 248.

4. Click the **Android** sub-tab, and set the following properties:



a. In the **Supported Screens** section:

- **Any Density** - If this option is enabled, the application can accommodate any screen density for a resource. You do not have to use this option if your app directly manipulates bitmaps. Generally this option must always be set to true.
- **Small Screens** - If this option is enabled, the application supports smaller screen form-factors. A small screen is one with a smaller aspect ratio than the normal (traditional HVGA) screen. An application that does not support small screens is not available for small screen devices from external services (such as Google Play).
- **Normal Screens** - If this option is enabled, the application supports normal screen form-factors. Traditionally, this is an HVGA medium density screen, but WQVGA low density and WVGA high density are also considered to be normal.
- **Large Screens** - If this option is enabled, the application supports larger screen form-factors. A large screen is defined as a screen that is significantly larger than a normal handset screen, and might require some special care on the application's

part to make good use of it. The application may rely on resizing by the system to fill the screen.

If this option is unchecked, it enables screen compatibility mode.

- **Extra Large Screens** - If this option is enabled, the application supports extra large screen form-factors. An extra large screen is defined as a screen that is significantly larger than a large screen, for example, tablet (or something larger) and may require special care on the application's part to make good use of it. The application may rely on resizing by the system to fill the screen. If the option is unchecked, this will generally enable screen compatibility mode.
- **Resizable** - If this option is enabled, the application is resizable for different screen sizes. This property is deprecated by Android SDK and is supported only for customers on the 2.6 plug-in. This property enables you to run an application in the compatibility mode. For more information, see Support Screen Elements for the Android App Manifest.

b. In the **SDK Versions** section:

- Select the **Minimum** SDK Version that needs to be supported for the application. The default minimum SDK value is 4.0.

> ***Notes:***
> - Kony Visualizer does not support SDK Versions less than 4.0.
> - You must keep the minimum SDK value between 4.0 and 4.4.
> - The SDK values of 5.0 and above results in a build error(technical limitation).
> - The application must be built with a minimum version matching the device SDK version. For example, a device with 5.0 version of SDK cannot run an application built on 4.0.

- Select the **Target** SDK Version that needs to be supported for the application.

  > ***Note:*** The Target SDK Version must be greater than or equal to the Minimum SDK Version.

- Select the **Maximum** SDK Version that needs to be supported for the application.

c. **Package Name**: Package name is a unique name to identify a specific application. It is generally in the format `domain.company.application`.

> ***Note:*** The name you specify for *Android Package* must contain at least two segments.

A segment is a valid Java package name. The following are a few examples of valid Android Package names:

- `com.konylabs.<ApplicationName>`
- `com.kony.<ApplicationName>`
- `com.konysolutions.<ApplicationName>`
- `com.kony.<ApplicationName>_Android`.

d. **Version Code**. This is an internal version number. This number is used to determine whether the application is a recent version. This version number is not shown to users. The value must be an integer. You can increase each version by one to indicate a newer version.

e. In **Push Notification** section:

- **GCM** - Select this option to enable Push Notifications for the application. This option copies the libraries required for push notification into the project during build time.

> ***Important:*** GCM (Google Cloud Messaging) is supported only for Android SDK Versions 2.3 and above.

- **Custom GCM Broadcast Receiver (Optional)** - If your application requires to override the default GCM broadcast receiver behavior, you can provide your own custom broadcast receiver. To customize the GCM receiver, see Customizing GCM Broadcast Receiver.

- **FCM** Select this option to enable Push Notifications for the application. This option copies the libraries required for push notification into the project during build time.
  - **Custom FCM Service (Optional)** - If your application requires to override the default FCM service, you can provide your own custom FCM service. To customize the FCM service, see Customizing FCM Service.

f. In the **Miscellaneous** section:

- **Protected Mode** - Selecting this option ensures that your app is not run on a rooted/jail-broken device.

> ***Note:*** This option works only if the application is built in Release mode. The Protected Mode option works only if the application is built in Release mode. To know more about protecting your application, see Applying Application Security.

- **Enable Local Notifications** - Select this option to enable notifications scheduled by an app and delivered on the same device. They are suited for the apps with time-based behaviors, such as calendar events.

- **Enable Payment API** - Selecting this option enables online transactions in applications.

- **Network Trust Config** - Using this option, you can control the certificates that are used.

- **None** - No certificates are allowed. This means that if the certificate is present in the Android Trust store, it will allow the N/W call to proceed; otherwise, it throws an exception. With this option, servers having non-trusted or self-signed certificates are not accessible via the app on the device.
- **All** - All types of certificates are allowed regardless of whether they are bundled. This option is useful during the development phase of an app, but not for publication. With this option, all servers are accessible regardless of the kind of certificate they hold (i.e. self-signed, non-trusted, trusted). Due to the lack of security inherent in this option, the Google Play store rejects such apps when they are submitted for publication.
- **Allow Bundled** - Only the certificates that are bundled along with the app are allowed. With this option, the app can communicate **only** with servers that have the certificate(s) that are bundled with the app.
  - To bundle the certificate in the application, copy the certificate under the following folder: For mobile, `<workspace>/<app>/resources/mobile/native/android/assets/certs`. For tablet, `<workspace>/<app>/resources/tablet/native/androidtab/assets/certs`

    > ***Note:*** If an *assets* folder does not exist, create an *assets* folder under respective locations as indicated above.
    > Create *certs* directory under the *assets* folder and add all the certificates into this folder.

> ***Important:*** Allow Bundled option will not work in Android 2.3.x OS versions due to certificate chaining issue (causes certificate exception). This is a known Android native issue. For more information, see Issue 25152 on the Android Open Source Project Issue Tracker. Among other topics, Issue 25152 documents that to make the bundled option work in Android 2.3.x devices, the root CA needs to be omitted from the server end.

- **Allow Pinned** - Only the certificates that are pinned or associated to the host. Pinning makes use of knowledge of the pre-existing relationship between the user and an organization or service to make the security-related decisions better.

g. In the **Install Location** section:

- **Use Location Preference**. This property defines the location where the application is deployed.
  - **Auto** - Indicates that the application is deployed on the device and can be moved to the SD Card later if required.
  - **Prefer SD Card** - Implies that the application is deployed on the storage card and cannot later be moved to the device memory.

> ***Note:*** This works only if the Minimum SDK is 2.3 or above.

h. **Signing** - Use this option to sign the android binary automatically during the build process.

- **Key Alias**- Use this option to enter the alias of the key.
- **Key Password** - Use this option to enter the password for the key.
- **Store File** - Use this option to locate and configure the store file.
- **Store Password** - Use this option to enter the password for the store.

i. **Splash Screen Orientation Mode** - When resource folders are created from IDE by **Add resource folders** option, then directories like drawable-port and drawable-land are created automatically inside the directory `<workspace>/<app>/resources/mobile/native/android`.

- **Portrait** - Use this option if splash screen support is required only for portrait mode, and copy it inside drawable-port folder.
- **Landscape** - Use this option if splash screen support is required only for landscape mode, and copy it inside drawable-land folder.
- **Both** - Use this option if splash screen support is required for both modes. If splash screen images are different images, then place the image in their respective directories. If you use same image for both modes, then copy the resources under `mobile > native > android` instead of drawable-port or drawable-land folder.

j. **ActionBar** - Enabled only if target SDK is 3.0 or above. Use this option to enable *Action Bar* feature.

k. **Support x86 Devices** - Select this option to support any Android-x86 devices.

l. **Support 32-bit Devices** - Select this option to build an Android APK with 32-bit support. Once you select this option, only 32-bit `.so` files (armeabi-v7a and x86) are packed and the application leaves out 64-bit `.so` files (arm64-v8a and x86_64). If you do not select the Support 32-bit devices option, 64-bit libraries / `.so` files (arm64-v8a and x86_64) are packed by default.

> ***Note:*** From Kony Visualizer V9 onwards, 64-bit APK's are generated, by default.

> ***Note:*** From August 1st 2019 onwards, all apps published on Google Play must support 64-bit architectures.

To support both 64-bit and 32-bit architectures in Google Play store, you must make sure that you perform the following actions:

- Build the 32-bit and 64-bit APKs with two different version codes, which are separated by at least 1000.
  For example, `64-bit version code = 32-bit version code + 1000`.
- Ensure that the third-party libraries (`AAR` files) contain the respective `.so` files in all supported architectures: lib/armeabi-v7a, lib/arm64-v8a, and lib/x86 lib/x86_64 for 32-bit and 64-bit architectures, respectively.

> ***Note:*** To enable your application to be built with both 32-bit and 64-bit native libraries, refer to Support for 32-bit and 64-bit Architectures in a Single APK.

m. **Support SQL DB Encryption (FIPS)**- In Android, if you select this option, Kony Visualizer automatically bundles Federal Information Processing Standard (FIPS) compliant SQL Cipher third-party library with the application. After the application is complied with this option selected, the APIs in Web SQL support database encryption. For more info, see **API Reference Guide > Offline Data Access APIs > Web SQL APIs.**

> ***Note:*** FIPS does not provide the mechanism to build the x86_64 architecture for Android. From V8 SP4 onwards, for the x86_64 architecture, Kony Visualizer automatically bundles libraries that are non-compliant to FIPS for the following items:
> SQLCipher
> OpenSSL

n. **Bundle OpenSSL Library**- In Android, if you select Bundle OpenSSL Library option, Kony Visualizer automatically bundles a third-party OpenSSL native library along with the application. The following APIs use this OpenSSL library to support additional hashing algorithms than the algorithms supported by Native Android SDK (Java Implementation). For information on supported algorithms in these APIs, see **API Reference Guide**

> Cryptography APIs.

i. kony.crypto.createPBKDF2Key

ii. kony.crypto.createHMacHash

> ***Note:*** FIPS does not provide the mechanism to build the x86_64 architecture for Android. From V8 SP4 onwards, for the x86_64 architecture, Kony Visualizer automatically bundles libraries that are non-compliant to FIPS for the following items:
> SQLCipher
> OpenSSL

o. **Disable Application Screenshot** - This option specifies whether the user can take a screenshot of your application.

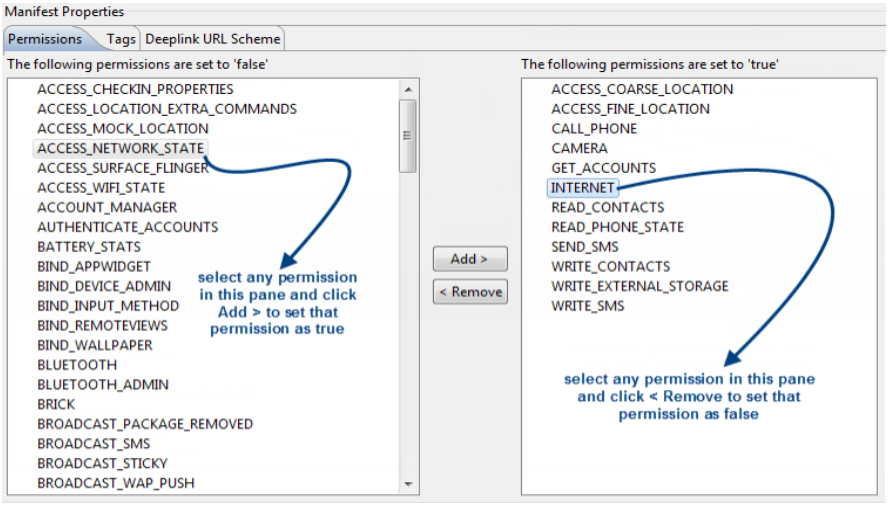
p. **Enable File Upload** - Enables you to upload files to a remote sever by using the HttpRequest API.

q. In **Manifest Properties** section:

- **Permissions** tab: Set the permissions to **true** or **false** based on the application requirements. Set the appropriate permissions for Android Manifest file. For more information, see The Android Manifest File.

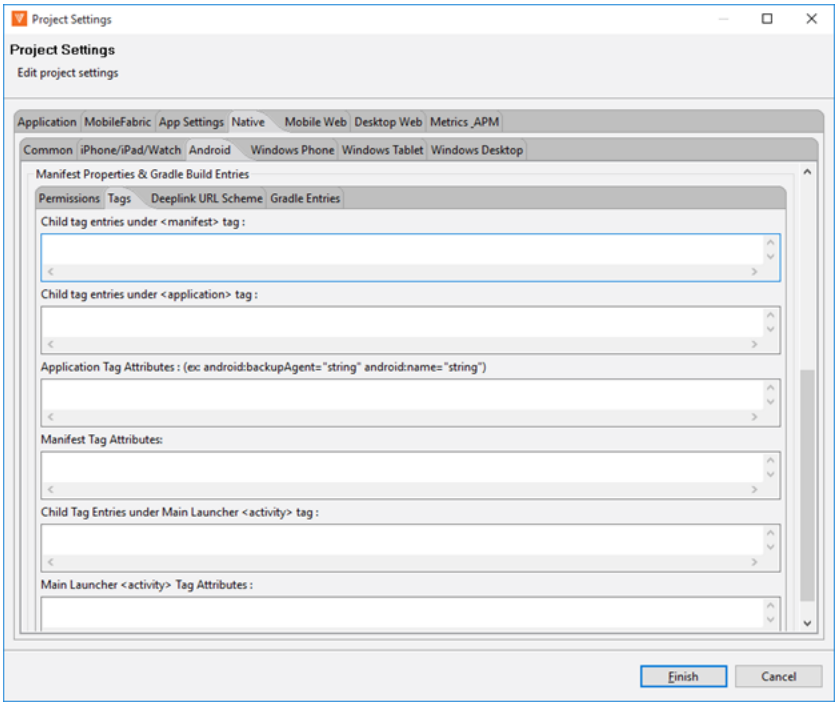
  i. To enable permissions, select the permissions from the left pane and click **Add >**.

  ii. To disable permissions, select the permissions from the right pane and click **< Remove**.



> ***Important:*** Add the `WRITE_EXTERNAL_STORAGE` setting if you need to save images in an external storage like SD Card.

- **Tags** tab: You can add tags to the Android manifest file directly from Kony Visualizer by specifying tag entries and attributes on the Tags tab. You can specify child tag entries and attributes for <manifest> and <application> tags, and the Main Launcher <activity> tag. For more information on the tags you can add with the manifest or application tags, see http://developer.android.com/guide/topics/manifest/manifest-intro.html.



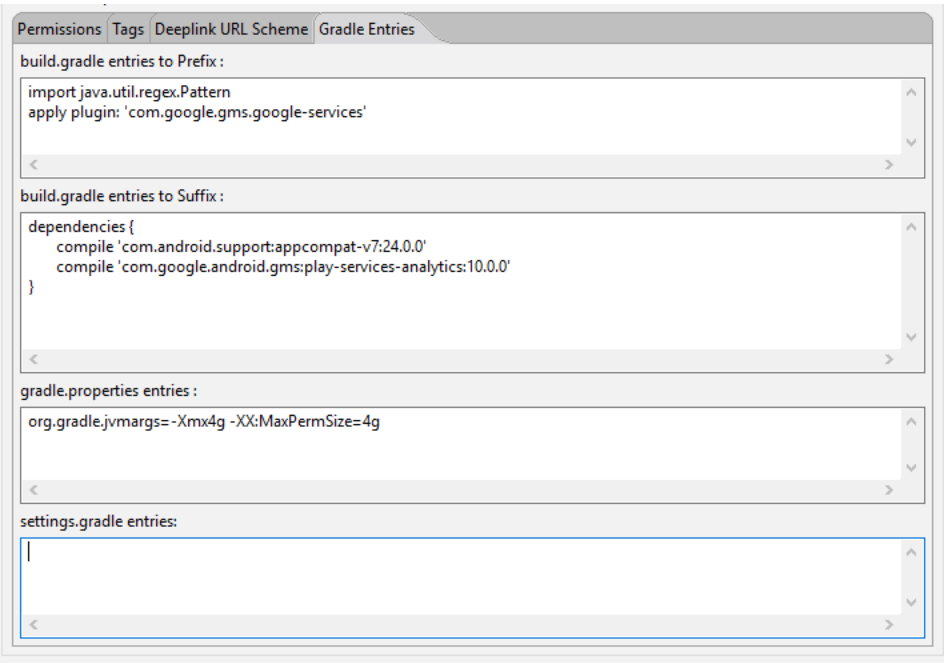
For more information on the tags you can add, see http://developer.android.com/guide/topics/manifest/manifest-intro.html.

- **Deeplink URL Scheme** tab: You can use the URL Scheme tab for Deeplinking. The values for Scheme/ port/ path/host/pathprefix/path pattern specified under this tab can be used to deep-link to a particular URL directly. For instance, if you set the masterdata for a browser widget to be a URL to deep-link, use the following format, `scheme://host:port/path`*or*`pathPrefix`*or*`pathPattern`. For more information on each of the values available under the URL Scheme tab, see http://developer.android.com/guide/topics/manifest/data-element.html#path.

  > ***Important:*** For a URL scheme in Android, please note that the scheme name should be in lowercase; otherwise the scheme name will not work in higher versions (Android 4.0 and above) or Android devices.

- **Gradle Entries** tab: You can use the Gradle Entries tab to import additional gradle packages, apply external plugins, or specify build-related configuration information,

build dependencies, or the location of any external repositories or modules used by your Android application.



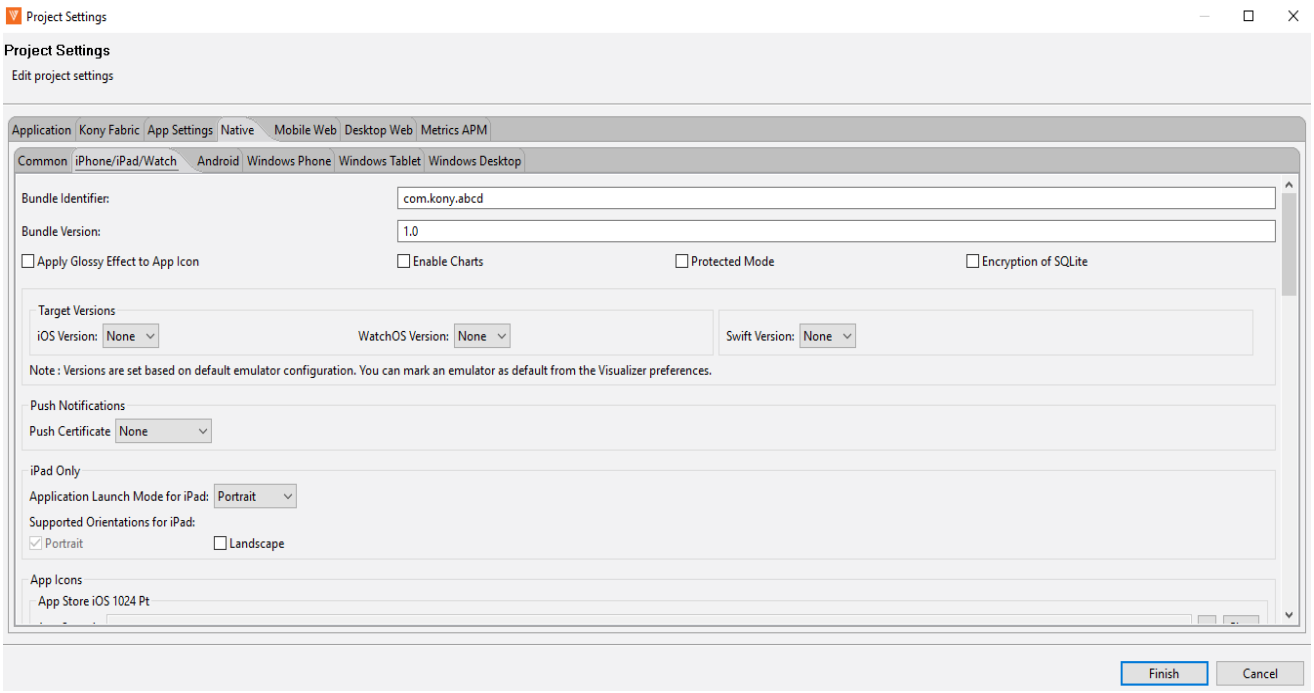
You can specify build.gradle entries as a prefix or suffix entries:

- Prefix entries are added just below any existing import statements in the build.gradle build script file. Use prefix entries to import additional gradle packages, or to specify external plugins to use in the build.
- Suffix entries are appended to the end of the generated build.gradle file. Use suffix entries to customize build logic; for example, to add compilation dependencies such as Google and Android support repositories, local library modules, or local and remote repository paths. For more information, see Organizing Build Logic.

Use the gradle.properties entry to configure project-wide Gradle settings, such as the Gradle daemon's maximum heap size or proxy settings. For more information, see The Build Environment.

Use the settings.gradle entry to specify external modules (Gradle-based third-party Android libraries) to include when building your application. For more information, see Configure Your Build.

5. Under **iPhone/iPad/Watch** tab, set the following properties:



a. **Bundle Identifier** - Provide an unique name that identifies the application bundle. This is usually in three parts and follows the convention of `com.kony.<appname>`.

b. **Bundle Version** - a number that identifies the version of the application bundle.

c. **Apply Glossy Effect to App Icon** - specifies if the glossy effect must be applied to the app icon. The default value is *false*.

d. **Enable Watch** - Select this option to enable wearable functionality in your application. If you select this option, wearable binaries are bundled with your application.

e. **Protected Mode** - Selecting this option ensures that your app is not run on a rooted/jail

broken device. To use this option, you have to patch Xcode with the Finalizer utility. For information on patching Xcode with Finalizer, see the Install Finalizer Package. This option works only if the application is built in Release mode. The Debug mode will generate unprotected binaries.

> ***Note:*** The Protected Mode option works only if the application is built in Release mode. To know more about protecting your application, refer Protect your app topic.

f. **Push Certificate** - Choose the required option from the list to either enable push notifications in different types of environments. You can also disable receiving push notifications for any environment. This field contains the following options:

- **development**: Select this option to receive push notifications when in the developer environment.
- **production**: Select this option to receive push notifications when in the production environment.
- **None**: Select this option to disable push notifications in any environment.

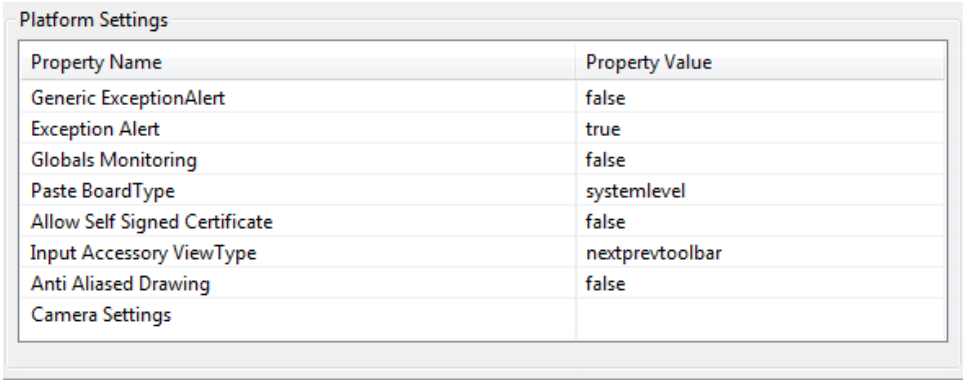
> ***Note:*** This feature is available from Kony Visualizer V8 SP4 FP19 onwards, and is available in both [[[Undefined variable MyVariables.ProdNameVizEnterprise]]] and [[[Undefined variable MyVariables.Kony QuantumViz]]].

g. **Application Launch Mode for iPad** - specifies the default mode of launching the application on iPad. *Portrait* is the default value.

h. **Supported Orientations for iPad** - specifies the supported orientations for the iPad. This depends on the launch mode. The different orientations for a form and at application level are listed at

i. **Deeplink URL Scheme**: specifies a url to which the application will deep-link to. If the application name is southwest then the url scheme that the other applications can use to launch the southwest application is *southwest://*. For more information about deep-linking, see Appendix E: the App Service Event.

- **Platform Settings**: Using the Platform Settings Area, you can set certain default properties for an application for iPhone.


Platform Settings

Property Name	Property Value
Generic ExceptionAlert	false
Exception Alert	true
Globals Monitoring	false
Paste BoardType	systemlevel
Allow Self Signed Certificate	false
Input Accessory ViewType	nextprevtoolbar
Anti Aliased Drawing	false
Camera Settings	

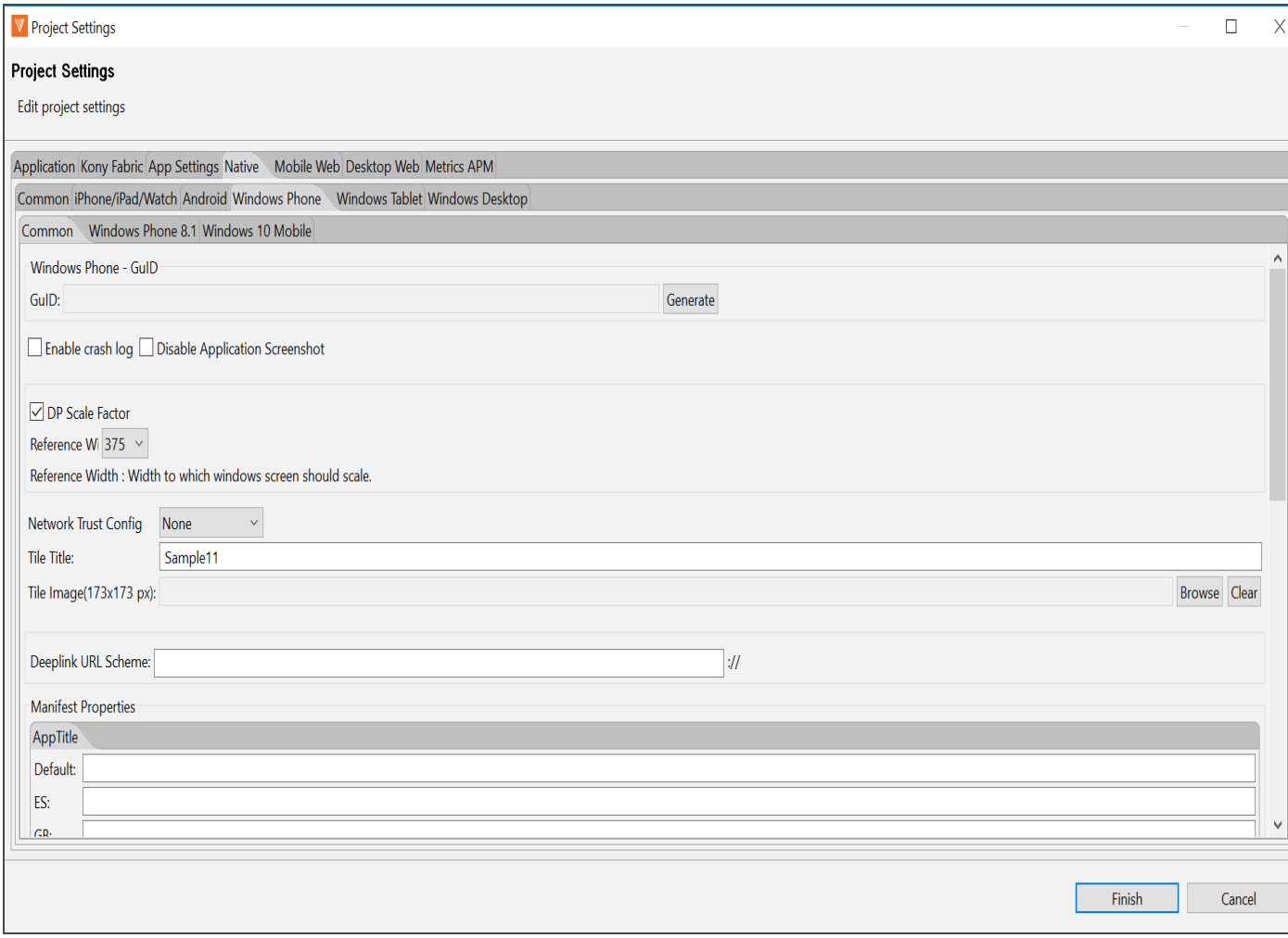
  - Generic exception alert: When true, generic exception alerts like "system error" are fired and when false detailed exception message is shown as alert. Best practice is to be use true for release mode and false for debug mode.
  - Exception alert: When true, system throughs the exception alerts otherwise (with false) app would crash in case of exception instead of alert. Best practice is to be use true for release mode and true for debug mode.
  - Globals monitoring: If set to true, the information like number of variables of the given type used in the app. Possible types are Strings, tables, numbers, closures, forms, other objects is printed in the logs.
  - Paste Board Type: It will allow the user to copy paste content from the app to external writable area like message etc. system level - it will allow to copy paste into other applications. Applevelpersistent - it will allow to copy paste within the app and the messages are persistent will be available across the app restarts. Applevelnonpersistent - it will allow to copy paste within the app and the messages are Not persistent and will not be available across the app restarts. Nopasteboard - it will not allow paste anywhere.
  - Allow Self Signed Certificate: By default it is false, if true, it allows self

signed certificate for development.

> ***Note:*** Self Signed Certificate option is only applicable if you use Network APIs in your application.

- Input Accessory View Type: The input accessory view type for widgets like text box, calendar, grouped widgets etc where you have next previous cancel buttons. This can be overridden by form level Input Accessory View Type.
- Anti Aliased Drawing: If set to true, allows smoother widgets and layout without any jagged edges.
- Camera Settings: Allows to set images to the icons which appear on the camera such as cancel icon, settings icon, tapanywhere.
- Backward_compatibility_mode: By default it is false, if true it will allow the application feature to behave as it would have behaved on earlier version (if there is any behavioral change in the latest version)

6. Click the **Windows Phone** sub-tab, and set the following properties:



   a. Under the **Common** sub-tab:

   b. In the **Capabilities** tab, set the permissions to true or false based on the application's requirements. For more information about each of the permissions refer Capabilities and requirements.

      - To enable permissions, select the permissions that are currently *false* and click **Add >**.

- To disable permissions, select the permissions that are currently *true* and click **< Remove**.

c. In the **Packaging** tab, set the following properties:

- **Package name:** A unique name to identify a specific application. It is generally in the format `domain.company.application`.
- **Package display name:** The name with which the application is submitted to Google play. This name is used to search the application in Google play.
- **Publisher name:** The publisher of the app. This default value is the name of the project. This attribute is required for certain types of apps, such as push-enabled apps.
- **Publisher display name:** Display name of the publisher.
- **Version:** Internal version number of the package.
- **Publisher ID:** Kony is the default Publisher ID of the project.

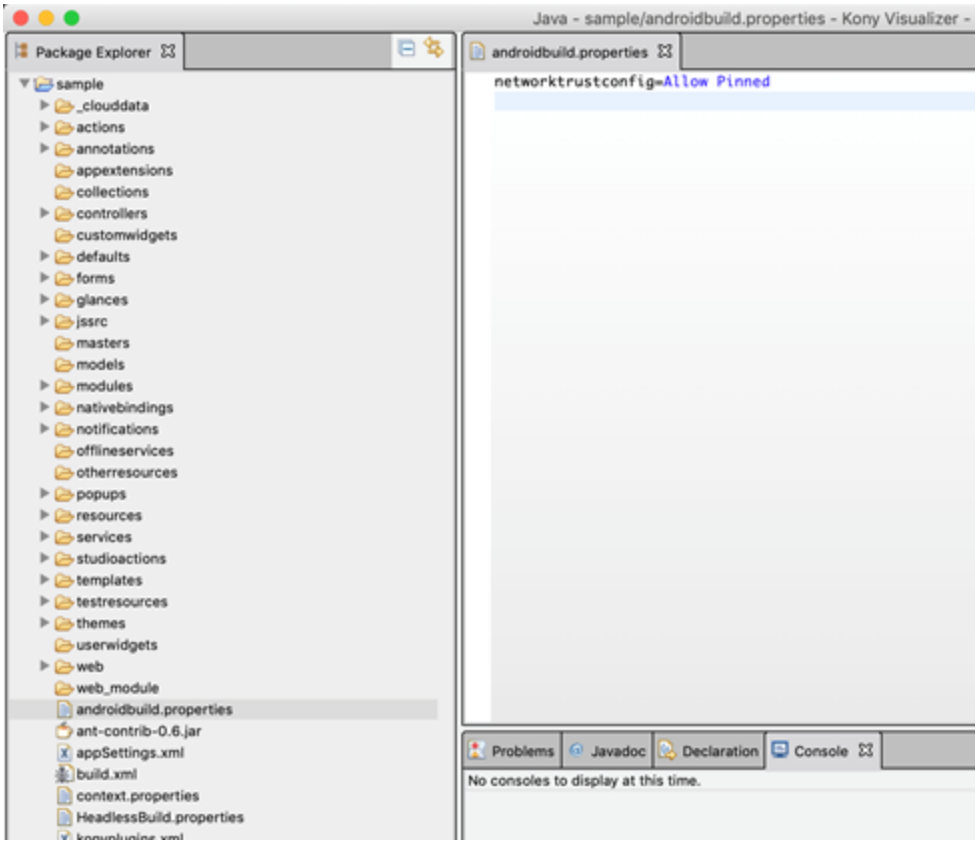
7. Click the **Windows Tablet** and **Windows Desktop** sub-tabs to set their respective properties.

   In Windows Desktop, if you select **Disable Application Minimize** feature, the application will not be allowed to minimize.

8. Click **Finish**.

Add Android Properties to androidbuild.properties File

From V8 SP4 onwards, you can enable or disable certain Android features by adding the corresponding properties to the `androidbuild.properties` file. You must first create the `androidbuild.properties` file in your Kony Visualizer project workspace (**workspaceLocation\\project**) as shown in the image.



> ***Note:*** If the same property is passed from Kony Visualizer in future plugins, the Kony Visualizer property takes precedence.

Enable Android Features

You can enable the following Android features by adding these properties in the androidbuild.properties file:

Public Key Pinning

You can enable the Public Key Pinning feature by specifying the following property in the androidbuild.properties file as shown.

This feature is enabled only if the `networktrustconfig` property is set as `Allow Pinned` in the androidbuild.properties file.

```
networktrustconfig = Allow Pinned
```

React Native

You must set both the following properties in the androidbuild.properties file as shown, to enable React Native feature support:

```
enableReactNative = true
```

Add the enableReactNative flag to enable ReactNative support.

```
reactNativeAppsList = "<reactNativeApp1>,<reactNativeApp2>..."
```

Specify the list of ReactNative apps (root folder name of ReactNative apps that are placed in the ReactNativeProjects folder) to be embedded into your Kony app.

APK Tamper Protection

The APK Tamper Protection feature helps you to verify if an APK has been tampered with (modified from its original version). If a tamper is detected, the application safely exits during the bootup process. This is an optional feature that is supported only in Release and Protected modes. APK Tamper Protection is available from V8 SP4 onwards. From V8 SP4 Fixpack 20 onwards, support for Google Play App Signing has been added. Google Play App Signing is a mandatory signup for Android App Bundle support.

This feature is enabled only if the `addAPKTamperProtection` property is set as `true` in the androidbuild.properties file.

```
addAPKTamperProtection = true
```

For this feature to work, you must provide either of the information as follows:

- Add KeyStore entries from the Signing section in Visualizer: **Project Settings > Native > Android > Mobile/Tablet > Signing**.
- Alternatively, you can add the developerSigningKeyHash key in the androidbuild.properties file. An example of a typical developerSigningKeyHash is shown here.

```
developerSigningKeyHash:2otpMeAC68KCm7Q+F48tzTFtzmU=
```

The developerSigningKeyHash key helps you to utilize the APK Tamper Protection feature in the following scenarios:

- Google Play App Signing, where the key used to sign the APK that is being uploaded is different from the final APK delivered to customers from Google Play. For more information on how the Google Play App Signing process works, click here.
- CI/Cloud build environment, without actually revealing the original developer signing KeyStore information.

Generate the developerSigningKeyHash Key for Google Play App Signing

1. When customers enroll into the Google Play App Signing process , the **SHA-256** or **SHA-1** hash of the public key can be obtained. They can obtain the hash by signing in to Google Play Store Console, navigating to **Release Management** > **App Signing**, and then copying the **SHA-256/SHA-1 certificate fingerprint**.
2. Go to `<WorkSpace>\temp\<AppID>\build\luaandroid\extres`.
3. Locate and open the `PrintApkSignatureHash.jar` file.
4. Run the following command to generate the developerSigningKeyHash:
   `java -jar PrintApkSignatureHash.jar --fingerprint "<Hash-Algorithm>: <certificate fingerprint in Hexadecimal>"`
   Here, `Hash-Algorithm` can either be `SHA-1` or `SHA-256`.
   For example:

```
java - jar PrintApkSignatureHash.jar--fingerprint‟ SHA256: EB: 71: 4E:
90: 3D: 2A: 7E: 14: 4B: D1: 73: 47: 3A: EA: 3D: 06: C5: F2: 69: B5:
DC: BB: 28: 44: A0: 8D: AC: 17: E7: F2: 7F: 8F”
```

5. Here is a sample output that is generated.

```
developerSigningKeyHash : xxxxxxxxxxxxxxxxx
```

6. Copy and paste the output value in the `androidbuild.properties` file.

Important Points

a. If you specify KeyStore entries through Kony Visualizer, the KeyStore entries will take precedence over the developerSigningKeyHash key until Kony Visualizer V8 SP4 Fixpack 19.

b. From Kony Visualizer V8 SP4 Fixpack 20 onwards, if you specify Visualizer KeyStore entries and developerSigningKeyHash, both items are respected simultaneously and the app is launched if any one of these items matches. This enhancement helps you to test the APK locally before uploading it to Google Play.

c. If you do not want to upload the Upload Signing key to the CI cloud, use the `uploadSigningKeyHash` property in the **additionalbuild.properties** file as an alternative to specifying KeyStore entries.

d. You must provide either the KeyStore or uploadSigningKeyHash to test Google Play App Signing locally for the enrolled APK, which is tamper-protected. If you do not provide any of those values, the test APK that is generated will be signed by the debug key and the APK will not boot as the hash validation process fails at run time.

Generate the developerSigningKeyHash Key by using the KeyStore File

To manually sign the application by using your own keystore file, follow these steps:

1. Go to `<WorkSpace>\temp\<AppID>\build\luaandroid\extres`.
2. Locate and open the `PrintApkSignatureHash.jar` file.
3. Run either of the following commands to generate the developerSigningKeyHash:
   - For apps built in Kony Visualizer V8 SP4 Fixpack 19 or earlier, use the following command. This command generates the hash with SHA-1 algorithm.
     ```
     java -jar PrintApkSignatureHash.jar keyStorePath keyStorePassword keyAlias
     ```
   - For apps built in Kony Visualizer V8 SP4 Fixpack 20 or later, use the following command. This command generates the hash with either SHA-1 or SHA-256 algorithm, depending on the `-- algorithm` input parameter.
     ```
     java -jar PrintApkSignatureHash.jar --storepath keyStorePath
     ```

```
--storepass keyStorePassword --alias keyAlias --algorithm
hash-logo
```

Here, the items are as follows:

- keyStorePath: Path to your actual developer signing key, which is used to upload your app's APK to the Google Play Store.
- keyStorePassword: Password of your developer KeyStore.
- keyAlias: Signing key alias of your developer KeyStore.
- hash-algo: Hashing algorithm that is used to generate the signing key hash. It can either be `SHA-1` or `SHA-256`.

> ***Note:*** If the hash is generated with SHA-256 algorithm, the hash will not work for apps built in Kony Visualizer V8 SP4 Fixpack 19 or earlier. However, if the hash is generated with SHA-1 or SHA-256 algorithm, the hash will work for apps built in Kony Visualizer V8 SP4 Fixpack 20 or later.

4. Here is a sample output that is generated.

```
developerSigningKeyHash : xxxxxxxxxxxxxxxxx
```

> ***Note:*** If you are generating the hash of the upload signing key to support Google Play App Signing, use `uploadSigningKeyHash` as the key instead of `developerSigningKeyHash`.

5. Copy and paste the output value in the `androidbuild.properties` file.

Support for 32-bit and 64-bit Architectures in a Single APK

From V9 onwards, when you enable the `support64bit` property in the `androidbuild.properties` file and select the **Support 32-bit Devices** check box from **Project Settings > Native > Android**, the Kony Android build generates a `Fat` application. This `Fat` application supports all architectures: armeabi-v7a, x86, arm64-v8a, and x86_64.

```
support64bit = true
```

Note: Kony recommends that you use a `Fat` binary for testing purposes only, and Kony does not recommend you to upload a `Fat` binary to Google Play. Use either the Split APK feature or the Android App Bundle feature to reduce the size of the binary that is downloaded to customers' devices.

Split APKs based on Supported Architecture

Bundling all architectures into a single fat APK increases the APK size of the app that is delivered to customers. This APK Splitting feature, which has been introduced from V8 SP4 Fixpack 12 GA onwards, helps you to decrease the APK size that is downloaded based on the target platform architecture. For more information on the splitting of APKs, click here.

You must enable the `splitapks` property as `true` in the androidbuild.properties file. This action generates the architecture-specific individual .apk files and universal .apk file that supports all the architectures. This approach reduces the size of the .apk file.

```
splitapks = true
```

Note: Based on the specified value of the support64bit property in the androidbuild.properties file as well as on the selection of the Support x86 Devices and Support 32-bit Devices check boxes in the Kony Visualizer Project Settings, the Kony Android build generates a set of APKs, each with a single supported architecture.

- Architecture-specific .apk files are generated with this naming convention: `<appid>-<architecture>-<buildtype>.apk`
  Consider an app with appid as **KonySample**, for which the 64-bit ARM APK names for Debug and Release modes are as follows:
  - Debug mode: **KonySample-arm64-v8a-debug.apk**
  - Release Unsigned mode: **KonySample-arm64-v8a-release-unsigned.apk**
  - Release Signed mode: **KonySample-arm64-v8a-release-signed.apk**

- Universal .apk files with all selected architectures are generated with this naming convention: `<appid>-universal-<buildtype>.apk`
  Consider an app with appid as **KonySample**, for which the APK names for Debug and Release modes are as follows:
  - Debug mode: **KonySample-universal-debug.apk**
  - Release Unsigned mode: **KonySample-universal-release-unsigned.apk**
  - Release Signed mode: **KonySample-universal-release-signed.apk**

The APKs are generated under the following paths:

- For Mobile: **<workspace>\temp\<appid folder>\build\luaandroid\dist\<appid folder>\build\outputs\apk** folder.
- For Tablet: **<workspace>\temp\<appid folder>\build\luatabandroid\dist\<appid folder>\build\outputs\apk** folder.

> ***Note:*** For architecture-specific .apk files, the version code of the individual .apk file must be unique. This is because since both 32-bit and 64-bit APKs are supported in 64-bit devices, there will be a conflict while choosing the APK file. Specifying a higher version code for the 64-bit APK file results in a greater precedence, and thus the 64-bit APK file would be chosen for a 64-bit device. This process helps to leverage the higher performance of a 64-bit APK in a 64-bit device.

The Kony Android `build.gradle` file automatically handles the use cases related to APK versioning in the following manner:

1. The Kony Android `build.gradle` file assigns **architecture codes** in the following order of priority:
   'armeabi-v7a':1 < 'x86':2 < 'arm64-v8a':3 < 'x86_64':4
2. The final Google Play version code of the individual .apk file is: **{(architecture code) * 1000} + (version code from project settings)**
   For example, if the version code in the Kony Visualizer Project Settings is 3. Then, the Google Play version codes of each architecture APK is as follows:

- armeabi-v7a: **1003** {(1 * 1000) +3 }
- x86 : **2003** {(2 * 1000) +3 }
- arm64-v8a : **3003** {(3 * 1000) +3 }
- x86_64 : **4003** {(4 * 1000) +3 }

> ***Note:*** If you want to split APKs based on density along with architecture, customize the build by adding the appropriate `build.gradle` entries in the Gradle Entries tab > build.gradle entries to Suffix section.

Generate Android App Bundle

Google Play's Dynamic Delivery feature uses your Android App Bundle to build and serve APKs that are optimized for each device configuration. This results in a smaller app download for customers by removing unused code and resources needed for other devices. The support for Android App Bundle generation has been added from V8 SP4 Fixpack 12 GA onwards. For more information about Android App Bundle, click here.

To enable this feature, you must set the `generateAppBundle` property as `true` in the androidbuild.properties file. The Kony Android build then generates the binary in App Bundle (aab) format. The AAB file is a Google Play upload format file, and is not an installable file.

> ***Note:*** Kony Visualizer and CI builds in Kony Visualizer do not provide support for the generation of binaries in the Android App Bundle (aab) format. You must configure Kony Visualizer to support the generation of 32-bit and 64-bit architectures in a single APK for generating the Android App Bundle.

```
generateAppBundle = true
```

The AAB file is generated in the following paths after the build:

- For Mobile : **<workspace>\temp\<appid folder>\build\luaandroid\dist\<appid folder>\build\outputs\bundle** folder.

- For Tablet : **<workspace>\temp\<appid folder>\build\luatabandroid\dist\<appid folder>\build\outputs\bundle** folder.

> ***Note:*** The Kony Android build generates an AAB file with all the selected architectures. The selected architectures are based on the specified value of the support64bit property in the androidbuild.properties file as well as on the selection of the Support x86 Devices and Support 32-bit Devices check boxes from the Kony Visualizer Project Settings.

To test how Google Play uses the AAB file to generate APKs and how those APKs behave when deployed to a device, follow these steps:

1. Download bundletool.

   > ***Note:*** Android provides a tool, named **bundletool**, to extract APKs from the AAB file and test them. For more information on the usage of bundletool, click here.

2. Extract the **APKs set** file from the aab file by using this command:
   ```
   java -jar bundletool.jar build-apks --bundle=<app-id>.aab --output=<app-id>.apks --overwrite
   ```
   Here, **APKs set** is an archive file with the extension as **.apks**.

3. The following command fetches the configuration details of the connected device and installs the device-compatible set of APKs:
   ```
   java -jar bundletool.jar install-apks --apks=<app-id>.apks --device-id=serial-id
   ```
   Here, **--device-id**: Optional parameter that helps to install the device-compatible set of APKs to a specific device (identified by **serial-id**), when multiple devices are connected.
   serial-id: Device identifier that the **adb devices** command returns.

> ***Note:*** If you want to estimate the download size of the APKs from Google Play, generate the device-specific set of APKs by following the procedure specified here.

As part of the V8 SP4 Fixpack 12 GA release, the Dynamic Delivery feature (on-demand delivery of modules when an app developer requests) is not supported for Kony Framework libraries. You can, however, implement the Dynamic Delivery feature for third-party libraries by using FFI code.

To sign the App Bundle file and ensure that the file is ready to be uploaded to Google Play, follow these steps

1. It is mandatory that you enroll into the Google Play App Signing process for you to be able to upload your App Bundle to the Play Console. Otherwise, you cannot upload the App Bundle to Google Play. For more information on how the Google Play app signing process works, click here.
2. The Release build generates a signed App Bundle with the Release Key value, which is specified in Kony Visualizer. This Release Key is usually either the Upload Key (if you have already enrolled for the Google Play App Signing process) or the actual Release Key (if you are opting for the Google Play App Signing process for the first time to submit an update in the form of an App Bundle to the existing app in Google Play). The signed App Bundle can then be uploaded to Google Play. While delivering optimized APKs to the device, Google Play signs the APKs with its own app signing key.
3. If the Release Key value is not specified in Kony Visualizer, the Release build generates an unsigned App Bundle. The **jarsigner** command can be used to sign the aab file. For further information on jarsigner, click here.

   ```
   jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
   <keystorefile>
   <app bundle file> alias_name
   ```

   > ***Note:*** apksigner is not supported to sign the App Bundle.

4. In Debug build, Gradle automatically signs the App Bundle with the Android SDK Debug Key.

Bundle a Customized Cordova-Generated Android Project

To bundle the manually customized version of your Cordova-generated Android project, you must set the `cordovabuildmode` property as `incremental` in the androidbuild.properties file. This feature is available from Kony Visualizer V8 SP4 Fixpack 47.

```
cordovabuildmode = incremental
```

For more information about how to manually customize the Cordova-generated Android project, click here.

The Android Manifest File

Applies to *Kony Visualizer Classic*.

When you build an Android app by using Kony Visualizer, an `AndroidManifest.xml` file is created in the app's corresponding `dist` folder. The file is located at *WorkspaceName>/temp/<AppName>/build/luaandroid/dist*. The manifest file provides essential information about your app to the Android operating system, and Google Play store.

The Android manifest file helps to declare the permissions that an app must have to access data from other apps. The Android manifest file also specifies the app's package name that helps the Android SDK while building the app. The Android manifest file provides information such as activities, services, broadcast receivers, and content providers of an android application.

With Kony Visualizer, you can define the following options in the `AndroidManifest.xml` file:

- Supported screen sizes
- Supported SDK versions: minimum, target, and maximum
- Ability to send Push Notifications
- Various permissions for the application

To modify the `Android Manifest.xml` file from Visualizer, follow these steps:

1. From the Project Explorer, click **Project Settings**. The **Project Settings** window appears.
2. Click the **Native** tab.
3. Click the **Android** sub-tab, and then scroll down to the **Manifest Properties & Gradle Entries** section.
4. Configure the Permissions, Tags, and Deeplink URL scheme tabs. More information on how to

to configure manifest properties such as Permissions, Tags,and Deeplink URL in the Android Manifest file. Here is an example of a basic Android Manifest file generated with default permissions.

Permissions

An app must have certain permissions to access data from the other apps. By default, Kony Visualizer enables and disables certain permissions in the `AndroidManifest.xml` file. When you build an application, an `AndroidManifest.xml` file is automatically generated for the app. This manifest file will contain permissions based on how you have configured those permissions. If you have not specified any permissions explicitly, default permissions would apply.

For more information on the `AndroidManifest.xml` file, refer App Manifest on the Android Developer site.

The following permissions are set to true and added by default:

- ACCESS_NETWORK_STATE
- INTERNET
- READ_PHONE_STATE

These permissions are set to false by default. You can add the permissions according to the requirements.

ACCESS_COARSE_LOCATION	ACCESS_FINE_LOCATION	ACCESS_LOCATION_EXTRA_COMMANDS

ACCESS_MOCK_ LOCATION	ACCESS_SURFACE_ FLINGER	ACCESS_WIFI_STATE
ACCOUNT_MANAGER	AUTHENTICATE_ ACCOUNTS	BATTERY_STATS
BIND_APPWIDGET	BIND_DEVICE_ADMIN	BIND_INPUT_METHOD
BIND_REMOTEVIEWS	BIND_WALLPAPER	BLUETOOTH
BLUETOOTH_ADMIN	BRICK	BROADCAST_PACKAGE_ REMOVED
BROADCAST_STICKY	BROADCAST_WAP_PUSH	CALL_PHONE
CALL_PRIVILEGED	CAMERA	CHANGE_COMPONENT_ ENABLED_STATE
CHANGE_ CONFIGURATION	CHANGE_NETWORK_ STATE	CHANGE_WIFI_MULTICAST_ STATE
CHANGE_WIFI_STATE	CLEAR_APP_CACHE	CLEAR_APP_USER_DATA
CONTROL_LOCATION_ UPDATES	DELETE_CACHE_FILES	DELETE_PACKAGES
DEVICE_POWER	DIAGNOSTIC	DISABLE_KEYGUARD
DUMP	EXPAND_STATUS_BAR	FACTORY_TEST
FLASHLIGHT	FORCE_BACK	GET_ACCOUNTS
GET_PACKAGE_SIZE	GET_TASKS	GLOBAL_SEARCH
HARDWARE_TEST	INJECT_EVENTS	INSTALL_LOCATION_ PROVIDER

INSTALL_PACKAGES	INTERNAL_SYSTEM_ WINDOW	KILL_BACKGROUND_ PROCESSES
MANAGE_ACCOUNTS	MANAGE_APP_TOKENS	MASTER_CLEAR
MODIFY_AUDIO_ SETTINGS	MODIFY_PHONE_STATE	MOUNT_FORMAT_ FILESYSTEMS
MOUNT_UNMOUNT_ FILESYSTEMS	NFC	PERSISTENT_ACTIVITY
PROCESS_OUTGOING_ CALLS	READ_CALENDAR	READ_CONTACTS
READ_FRAME_BUFFER	READ_HISTORY_ BOOKMARKS	READ_INPUT_STATE
READ_LOGS	READ_SMS	READ_SYNC_SETTINGS
READ_SYNC_STATS	REBOOT	RECEIVE_BOOT_COMPLETED
RECEIVE_MMS	RECEIVE_SMS	RECEIVE_WAP_PUSH
RECORD_AUDIO	REORDER_TASKS	RESTART_PACKAGES
SEND_SMS	SET_ACTIVITY_WATCHER	SET_ALARM
SET_ALWAYS_FINISH	SET_ANIMATION_SCALE	SET_DEBUG_APP
SET_ORIENTATION	SET_PREFERRED_ APPLICATIONS	SET_PROCESS_LIMIT
SET_TIME	SET_TIME_ZONE	SET_WALLPAPER
SET_WALLPAPER_HINTS	SIGNAL_PERSISTENT_ PROCESSES	STATUS_BAR

SUBSCRIBED_FEEDS_READ	SUBSCRIBED_FEEDS_WRITE	SYSTEM_ALERT_WINDOW
UPDATE_DEVICE_STATS	USE_CREDENTIALS	USE_SIP
VIBRATE	WAKE_LOCK	WRITE_APN_SETTINGS
WRITE_CALENDAR	WRITE_CONTACTS	WRITE_EXTERNAL_STORAGE
WRITE_GSERVICES	WRITE_HISTORY_BOOKMARKS	WRITE_SECURE_SETTINGS
WRITE_SETTINGS	WRITE_SMS	WRITE_SYNC_SETTINGS

Set Android Manifest Permissions

You can modify the permissions in the `AndroidManifest.xml` file based on the requirements of the application.

To set the permissions in the Android Manifest file, follow these steps:

1. In Kony Visualizer, from the Project Explorer, click **Project Settings**. The **Project Settings** window appears.
2. Click the **Native** tab.
3. Click the **Android** sub-tab and then scroll down to the **Manifest Properties & Gradle Entries** section.
4. To enable permissions, select the permissions from the left pane and click **Add >**.
   For example, If you need to save images to an external storage device such as USB drive or SD card, add the WRITE_EXTERNAL_STORAGE setting.

> ***Note:*** To select multiple permissions, hold the **Ctrl** key and click the permissions. To select continuously listed permissions, hold the **Shift** key and click the permissions.

5. To disable permissions, select the permissions from the right pane and click **< Remove**.
6. Click **Finish**.
   When you build the app, Kony Visualizer generates the manifest file with the permissions that you specified.
   - If the app lists *normal* permissions in its manifest (permissions that don't pose risk to the user's privacy or the device's operation), the system automatically grants those permissions to the app.
   - If the app lists *dangerous* permissions in its manifest (permissions that could potentially affect the user's privacy or the device's normal operation), the app must explicitly request those permissions.

For more information on requesting runtime permissions, refer Runtime Permissions.

For more information on the list of dangerous permissions, refer Dangerous Permissions List.

Tags

Apart from the permissions, you can also configure tags such as application and activity. Following is a list of the manifest tags that can be configured from Visualizer:

- Child Tag entries under <manifest> tag
- Child Tag entries under <application> tag
- Application Tag Attributes (ex: android:name=”string”)
- Manifest Tag Attributes
- Child Tag Entries under Main Launcher <activity> tag
- Main Launcher <activity> Tag Attributes

For example, manifest attributes can be modified in the Android manifest file as follows.

```
<manifest xmlns: android =
"http://schemas.android.com/apk/res/android"
xmlns: tools = "http://schemas.android.com/tools"
package = "com.orgname.Sample"
android: versionCode = "1"
android: versionName = "1.0.0" //Manifest attributes>
```

For more information on Android Manifest tags and their attributes, refer Manifest Elements

Deeplink URL Scheme

You can use the URL Scheme tab for Deeplinking. The values for Scheme/ port/ path/host/pathprefix/path pattern specified under the Deeplink URL tab can be used to deep-link to a particular URL directly.

For example, if a scheme is defined as https and a host is defined as www.example.com, the following entry will be added to the `Android manifest.xml` file under intent-filter tag of default activity.

```
<intent-filter> <data android: scheme = "https"
android: host = "www.example.com"/> </intent-filter>
```

Android Manifest Example

The following is a basic `AndroidManifest.xml` file that is generated with default permissions.

```
<? xml version = "1.0"
encoding = "utf-8" ?>
//Manifest attributes

< manifest
xmlns: android = "http://schemas.android.com/apk/res/android"
xmlns: tools = "http://schemas.android.com/tools"
package = "com.orgname.Sample"
android: versionCode = "1"
```

```
android: versionName = "1.0.0" >
//Application attributes are added here

< application
android: name = "com.konylabs.android.KonyApplication"
android: icon = "@drawable/Sample_icon"
android: label = "@string/app_name"
tools: remove = "supportsRtl"
tools: replace = "icon" >
//Activity attributes are added here

< activity
android: name = ".Sample"
android: configChanges =
"locale|keyboardHidden|orientation|screenSize|screenLayout"
android: label = "@string/app_name"
android: launchMode = "singleTask"
android: screenOrientation = "sensor"
android: theme = "@style/Theme.AppCompat.NoActionBar"
android: windowSoftInputMode = "adjustResize" > < intent - filter > <
action android: name = "android.intent.action.MAIN" / > < category
android: name = "android.intent.category.LAUNCHER" / > < /intent-
filter>
            <intent-filter>
                <action android:name="android.intent.action.SEARCH" /
> < category android: name = "android.intent.category.DEFAULT" / > <
/intent-filter>
            <meta-data
                android:name="android.app.searchable"
                android:resource="@xml/searchable
" /> //Deeplink text is added here
//Add Child Tag Entries under Main Launcher here if required
```

```
        </activity>
        <meta-data
            android:name="
android.app.default_searchable "
            android:value=".KonyMain " />
        <provider
            android:name="
com.orgname.Sample.SampleSearchSuggestionProvider "
            android:authorities="
com.orgname.Sample.SampleSuggestionProvider " />
    </application>
    //supported screens
    <supports-screens
        android:anyDensity="
true "
        android:largeScreens="
true "
        android:normalScreens="
true "
        android:smallScreens="
true "
        android:xlargeScreens="
true " />
       //These are the default permissions set as true
    // Add permissions here

    <uses-permission android:name="
android.permission.READ_PHONE_STATE " />
    <uses-permission android:name="
android.permission.ACCESS_NETWORK_STATE " />
    <uses-permission android:name="
android.permission.INTERNET " />
```

```
       //Add the Manifest child tags here if required


</manifest>
```

Add Global Variables

You add global variables to your project using the Global Variables dialog box. The variables added here are accessible globally, that is, they are accessible across:

- All the containers in the application
- Code modules
- The Action Editor
- The Mapping Editor

Alternatively, you can also add variables in the Action Editor as a part of the action items. The variables added in the Action Editor are local variables and are available only to the action items defined from that point onwards.

> ***Notes:***
> - All the global variables (except Pre Appinit) defined in the Global Variables dialog box are accessible in the code modules, but global variables that are defined in the code modules are not accessible from the Global Variables dialog box.
> - To ensure the best performance possible for your application, you should use global variables only if necessary. Be sure to delete any global variables that your app doesn't use.
> - Variables are application-related. After you log out from the application, the variables are no longer available.

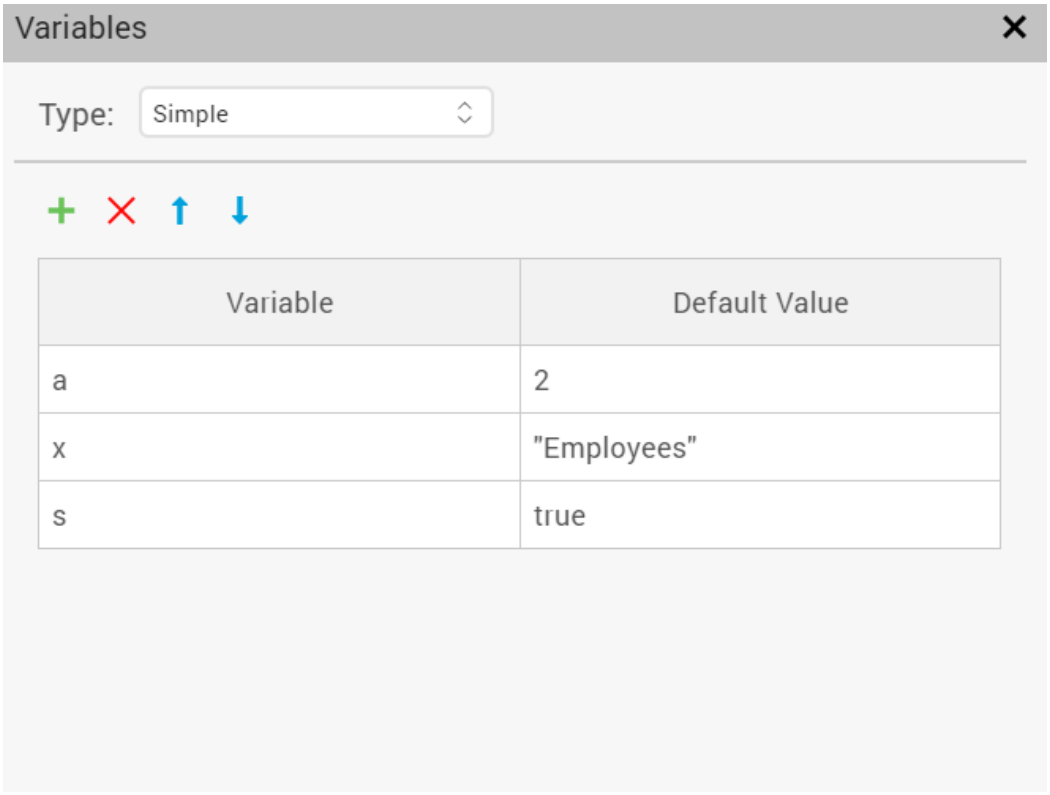
Global Variables are of two types:

- **Simple.** A variable that contains a single record.
- **Collection.** A variable that contains multiple records. You can add as many records as you want to a collection.

Add a Simple Global Variable

To add a Simple global variable, follow these steps:

1. In Kony Visualizer, point to the **Edit** menu, and then click **Global Variables**. The **Variables** dialog box opens.
2. From the **Type** list box, select **Simple**. The Simple variable can either be a String, Number, or a Boolean value. You must provide the String variable details in quotes.
3. In the **Variable** column field, type the name of the variable. You must enter a name that is unique to the application and one that is not used for another variable.
4. In the **Default Value** column field, type the initial value that you want the variable to hold when the app is first opened.
5. To add a new row for another variable, click the **+** icon. To delete an added variable, select the row and click the red **X** icon. If you want to modify the order of the variables, click the upper and lower arrow icons as required.
6. Repeat steps 3 through 5 until you have added all the Simple variables that you want.
7. Close the Variables dialog box by clicking the gray **X** icon at the upper-right corner of the dialog box.

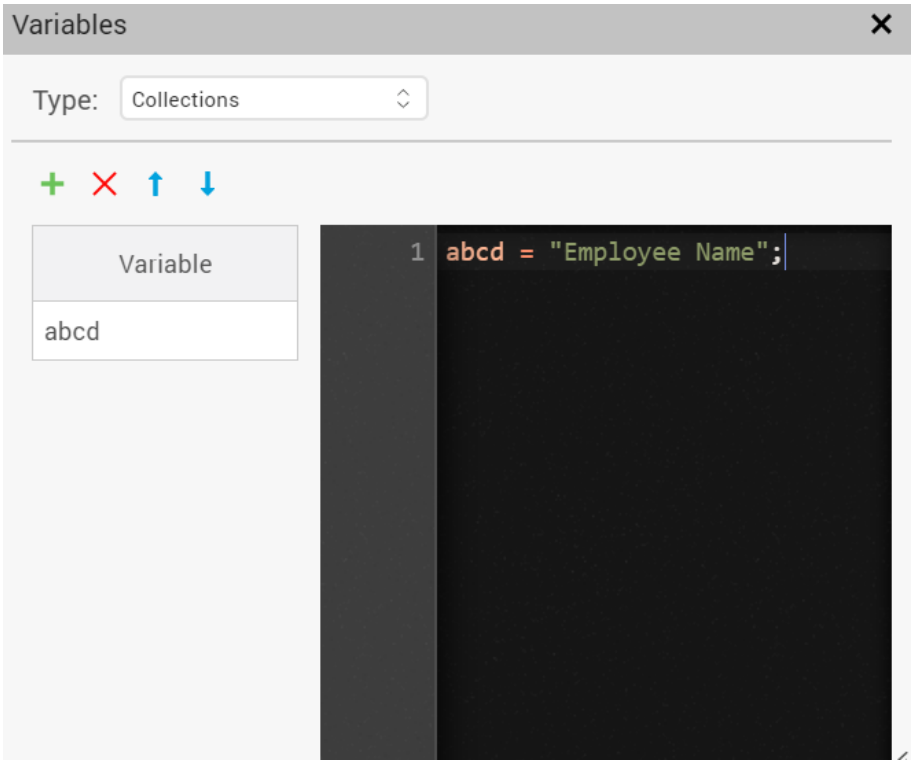


Add a Collection Global Variable

To add a Collection global variable, do the following:

1. In Kony Visualizer, point to the **Edit** menu, and then click **Global Variables**. The **Variables** dialog box opens.
2. From the **Type** list box, select **Collections**.
3. In the **Variable** column field, type the name of the variable. You must enter a name that is unique to the application and one that is not used for another variable.
4. In the right column, enter the assignment statements or function calls that you want in JSON format.

5. To add a new row for another variable, click the **+** icon. To delete an added variable, select the row and click the red **X** icon. If you want to modify the order of the variables, click the upper and lower arrow icons as required.
6. Repeat steps 3 through 5 until you have added all the Collection variables that you want.
7. Close the Variables dialog box by clicking the gray **X** icon at the upper-right corner of the dialog box.



Invoke Sublime Text from Kony Visualizer

From V8 SP4, you can invoke the latest version of Sublime Text (a third-party source code editor) from within Kony Visualizer. This feature ensures that developers have the option to use an alternative tool to write and modify code within Visualizer, in addition to Visualizer's in-built code editor. You can also make use of the auto-complete Intellisense feature for Kony UI, API, and SDK functions while working with Sublime Text.

Sublime Text is a proprietary cross-platform source code editor with a Python API. This source code editor natively supports various programming and markup languages. You can use Sublime Text to add functions with plugins, which are typically community-built and maintained under free-software licenses.

Prerequisites

Before you can start using Sublime Text from Visualizer, you must meet the following prerequisites:

- Use Kony Visualizer V8 SP4
- Install the latest version of Sublime Text
- Install Package Control
- Install Ternjs for Sublime Text (to add Intellisense for Kony APIs)

Enable Sublime Text

After you have met the prerequisites mentioned previously, you must enable the feature to invoke Sublime Text from Visualizer.

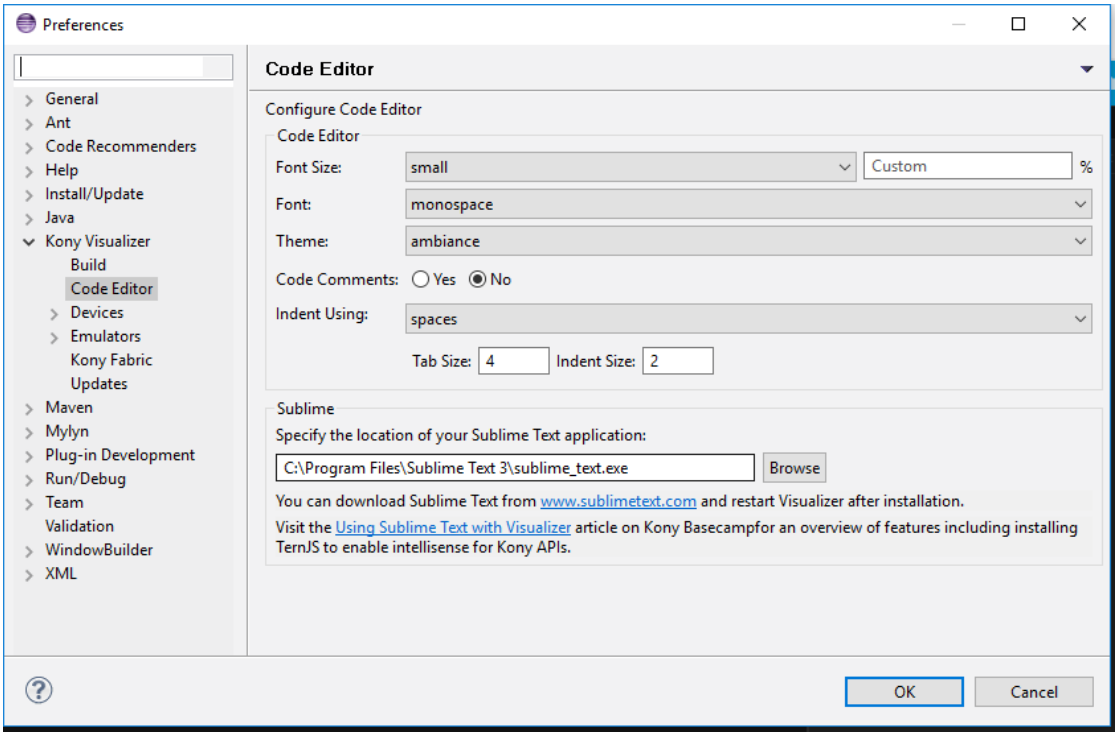
> ***Note:*** By default, Visualizer will recognize Sublime Text if it is installed at a default location.

In Kony Visualizer Classic

To enable the feature in Kony Visualizer Classic, follow these steps:

1. In Kony Visualizer Classic, do the following:
   - **For Windows**: On the main menu, click **Window** and then click **Preferences**.
   - **For Mac**: On the main menu, click **Kony Visualizer** and then click **Preferences**.

   The **Preferences** window appears.
2. From the left pane, expand **Kony Visualizer** and click **Code Editor**. The Code Editor section appears.
3. Under the **Sublime** section, click **Browse** and select the absolute file path where you installed Sublime Text in your local system.
   For example, *C:\Program Files\Sublime Text 3\sublime_text.exe*.
   For more information, click the Using Sublime Text with Visualizer link. You will be navigated to the Kony Basecamp article on using Sublime Text.



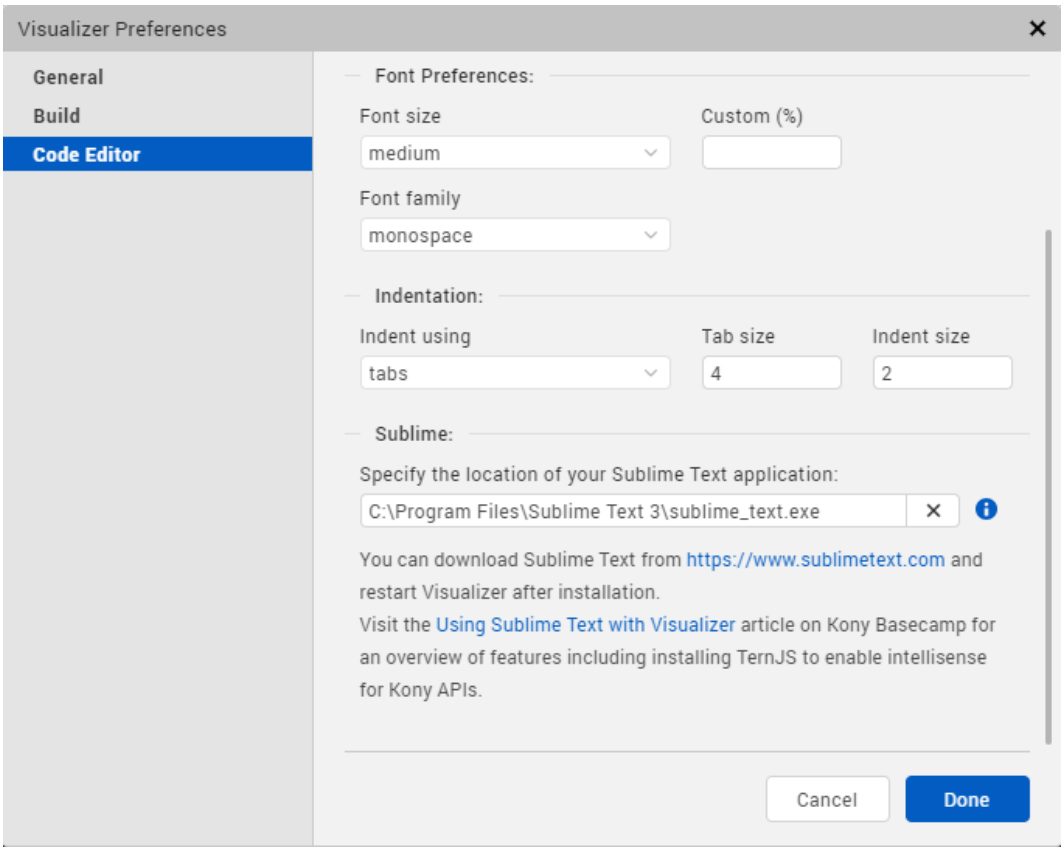
4. Click **OK**. The Sublime Text feature is enabled.

For Kony Visualizer

To enable the feature for Kony Visualizer, follow these steps:

1. In Kony Visualizer, do the following:
   - **For Windows**: On the main menu, click **Edit** and then click **Preferences**.
   - **For Mac**: On the main menu, click **Kony Visualizer** and then click **Preferences**.

   The **Visualizer Preferences** window appears.
2. From the left pane, click **Code Editor**. The Code Editor section appears.
3. Under the **Sublime** section, click **Browse** and select the absolute file path where you installed Sublime Text in your local system.
   For example, *C:\Program Files\Sublime Text 3\sublime_text.exe*.
   For more information, click the Using Sublime Text with Visualizer link. You will be navigated to the Kony Basecamp article on using Sublime Text.



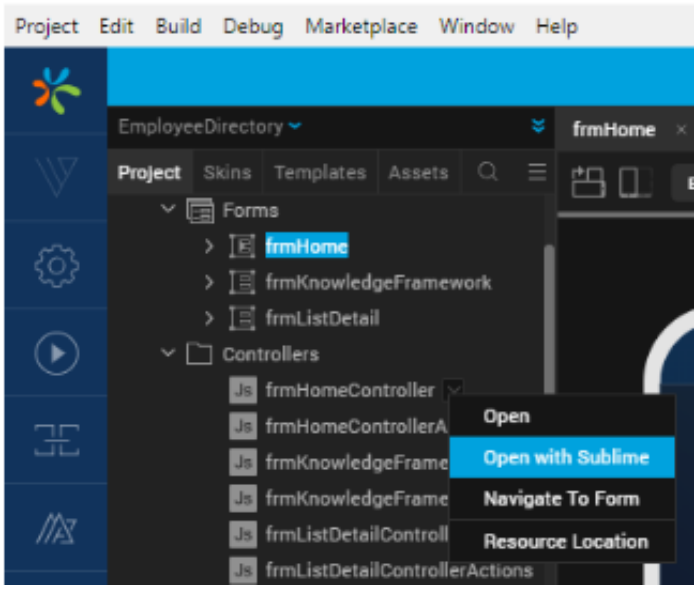
4. Click **Done**. The Sublime Text feature is enabled.

Open a JavaScript File with Sublime Text

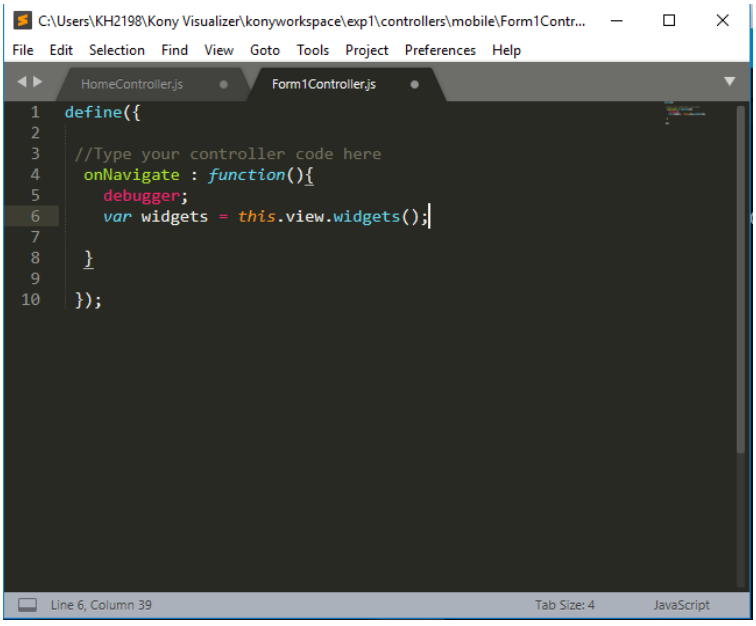
Now that you have enabled the required option, you can open any JavaScript file in Sublime Text from Visualizer.

To do so, follow these steps:

1. In your Kony Visualizer project, in the Project Explorer, right-click a JavaScript file/folder. A context menu appears.



2. Click **Open with Sublime**. The Sublime Text source code editor appears with the code snippet of that particular JavaScript file/folder.



You can make any changes to the code in any JavaScript file/folder by using Sublime Text.

Add Intellisense for Kony APIs

You can add the auto-complete Intellisense feature for Kony UI, API, and SDK functions while working with Sublime Text from Visualizer. You must install the Ternjs for Sublime package to enable this feature. Ternjs is a package for Sublime Text that provides JavaScript auto-fill intelligence.

To install Ternjs, follow these steps:

1. Install the Ternjs for Sublime package from GitHub.
2. Check out the following code into a sub-directory of your Sublime Text's Packages directory.

```
cd /path/to/sublime-text-N/Packages
git clone https://github.com/ternjs/tern_for_sublime.git
```

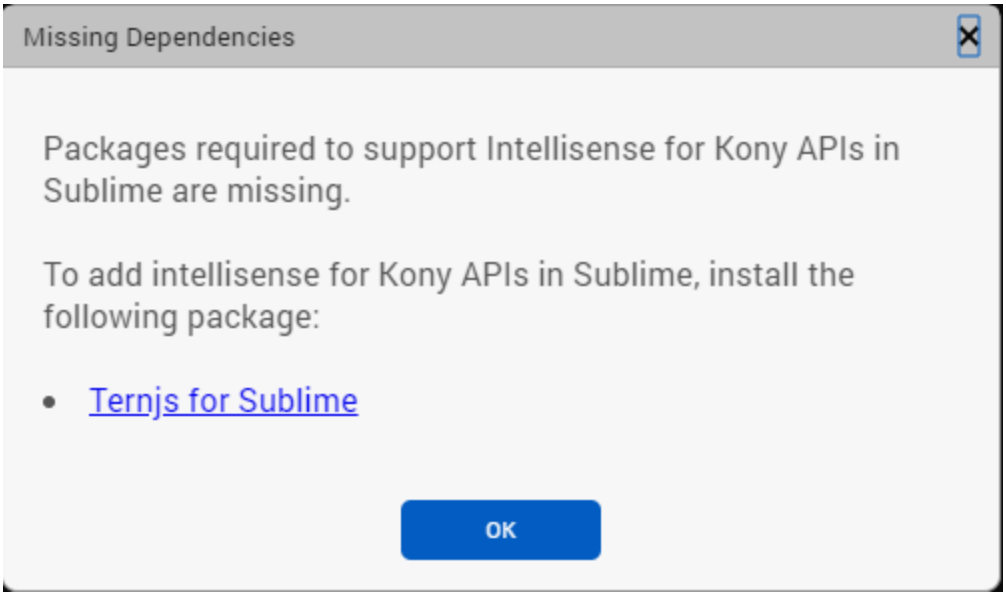
3. Ensure that you have installed node.js and npm (Tern is a JavaScript program), and then install the dependencies of the package.

```
cd tern_for_sublime
npm install
```

> ***Note:*** On OS X, you might also need to install the Fix Mac Path Sublime plugin to help Sublime Text to locate your node binary.

Once you successfully install the Ternjs for Sublime package, the Kony JavaScript auto-complete feature appears as you start typing in the Sublime Text editor.

> ***Note:*** If you have not installed the Ternjs for Sublime package, the **Missing Dependencies** window appears asking you to install that package.



Find and Replace

With Kony Visualizer, you can find and also replace instances of code and text in your modules, assets, and action sequences.

To find and replace data, do the following:

1. On the **Edit** menu, click **Find/Replace** (**Search** in *Kony Visualizer*). The Search tab opens in the Console.
2. If doing so aids your search, click one or more of the four available search options. They are:
    - Regular expression, which interprets your query as written using regular expression notation
    - Case-sensitive
    - Whole word
    - Wrap

3. In the Find text box, type what you want to search for. As you type, the search results are immediately returned.

    - To open the file where a particular instance of the query resides, click the search result. To highlight that particular instance in its file, click the search result again.
    - To clear your query, hover over the far right end of the text box, and then click the X that appears.
    - To highlight the next instance of your query in the search results, click the Next arrow.
    - To highlight the previous instance of your query in the search results, click the Previous arrow.

4. To replace a query, type the replacing data in the Replace With text box, and then click **Replace**. To replace all instances, click **Replace All**. To undo a change, click in the file that was modified, and then press **Ctrl+Z**, or click **Undo** from the **Edit** menu. To undo additional replacements, repeat. You can also undo changes by closing the file without saving the changes, although this will also lose any unsaved changes you made to the file before the find and replace operation.

    > ***Note:*** You cannot undo a Replace All operation.

Capture Product Requirements with Review Notes

Using the Review Notes feature in Kony Visualizer, you can capture feedback from users who are evaluating your app design. Such requirements capturing helps ensure that the design of your app successfully meets the requirements of potential users. The Review Notes feature supports rich text formatting such as font type and size, paragraph alignment, numbered and bulleted lists, block quotes, and even tables. There's no size limit to the notes, and you can carry on conversation threads with as many users as want to participate. A note can be associated with a widget, a form, or even an entire project, and when you add a note, a numbered badge is associated with that element of the app's user

interface on the Visualizer Canvas so that you can easily see what parts of the app have been commented on. You can turn the badges on and off so that they don't interfere with your previewing the app, and comments can be marked as either New or Resolved so that you can keep track of what's been acted upon.

> ***Note:*** To use Review Notes among multiple people, you need to share the project. For information on how to do so, see Share a Project.

For information on using Review Notes, click any of the following topics:

Add a Review Note

To add a Review Note in Kony Visualizer, do the following:

1. On the Project Explorer, click the Project tab.
2. On the Project tab, select the item you want to add a note to. This item is now the item of focus on the Visualizer Canvas.
3. Right-click the item, click **Add Note**, and then click in the text field that displays. Alternately, you can click the item's Review tab on the Properties pane, and then click **Notes**.
4. Format text by using the rich text controls located above the note's text field. Some text may need to be selected first before you apply certain kinds of formatting.
5. When you're finished, click anywhere outside the text field. A badge is assigned to the note, which appears in the upper right corner of the item on the Visualizer Canvas. Widget badges are orange with white numbers while form badges are blue with a white uppercase "F".

Format a Review Note

You format the text of a Review Note by using the rich text controls located above the note's text field. Some text may need to be selected first before you apply certain kinds of formatting. The following table lists the kinds of formatting available.

Type of Formatting	Description
Font Name	Formats the selected text or text that follows the insertion point with the font typeface that you select from the Font drop-down list. The available typefaces are as follows: Arial, *Comic Sans MS*, `Courier New`, Georgia, Lucida Sans Unicode, Tahoma, Times New Roman, Trebuchet MS, and Verdana.
Font Size	Formats the selected text or text that follows the insertion point with the font size that you select from the Font Size drop-down list. The available font sizes range from 8 point to 72 point.
Font Formatting	Formats the selected text or text that follows the insertion point with the font formatting that you select. The available font formatting includes Bold, Italic, Underlined, Font Color, and Font Background Color.
Alignment	Formats the paragraph that the insertion point is currently in with the Alignment that you select. The available paragraph alignment options include Left-aligned, Centered, Right-aligned, and Justified.

Lists	Formats the paragraph that the insertion point is currently in with the List Type that you select. You can create either a numbered list or a bulleted list.
Indent Level	Formats the paragraph that the insertion point is currently in with the Indent Level that you select. You can either increase the Indent Level of a paragraph or decrease it.
Block Quote	Formats the paragraph that the insertion point is currently in as a Block Quote, which sets it off uniquely from the text around it. The Block Quote button acts as a toggle, so to remove the Block Quote formatting, click the Block Quote button again.
Remove Formatting	Removes paragraph-level formatting from the paragraph that the insertion point is currently in, or removes font-level formatting from the selected text.
Table	Opens the Table Properties dialog box where you specify the number of rows and columns you want the table to have. You can also specify the percentage of the width of the text field you want the table to take up, specify the height, and indicate whether the table has a header, as well as setting the alignment, the cell spacing, and the cell padding. You can also create a caption for the table and a summary.

Undo and Redo an Action

You can undo the most recent instance of typing in the Review Note field, and redo the last action that you undid. To do so, do the following:

- To undo your last Review Note action, click the **Edit** menu, and then click **Undo**.
- To redo the Review Note action that you just undid, click the **Edit** menu, and then click **Redo**.

Review and Reply to Review Notes

With Review Notes, multiple users can review a project and provide input. You can also add your own comments to an existing note.

View Multiple Review Notes at Once

To view multiple Review Notes at once, do the following:

1. On the Project Explorer, click the Project tab.
2. On the Project tab, select the parent that contains the items that have Review Notes. For instance, if a number of widgets have Review Notes associated with them, click the parent of those widgets.
3. On the Properties Pane, click the Review tab, and then click **Notes**.

   All the Review Notes for the widgets in the parent you selected are displayed.

Edit Review Notes

To edit a Review Note, do the following:

1. On the **App** menu, click the Badge of the Review Note you want to edit. The note displays on the Review tab of the Properties pane.
2. Click the text field of the note. Doing so displays the rich text controls, and the insertion point begins blinking in the text field, indicating that you can begin editing.

Search Review Notes

To search Review Notes, do the following:

1. On the Project Explorer, click the Project tab.
2. On the Project tab, select the parent that contains the items that have Review Notes. For instance, if a number of widgets have Review Notes associated with them, click the parent of those widgets.
3. On the Properties Pane, click the Review tab, and then click **Notes**.
4. At the top of the Review tab, in the Search box, type the text you want to search for.

   All text that matches your search term or phrase is highlighted in yellow in the Review Notes.

Expand and Collapse a Review Note

As the number of Review Notes increases and each note grows in length, the list can become unwieldy. To mitigate this, you can collapse individual Review Notes so that only their title bar displays.

- To collapse a Review Note, click the minus icon located near the right edge of the Review Note's title bar. The icon changes from a minus to a plus.
- To expand a collapsed Review Note, click the plus icon located near the right edge of the Review Note's title bar.

Reorder Review Notes

You can change the order that the Review Notes are listed in so that you can move more important notes up and less important notes down.

To reorder review notes, do the following:

1. Display multiple Review Notes at once by following the instructions for View Multiple Review Notes at Once.
2. Hover the cursor over the title bar of the Review Note you want to move until the cursor becomes a four-headed arrow.
3. Click and drag the Review Note to a new location in the order of Review Notes.

Hide and Display Review Note Badges

There may be instances when you do not want to display the Review Notes Badges, such as when you are using the Functional Preview feature, or are simply wanting to see the app as a user would see it. To accommodate this need, you can turn off Review Notes Badges.

To turn Review Notes Badges on and off, do the following:

- To turn off Review Notes Badges, on the toolbar of the Visualizer Canvas, click the Badges switch so that the toggle is on the left side of the switch.

- To turn on Review Notes Badges, on the toolbar of the Visualizer Canvas, click the Badges switch so that the toggle is on the right side of the switch.

Delete a Review Note

To delete a Review Note, click the trashcan icon located at the right edge of the Review Note's title bar.

Additional Information about Review Notes

The following are additional facts about Review Notes.

- Each widget can have only one note, but that note can have as many replies as are needed.
- If you need more room to display a note, you can increase the width of the Properties pane.
- When you fork a form, all Review Notes of that form are also added to the forked form. There is no linkage back to the original form's Review Notes.
- If a project containing notes is imported into Kony Visualizer, these notes appear on the Review tab of the Properties pane.
- If a component or master has a Review Note, that one note applies regardless of what platform or channel you select for that component or master; changing the platform or channel does not change the note.
- When a widget with a Review Note is added to a collection, the Review Note associated with the widget is not copied into the collection.

Add Comments to Forms

Using the Comments feature in Kony Visualizer, you can add comments to forms to keep track of changes you've made, or changes you're considering. The Comments feature is available only at the form level, and only for the Mobile and Tablet channels.

For information on using comments, click any of the following topics:

Add a Comment

Add a Comment

To add a comment in Kony Visualizer, do the following:

1. On the Project Explorer, click the **Project** tab.
2. On the **Project** tab, select the form you want to add a comment to from either the Mobile or Tablet channel. This form is now the item of focus on the Visualizer Canvas.
3. In the Properties Editor, click the **Review** tab, and then click **Comments**.
4. In the text box at the bottom of the Comments pane, type a comment, and then click **Add**.

Edit a Comment

To edit a Comment, do the following:

1. Select the form whose comment or comments you want to edit. This form is now the item of focus on the Visualizer Canvas.
2. In the Properties Editor, click the **Review** tab, and then click **Comments**. Comments that have been added to the form are listed.
3. Click the pencil icon of the comment you want to edit. The comment populates the text box at the bottom of the Comments pane.
4. Make your edits to the comment, and then click **Save**. Alternately, if you want to discard the changes you've made to the comment, click **Cancel**.

Delete a Comment

To delete a comment, click the X located to the right of the comment's time and date stamp.

Time and Effort Savers

Kony Visualizer includes a number of features designed to help you make the most of work you've already done.

Keyboard Shortcuts

Display Widget Command Handles

Select Multiple Items

Undo and Redo an Action

Copy and Paste a Color or Gradient

Jump to the Definition of a Code Element

Open a Resource's Folder

Copy between Designer and Developer Actions

Copy and Paste Forms and Actions across Channels

Display a List of JavaScript Code Elements

Expand and Contract the Visualizer Canvas

Go to the Most Recent Edit Location

Jump to a Specific Line in a JavaScript File

Keyboard Shortcuts

The following keyboard shortcuts are available in Kony Visualizer.

Mac Key	Windows Key	Function
Control + Click	Right-click	Opens a widget's context menu.
⌘ + -	Ctrl + -	Decreases the Code Editor font size and becomes the new preferred size in the Preferences dialog box.
⌘ + [	Ctrl + [	Moves the selection down one layer in the z-index.
⌘ +]	Ctrl +]	Moves the selection up one layer in the z-index.
⌘ + =	Ctrl + =	Increases the Code Editor font size and becomes the new preferred size in the Preferences dialog box.
⌘ + ;	Ctrl + ;	Toggles the horizontal and vertical rulers on and off when working on a Desktop channel form on the Visualizer Canvas.
⌘ + A	Ctrl + A	In the context of a selected form or flex container, selects all the child widgets.
⌘ + A	Ctrl + A	In the context of the Code Editor, selects all code.
⌘ + Alt + O	Ctrl + Alt + O	Opens the Resources folder.
⌘ + Alt + T	Ctrl + Alt + T	Opens the Build folder.
⌘ + Alt + W R	Ctrl + Alt + W R	Opens the WebApp Native folder (applicable only in Kony Visualizer Classic).
⌘ + Alt + W T	Ctrl + Alt + W T	Opens the WebApp Mobile App folder (applicable only in Kony Visualizer Classic).
⌘ + B	Ctrl + B	Available only in Kony Visualizer Classic. Initiates the Build Last command, which uses the most recently-selected platform(s) and channel(s).
⌘ + C	Ctrl + C	Copies the selected widget(s).
⌘ + Shift + C	Ctrl + Shift + C	Copies the selected widget's skin.
⌘ + D	Ctrl + D	Duplicates the selected widget(s).
⌘ + Delete	Delete	Deletes the selected widget(s).
⌘ + Drag	N/A	Constrains the movement of the selected widgets to either the horizontal or vertical axis.
⌘ + F	Ctrl + F	When the Code Editor is active, opens the Code Editor Find and Replace feature. In other contexts, opens the project Search text box in the Project Explorer.

Mac Key	Windows Key	Function
F2	F2	When used, invokes the rename function. User can modify the name of the asset to a new name. The rename shortcut supprts the following assets: • Widgets (including forked forms, masters, notifications, glances) • Skins • Components • Modules • MVC Extensions • Web apps (including preview) • Test resources • Named actions • User widget modules • App extension JS files Renaming a splash screen, app events, controller files, custom resources, model files, external files, and read-only files is not supported.
⌘ + Alt + F	Ctrl + Alt + F	Opens the project Search text box in the Project Explorer.
⌘ + Shift + F	Ctrl + Shift + F	Opens the Search tab in the Console for conducting global searches.
⌘ + H	Ctrl + H	Hides the selected widgets.
⌘ + Shift + H	Ctrl + Shift + H	Unhides the selected widgets.
Ctrl + L	Ctrl + L	When used in the context of the Code Editor, opens a command bar for jumping to a specific line number.
⌘ + L	Ctrl + L	When used in the context of the Visualizer Canvas, locks the selected widgets.

Mac Key	Windows Key	Function
⌘ + Shift + L	Ctrl + Shift + L	Unlocks the selected widgets.
⌘ + O	Ctrl +O	Used within an open JavaScript file in the Code Editor. Displays a list of the code elements within the JavaScript file, including objects, functions, arrays, numbers, and strings. Double-clicking a code element in the list navigates to that code element.
Alt + Q	Ctrl + Q	Navigates to the previously edited module.
⌘ + R	Ctrl + R	Initiates the Run Last command, building a functional preview of the app using the most recently-used platform and channel selections.
⌘ + Shift + R	Ctrl +Shift + R	Available only in Kony Visualizer Classic. Initiates the Run with the Kony Visualizer Classic Events command, building a functional preview using the Developer events mode.
⌘ + S	Ctrl + S	Saves the currently selected project entity (e.g. form).
⌘ + Shift + S	Ctrl + Shift + S	Saves the current project.
⌘ + Shift + <	Ctrl + Shift + <	Decreases the font size of the selected widget(s) by 10 units
⌘ + Shift + >	Ctrl + Shift + >	Increases the font size of the selected widget(s) by 10 units
⌘ + U	Ctrl + U	Un-parents the selected widget from its parent container, making it a child to the parent of the container it was un-parented from.
⌘ + V	Ctrl + V	Pastes the copied widget(s).
⌘ + Shift + V	Ctrl + Shift + V	Pastes the skin that is on the clipboard to the selected widget.
⌘ + W	Ctrl + W	Closes the item currently open on the Visualizer Canvas.
⌘ + X	Ctrl + X	Cuts the selected widget(s).
⌘ + Shift + Z	Ctrl + Y	Repeats the last action.
⌘ + Z	Ctrl + Z	Undoes the last action.
Alt + Drag	Alt + Drag	Changes the Snap mode to function to opposite of what is configured.
Down Arrow	Down Arrow	Moves the selected item(s) down by one unit.
Esc	Esc	Depending on the context, the Esc key cancels the current move or resize action on the Visualizer Canvas, or closes the search panel (if open) in the Code Editor window.

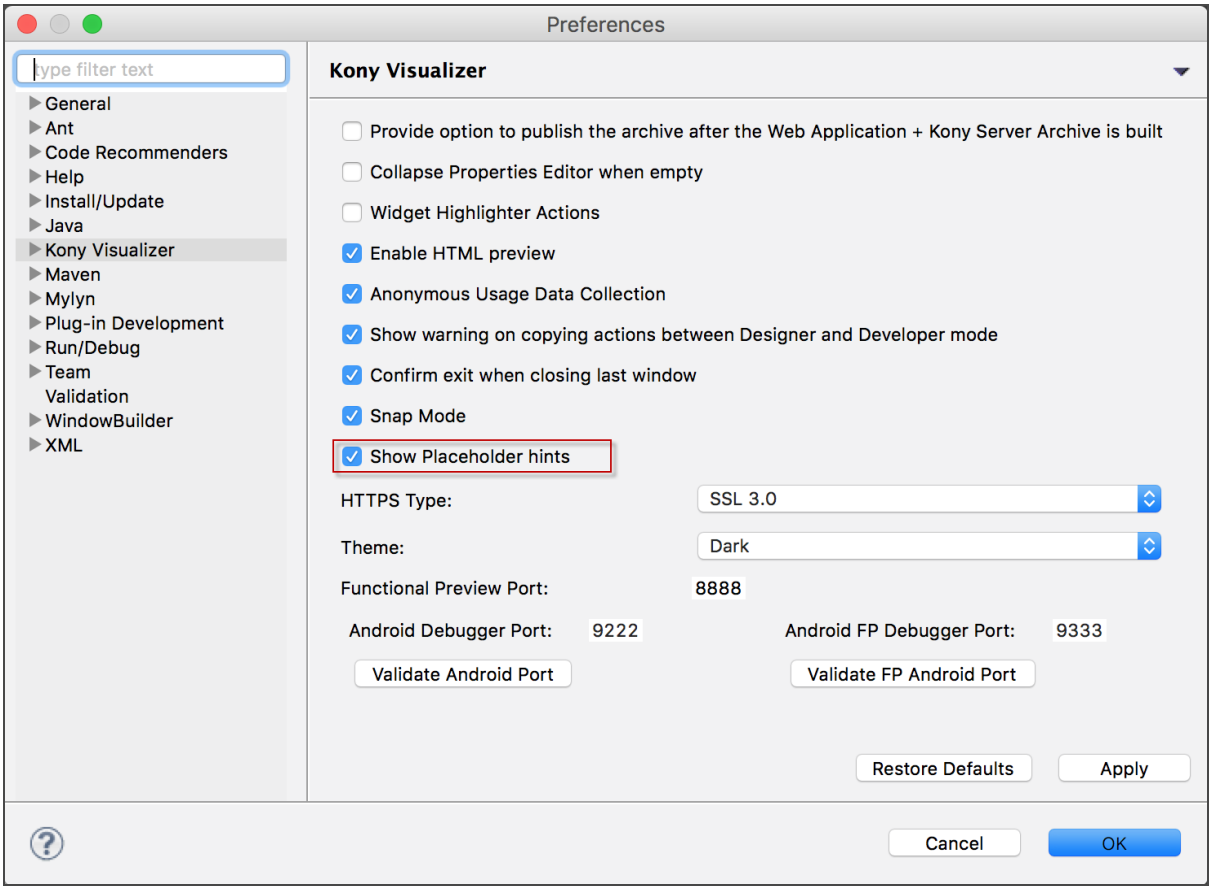
Mac Key	Windows Key	Function
Left Arrow	Left Arrow	Moves the selected item(s) to the left by one unit.
Option + '	Alt + '	Selects the right sibling.
Option + ;	Alt + ;	Selects the left sibling.
Option + [	Alt + [	Selects the parent container.
Option +]	Alt +]	Selects the first child widget.
Option + Drag	Ctrl + Drag	Disables grid snapping, widget snapping, and widget guides while moving or dragging the selection on the Visualizer Canvas.
Option + N	Alt + N	Creates a new project.
Option + Shift + V	Alt + Shift + V	Assigns the skin in the clipboard to the selected widget.
Right Arrow	Right Arrow	Moves the selected item(s) to the right by one unit.
Shift + Arrow	Shift + Arrow	Moves the selected item(s) in the direction of the arrow by 10 units.
Shift + Drag	Shift + Drag	Disables grid snapping, widget snapping, and widget guides while moving or dragging the selection along the given axis.
Shift + Resize	Shift + Resize	Constrains the aspect ratio while scaling the selected widget. Applicable only if scaling handles are selected.
Space bar + mouse motion	Space bar + mouse motion	Moves the canvas around in the BVR mode.
Up Arrow	Up Arrow	Moves the selected item(s) up by one unit.

Using Placeholder Text

Placeholder text is a feature that is introduced in Kony Visualizer V8 SP3 GA release. Placeholders on forms and segments help a user to identify surfaces and possible actions for drag and drop in Visualizer.

To turn on or off the Placeholder text, do the following:

1. In Kony Visualizer, open the project in which you want to make a change to the placeholder setting.
2. From the File menu, navigate to **Windows** > **Preferences** > **Kony Visualizer**.
3. Select or unselect **Show Placeholder Hints**.



The feature is turned on or off based on your selection.

Use Rulers and Guides for the Desktop Channel

For Desktop Web channel forms, you can display horizontal and vertical rulers to help you accurately position widgets. The rulers are available at a zoom level of 75% and higher. Guides are lines you pull from either ruler and position on the canvas to help you align widgets with one another. You can move guides, or lock them so that they're immovable, unlock them again, hide them, and remove them.

Rulers and guides are only available for the Desktop Web channel; they are not available for the Mobile, Tablet, and Watch channels.

For more information, click any of the following procedures:

Rulers

Show or Hide Rulers

Guides

Add a Guide

Reposition a Guide

Lock and Unlock Guides

Hide or Show Guides

Clear a Guide

Clear All Guides

Show or Hide Rulers

Horizontal and vertical rulers are available on the Visualizer Canvas for forms in the Desktop channel. If your work requires, you can hide the rulers, and also show them again.

To hide or show the rulers, do the following:

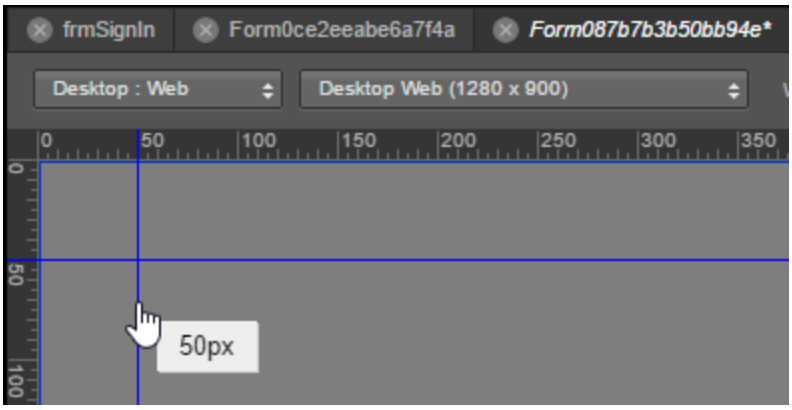
1. In the upper right corner of the Visualizer Canvas, click the round option button  , hover over **Ruler**, and then click **Hide Ruler**. The horizontal and vertical rulers become hidden.
2. To show the rulers again, click the round option button, and then click **Show Ruler**.

> ***Notes:***
> - You can also toggle rulers to show or hide by pressing **Ctrl+;**
> - For Desktop channel forms, the rulers display only at a zoom level of 75% and higher.

Add a Guide

To add a guide to your canvas, do the following:

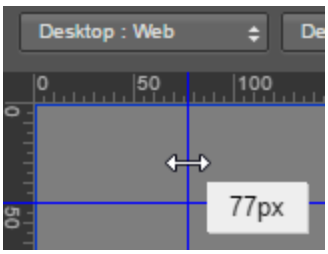
- Click on either the horizontal or vertical ruler, and drag onto the canvas. A guide tracks with the mouse pointer onto the Visualizer Canvas, displaying the position of the guide as you go.



Reposition a Guide

To reposition a guide, do the following:

1. Hover over the guide with the mouse pointer until it becomes a two-headed arrow.



2. Click and drag the guide to the position you want.

Lock and Unlock Guides

To lock and unlock guides, do the following:

1. In the upper right corner of the Visualizer Canvas, click the round option button, hover over **Ruler**, and then click **Lock Guides**.
2. To unlock the guides, click the round option button, hover over **Ruler**, and then click **Unlock Guides**.

Hide or Show Guides

To hide or show guides, do the following:

1. In the upper right corner of the Visualizer Canvas, click the round option button, hover over **Ruler**, and then click **Hide Guides**.
2. To show the guides, click the round option button, hover over **Ruler**, and then click **Show Guides**.

Clear a Guide

To clear a guide, do the following:

1. Hover over the guide with the mouse pointer until it becomes a two-headed arrow.
2. Click and drag the guide back to the ruler.

Clear All Guides

To clear all guides, do the following:

- In the upper right corner of the Visualizer Canvas, click the round option button, hover over **Ruler**, and then click **Clear All Guides**.

Display Widget Command Handles

Many widget commands can be accessed from the Visualizer Canvas by opting to display the widget command handles. By default, these are not activated. There are two handles. One is a delete handle X and the other is a context menu handle >. When the command handles are activated and a widget is selected, it looks like this:

Clicking the delete handle deletes the widget immediately with no additional prompts or warnings. Clicking the context menu handle displays all the commands available to that particular widget, and is equivalent to clicking a widget's context menu arrow from the Project tab of the Project Explorer.

The process for enabling the display of widget command handles varies depending on whether or not you are using Kony Visualizer or Kony Visualizer Classic.

Display Widget Command Handles for Kony Visualizer

Display Widget Command Handles for Kony Visualizer Classic

Display Widget Command Handles for Kony Visualizer

To display widget command handles, do the following:

1. On the **Edit** menu, click **Preferences**.
2. On the left side of the Visualizer Preferences dialog box, click **General**.
3. Set the Widget Highlighter Actions option to **On**.
4. Click **Apply**.

Display Widget Command Handles for Kony Visualizer Classic

To display widget command handles for Kony Visualizer Classic, do the following:

1. On the **Window** menu, click **Preferences**.
2. On the left side of the Visualizer Preferences dialog box, click **General**.
3. Set the Widget Highlighter Actions option to **On**.
4. Click **Apply**.

   > ***Important:*** You must click **Apply** to set your changes. If you click **OK** or navigate to another location in the Preferences dialog box without first clicking **Apply**, any changes you made will be lost.

5. Click **OK**.

Select Multiple Items

With Kony Visualizer, you can select multiple items located under the same parent at the same nesting level, making it easier to make changes to their common properties.

Select Multiple Items from the Project Explorer

Select Multiple Items from the Visualizer Canvas

Deselect All Items

Availability of Properties and Commands with Multiple Items Selected

The following tables list the common properties that are editable and the context-menu commands that are functional when multiple items are selected. These vary depending on where in the Kony Visualizer user interface you access them.

Availability of Properties

Availability of Tabs in the Properties Editor

The following table lists which tabs are available and unavailable in the Properties Editor when multiple items are selected:

Tab	Availability
Look	Some properties are available, depending on what properties are common among the selected items. The Look tab properties available across all selectable items are listed in this table.
Skin	Unavailable.
Widget	Unavailable.
Action	Unavailable.
Review	All functionality is available.

Available Properties on the Look Tab

The following properties are available to edit on the Look tab of the Properties Editor when multiple items are selected:

Property	Availability
Appearance properties	Unavailable.
Padding properties	Unavailable.
Visible	Available.
Left	Available.
Right	Available.

Top	Available.
Bottom	Available.
Width	Available.
Height	Available.
Min Width	Available.
Max Width	Available.
Min Height	Available.
Max Height	Available.
Center X	Available.
Center Y	Available.
Z Index	Available.

Availability of Context Menu Commands

Available Project Explorer Commands

Unavailable Project Explorer Commands

Available Visualizer Canvas Commands

Unavailable Visualizer Canvas Commands

Available Project Explorer Commands

The following context menu commands are available on the Project tab of the Project Explorer when multiple items are selected:

Context Menu Command	Description
Cut	Cuts the selected items, removing them and placing them on the clipboard.
Copy	Copies the selected items.
Paste	Deselects the selected widgets and pastes the widgets from the clipboard into the selected container or form.
Delete	Deletes the selected items.

Hide on Canvas and Unhide on Canvas	When items are visible on the Visualizer Canvas, the **Hide on Canvas** command is available. Selecting **Hide on Canvas** makes the selected items invisible on the Visualizer canvas, but they are still listed on the Project tab of the Project Explorer. When items are invisible on the Visualizer Canvas, the **Unhide on Canvas** command is available. Selecting **Unhide on Canvas** makes the selected items visible once again on the Visualizer canvas.
Lock and Unlock	Selecting Lock makes the selected items undraggable or resizable using the mouse on the Visualizer Canvas, although their common properties on the Look tab of the Properties Editor can be modified. Selecting Unlock unlocks the selected items and makes them capable of being dragged and resized using the mouse on the Visualizer Canvas.

Unavailable Project Explorer Commands

The following context menu commands are not available on the Project tab of the Project Explorer when multiple items are selected:

Context Menu Function	Description
Move Up	Disabled since widgets may belong to too many separate hierarchies
Move Down	Disabled since widgets may belong to too many separate hierarchies
Rename	
Unparent	Disabled since widgets may belong to too many separate hierarchies

Available Visualizer Canvas Commands

The following context menu commands are available on the Visualizer Canvas when multiple items are selected:

Context Menu Command	Description

Cut	Cuts the selected items, removing them and placing them on the clipboard.
Copy	Copies the selected items.
Paste	Deselects the selected widgets and pastes the widgets from the clipboard into the selected container or form.
Delete	Deletes the selected items.
Duplicate	Deselects the selected widgets and duplicates them into the selected container or form
Bring Forward	Brings the entire set of selected widgets to a higher Z Index
Bring to Front	Brings the entire set of selected widgets in front of the highest Z Index
Send Backward	Brings the entire set of selected widgets to a lower Z Index
Send to Back	Brings the entire set of selected widgets to the behind the lowest Z Index
Hide on Canvas and Unhide on Canvas	When items are visible on the Visualizer Canvas, the **Hide on Canvas** command is available. Selecting **Hide on Canvas** makes the selected items invisible on the Visualizer canvas, but they are still listed on the Project tab of the Project Explorer. When items are invisible on the Visualizer Canvas, the **Unhide on Canvas** command is available. Selecting **Unhide on Canvas** makes the selected items visible once again on the Visualizer canvas.

Lock and Unlock	Selecting Lock makes the selected items undraggable or resizable using the mouse on the Visualizer Canvas, although their common properties on the Look tab of the Properties Editor can be modified. Selecting Unlock unlocks the selected items and makes them capable of being dragged and resized using the mouse on the Visualizer Canvas.
Group Into	The available sub-commands are: Flex Container Flex Scroll Container
Convert to Component	Nests the selected items, along with any instance of a component that might be present, into a new component.
Select All	Selects all of the widgets within the Visualizer Canvas or the selected container. Selects only the widgets that are direct descendents of the Visualizer Canvas.

Unavailable Visualizer Canvas Commands

The following context menu commands are not available on the Visualizer Canvas when multiple items are selected:

Context Menu Command	Description
Widget Actions	No Widget Actions are available in the context menu when multiple widgets are selected.
Add to Collection	
Create Widgets From Component	Available only if a single component is selected.

Select Multiple Items from the Project Explorer

To select multiple items from the Project Explorer, do the following:

1. On the Project Explorer, navigate to the form that contains the widgets you want to select.
2. Expand the form on the Project Explorer so that all of its widgets are listed.
3. While holding down the Ctrl key, click the widgets within the same nesting level that you want to select. The name of each widget you select becomes highlighted.

> ***Note:*** If you mistakenly select an item, to deselect, click it again.

Select Multiple Items from the Visualizer Canvas

To select multiple items from the Visualizer Canvas, do one of the following:

- On the Visualizer Canvas, while holding down the Ctrl key, click the widgets that you want to select. The boundary handles of each widget you select become highlighted. For multiple to be selected, they must all be located under the same parent at the same nesting level.
- On the Visualizer Canvas, click anywhere outside of the outline of the device, and then drag the widgets, selecting the ones you want.

> ***Note:*** If you mistakenly select an item, to deselect, click it again.

Deselect All Items

- To deselect all items, on the Visualizer Canvas, click anywhere outside the outline of the device.

Undo and Redo an Operation

Kony Visualizer includes the ability to undo and redo an operation.

The Undo command allows you to discard the most recent change in Kony Visualizer. The Redo command reverses the most recent change made using Undo.

To undo an action, on the **Edit** menu, click **Undo**, or press Ctrl + Z. To redo an action, on the **Edit** menu, click **Redo**, or press **Ctrl +Y**.

The following are limitations of the Undo and Redo commands:

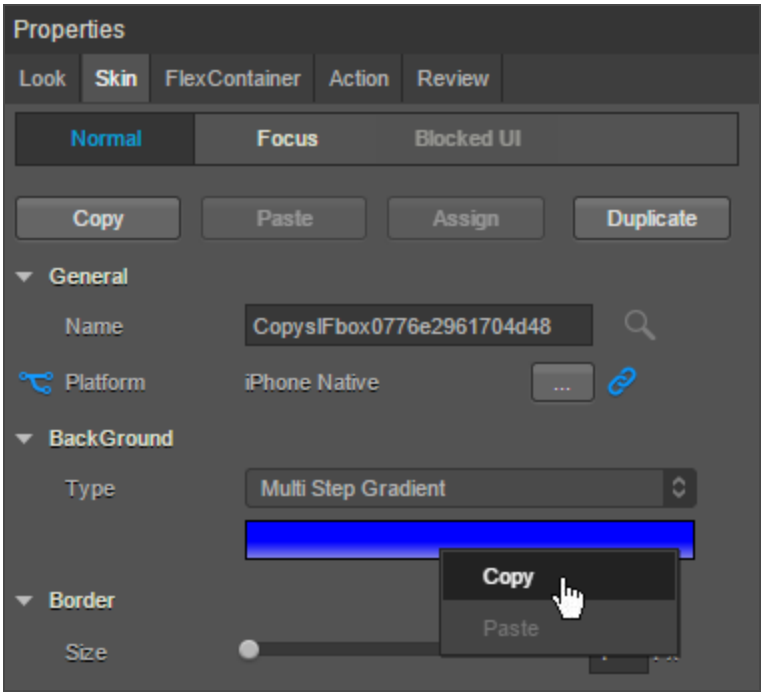
- Undo and Redo cannot be used related to operations involving skins and action sequences.
- You can use Undo to rollback all the widgets added to a form or pop-up, but you cannot rollback the addition of a form or pop-up.
- Undo and redo cannot be used on custom widgets.
- You cannot undo or redo an action performed from the shortcut menu of a form or pop-up.

Copy and Paste a Color or Gradient

With Kony Visualizer, you can reuse a color or gradient by copying it from one widget property and pasting it to another property, such as from a background to a border. Gradients are especially tedious to replicate, so being able to copy and paste them can save you considerable time and effort.

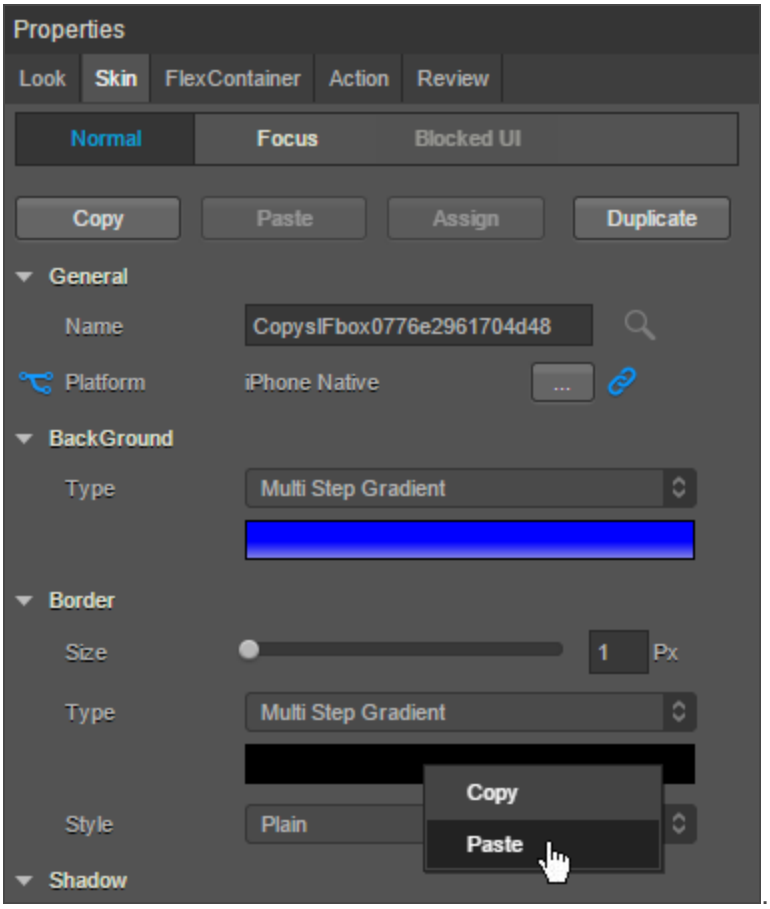
To reuse a color or gradient, follow these steps:

1. On the Properties tab of a widget, click **Skin**.
2. Right-click the color palette of a property and then click **Copy**.



3. Depending on whether you want to paste the color and gradient within the same widget or another widget, do one of the following:

    - **Within the same widget.** Right-click the color palette of a property, and then click **Paste**.
    - **Different widget.** Navigate to the Skin tab of the widget, right-click the color palette of a property, and then click **Paste**.



> ***Notes:***
>
> - The color palette of the copied property and the pasted property should be of the same type. That is, if the copied color is of multi-step gradient, the pasted property color should also be multi-step gradient.

- You can reuse a color across the widgets. That is, you can copy the color of a button widget and paste it to a Flex Container widget.
- You cannot copy and paste an image.

Jump to the Definition of a Code Element

As you're working on code, you can jump to the location where a given element of code is defined, such as a function or variable. Doing so expedites your coding by making it easy for you to identify a code element, such as what a particular function does.

To jump to the definition of a code element, do the following:

1. Do one of these two actions:
   - If you have a code module open such as a .js file, place the insertion point anywhere within the code element that you want to find the definition for. It isn't necessary to highlight the code element.
   - If the definition you're seeking is for a function in the search results of the Search tab, click the function in the search results that you want to find the definition for.
2. While holding down the **Alt** key, press the **period** key (i.e. **Alt + .**). Kony Visualizer opens the asset that contains the code element to the line number where the code element is defined.

Open a Resource's Folder

All the resources of your project, such as forms and widgets, or assets such as images, reside in folders in your workspace. With Kony Visualizer, you can open the folder location of any resource from the Project Explorer.

To open a resource's folder, do the following.

- In the Project Explorer, click the context menu arrow of any resource, and then click **Resource Location**.

  The folder where that resource is located opens.

Copy between Designer and Developer Actions

Applies to *Kony Visualizer Classic*.

If you have Designer actions in your project, you can copy them to Developer actions. Conversely, you can copy Developer actions to Designer actions.

> ***Important:*** If you copy Designer actions to Developer actions, any existing Developer actions are overwritten.

To copy between Designer and Developer actions, do the following:

- On the **Edit** menu, point to **Copy Actions**, and then do one of the following:
  - To copy Designer actions to Developer actions, click **Designer to Developer**. If your project has existing Developer actions, a dialog box displays warning you that proceeding will overwrite the existing Developer actions. To proceed, click OK. Otherwise, click **Cancel**.
  - To copy Developer actions to Designer actions, click **Developer to Designer**.

Copy and Paste Forms and Actions across Channels

By copying forms in one **channel**[1] and pasting them to another, you can cut down significantly on your design and development time. Any action sequences associated with a form are also copied and pasted. You can copy and paste both flex and VBox forms.

> ***Note:*** This functionality is only available for flex forms, and is not available for popups.

To copy and paste forms and their action sequences from one channel to another, do the following:

[1]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

1. In the Project Explorer, on the **Project** tab, navigate to the form tor forms that you want to copy.

   To select multiple forms, hold down the **Ctrl** key, and then click the forms you want.

2. Right-click the selection, and then click **Copy**.

3. Navigate to the channel you want to paste the form(s) to, and then expand it.

4. In the destination channel, click the Forms context menu arrow, and then click **Paste**. The form or forms are pasted into the channel, including all action sequences, and retains the original form name. You may want to rename the pasted form so that it is easier to distinguish from the original. To do so, click the pasted form's context menu arrow, and then click **Rename**.

5. Repeat the previous step for any additional channels you want to paste the form to.

Display a List of JavaScript Code Elements

When you are editing a JavaScript file in the Code Editor, Kony Visualizer gives you the ability to display a list of the code elements in that file, in the order that they appear. These code elements include globally available objects, functions, arrays, numbers, and strings. By double-clicking a code element in the list, the insertion point navigates to that code element.

> ***Note:*** This functionality does not apply to popups or code snippets.

> ***Notes:***
> - The listed results are drawn form the last saved instance of the JavaScript. Any unsaved code is not reflected in the results.
> - This functionality does not apply to popups or code snippets.

To display a list of code elements in a JavaScript file, do the following:

1. Open a JavaScript file in the Code Editor. To do so, in the Project Explorer, on the **Projects** tab, expand the **Modules** section, and then click the JavaScript file you want to open.

2. Press **Ctrl+O** (⌘ + O on the Mac). A list of code elements displays.

3. Navigate to any code element by double-clicking it from the list.

Expand and Contract the Visualizer Canvas

To maximize your work area on the Visualizer Canvas, you can expand it, regardless of whether you are working on a form, a code module, or an action sequence. When you maximize the Visualizer Canvas, it marginalizes the Project Explorer and Library Explorer on the left, the Properties pane on the right, and the Console along the bottom. Expanding the Visualizer Canvas works in both Single and Side by Side views. Contracting the Visualizer Canvas and restoring the Project Explorer, Library Explorer, Properties pane, and Console is as simple as a double-click.

To expand and contract the Visualizer Canvas, do the following:

1. Along the top of the Visualizer Canvas, double-click the tab of the form, action sequence, or module that you want to see in an expanded manner. The user interface elements to the left, right, and below the Visualizer Canvas are marginalized.

2. To contract the Visualizer Canvas, double-click the tab of the element you're working on. The user interface elements to the left, right, and below the Visualizer Canvas are restored to their previous dimensions.

Go to the Most Recent Edit Location

As you design an application, you may frequently switch between code modules, forms, and action sequences, making it challenging to return to the JavaScript file where you had last been coding. With Kony Visualizer, doing so is easy. The Last Edit Location feature returns you to the last JavaScript file you edited, even if it's currently closed.

To go to the most recent edit location, do the following:

- On the **Edit** menu, click **Last Edit Location**. Kony Visualizer displays the most recently edited JavaScript file from the **Module** section of the **Project** tab in the Project Explorer, placing the insertion point at the row and character position of the last keystroke you entered.

  > ***Note:*** You can also press **Ctrl + Q** (**Alt + Q** on the Mac).

Jump to a Specific Line of Code

While working on code in the Code Editor, you can jump to a specific line number. This functionality is available for any code snippet or code file.

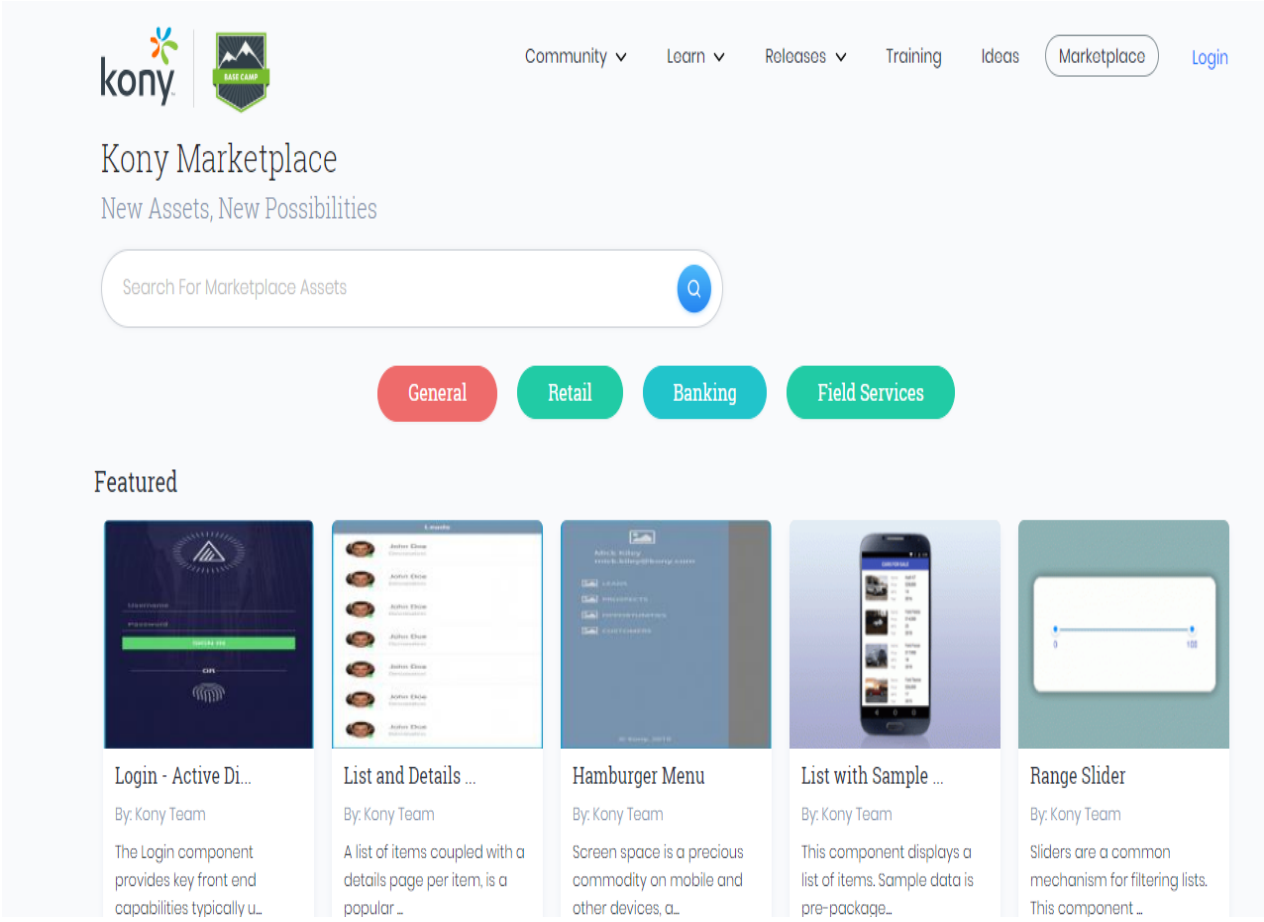
To jump to a specific line number, do the following:

1. Press **Ctrl + L**. A command bar opens.
2. In the **Jump to line** text box, type the line number you want to jump to, and then click **Go**. The insertion point jumps to the line number you specified.

Creating Applications With Components

Components help you quickly develop sophisticated applications with dynamic, responsive user interfaces across multiple channels – phones, tablets, wearables, and desktops. You can use components as building blocks for rapid application development without having to write all the code yourself.

For example, you can build your applications from a rich assortment of components created by the Kony Marketplace Assets team and other developers. These components are available for download from Kony Marketplace:



You can also build your own reusable components to use in your applications or publish to Kony Marketplace.

Components are an extension of *masters* in earlier versions of Kony Visualizer. You can continue to use existing masters or import other masters into your applications as components.

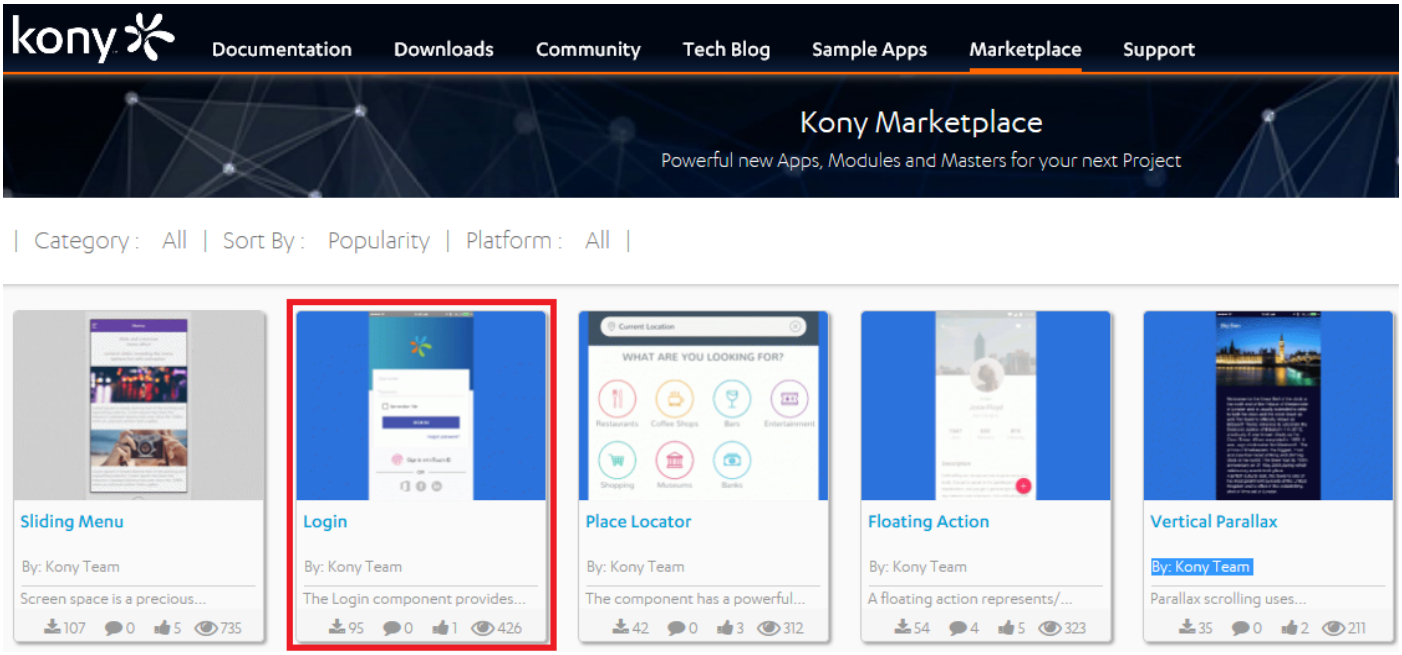
The following topics describe how to create digital applications with components:

Components Overview

Components provide a powerful way to create complex applications quickly. You can download a rich assortment of components from Kony Marketplace, or create your own reusable components, and then drag and drop the components into your application to create sophisticated, full-featured applications without writing all of the code. From V8 SP4 Fixpack 20, you can also leverage several Data & Services panel features for components.

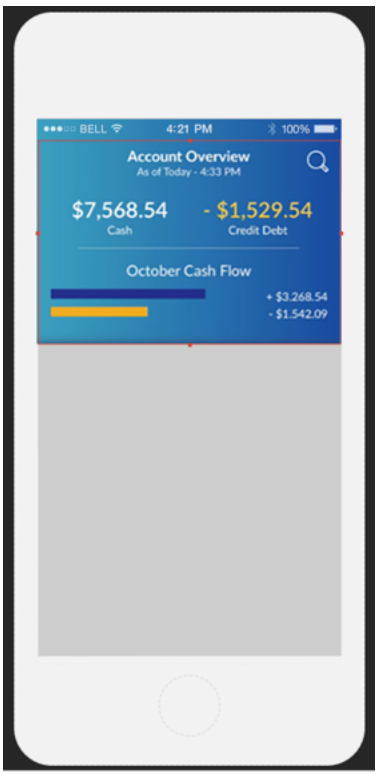
For example, suppose you want to create a digital application that lets users log in to their account and display an account overview. You can build a log-in screen by downloading a log-in component from Kony Marketplace, and then dragging and dropping it onto a form.

First, navigate to the Kony Marketplace site, either by selecting **Browse** from the **Marketplace** menu in Kony Visualizer or opening the site in your web browser. Then download the Login component.



You can easily import the component into your Kony Visualizer library, and then drag and drop it onto a form.

You could then create an account overview component that can be used in multiple applications, and drag and drop that component onto a form to display an account overview in your application. Your library can contain multiple collections of components that can be used and reused, serving as building blocks for your applications.



If you are familiar with earlier versions of Kony Visualizer, you may recognize components as an extension of *masters*. Masters helped streamline digital application developing by letting you define multiple user interface elements and action sequences as a single entity. Components extend that ability by providing self-contained and reusable entities that include:

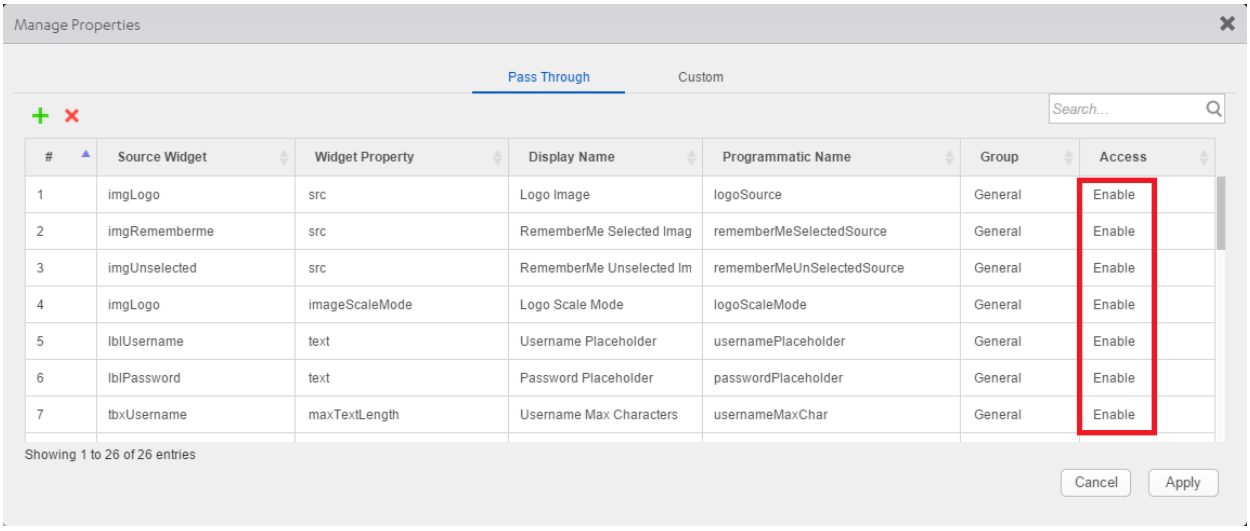
- Code modules
- Kony Fabric services
- A component *contract* to simplify usage and enable rapid digital application development

Because components are self-contained entities, you can:

- Reuse a component within a project, or across projects in your workspace.
- Download components created by the Marketplace Assets team and other developers from Kony Marketplace, and use them in your projects.
- Share your components with a wider audience via Kony Marketplace or a local share.

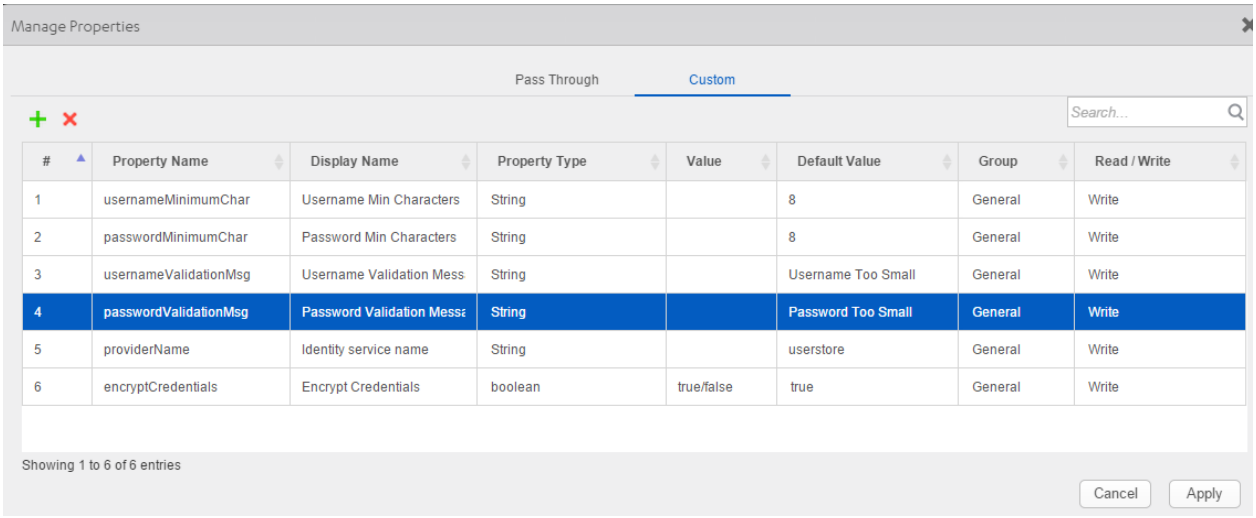
Components are also fully customizable. You can choose which properties, events, and methods of a component are exposed to users, or create your own custom properties, events and methods.

For example, you can specify which properties of the Login component are exposed to users by enabling them in the **Manage Properties** dialog box:



These are called *pass through* properties.

The Login component also includes custom properties. You can specify custom properties on the **Custom** tab of the **Manage Properties** dialog box, and then define their behavior in the component's controller module, *loginController.js*:



Here is the code that defines the passwordValidationMsg property:

```
defineGetter(this, "usernameValidationMsg", function() {
    konymp.logger.trace("----------Entering usernameValidationMsg Getter---------", konymp.logger.FUNCTION_ENTRY);
    return this._usernameValidationMsg;
});
defineSetter(this, "usernameValidationMsg", function(val) {
    konymp.logger.trace("----------Entering usernameValidationMsg Setter---------", konymp.logger.FUNCTION_ENTRY);
    this._usernameValidationMsg = val;
});
defineGetter(this, "passwordValidationMsg", function() {
    konymp.logger.trace("----------Entering passwordValidationMsg Getter---------", konymp.logger.FUNCTION_ENTRY);
    return this._passwordValidationMsg;
});
defineSetter(this, "passwordValidationMsg", function(val) {
    konymp.logger.trace("----------Entering passwordValidationMsg Setter---------", konymp.logger.FUNCTION_ENTRY);
    this._passwordValidationMsg = val;
});
defineGetter(this, "providerName", function() {
    konymp.logger.trace("----------Entering providerName Getter---------", konymp.logger.FUNCTION_ENTRY);
    return this._providerName;
});
defineSetter(this, "providerName", function(val) {
    konymp.logger.trace("----------Entering providerName Setter---------", konymp.logger.FUNCTION_ENTRY);
    this._providerName = val;
});
```

There are two types of Kony Visualizer components:

- Components with a contract.
- Components without a contract.

The contract is a JSON file that specifies the component's exposed properties, events, and methods. You specify those properties, events, and methods within the Visualizer interface using the **Manage Properties**, **Manage Events**, and **Manage Methods** dialog boxes. You can then define the behavior of custom properties, events, and methods by adding code to the component's controller module.

You can also create a component without contract. A component without contract is similar to a master in earlier versions of Kony Visualizer, except that it includes a controller module. You can publish either a component with contract or a component without contract to Kony Marketplace, but not a master. However, you can convert a master into either a component with contract or a component without component.

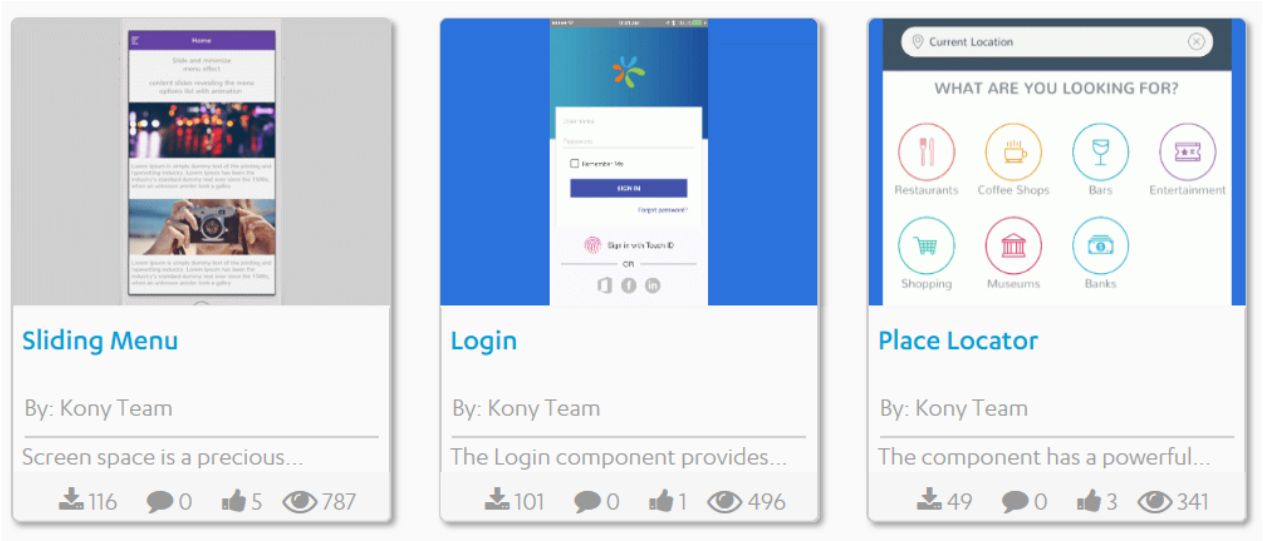
For more information about using components, refer Using Components. For details about creating a component, refer Creating a Component.

Use Components

You can simplify and speed up development of digital applications by using a variety of existing components as building blocks, allowing you to create sophisticated digital applications without having to write all the code yourself. You can download a variety of components from Kony Marketplace and import them into your projects, or import reusable components that you create yourself.

For example, you can download the following free components developed by the Kony Marketplace Assets team, and drag and drop them into your applications:

- **Sliding Menu**: Lets you hide the navigation beyond the edge of the screen, and reveal it only after a user's action.
- **Login**: Provides a standard log-in interface with the ability to enforce a minimum password and minimum user ID length, and local password encryption.
- **Place Locator**: Provides a powerful map interface that allows you to search for points of interest near a specified location.
- **Floating Action**: Represents and promotes the primary action in an application.
- **Employee Directory**: Provides standard features associated with an employee directory.
- **Amadeus**: Provides a data adapter that helps you rapidly build travel applications.



You can also import masters created in an earlier version of Kony Visualizer into a Free Form JavaScript project. You can continue to use the masters in your applications, or make them available as reusable components that can be published to Kony Marketplace by converting them to components. A master is similar to a component without a contract, except that a component includes a controller module.

For information about creating components, see Creating a Component. For information on the types of projects, you can create, see Types of Projects.

The following topics provide additional information about using components:

- Download a Component from Kony Marketplace
- Import a Component into Your Project
- Add a Component to a Collection
- Add Dependent Assets to a Component
- View Details and Documentation for a Component
- Add a Component to a Form
- Import a Master into Your Project
- Export a Component
- Publish a Component to Kony Marketplace
- Work With a Private Section of Kony Marketplace
- Moderator Capabilities for the Private Section of Kony Marketplace
- Configure Snap Mode for Components
- Flatten a Component

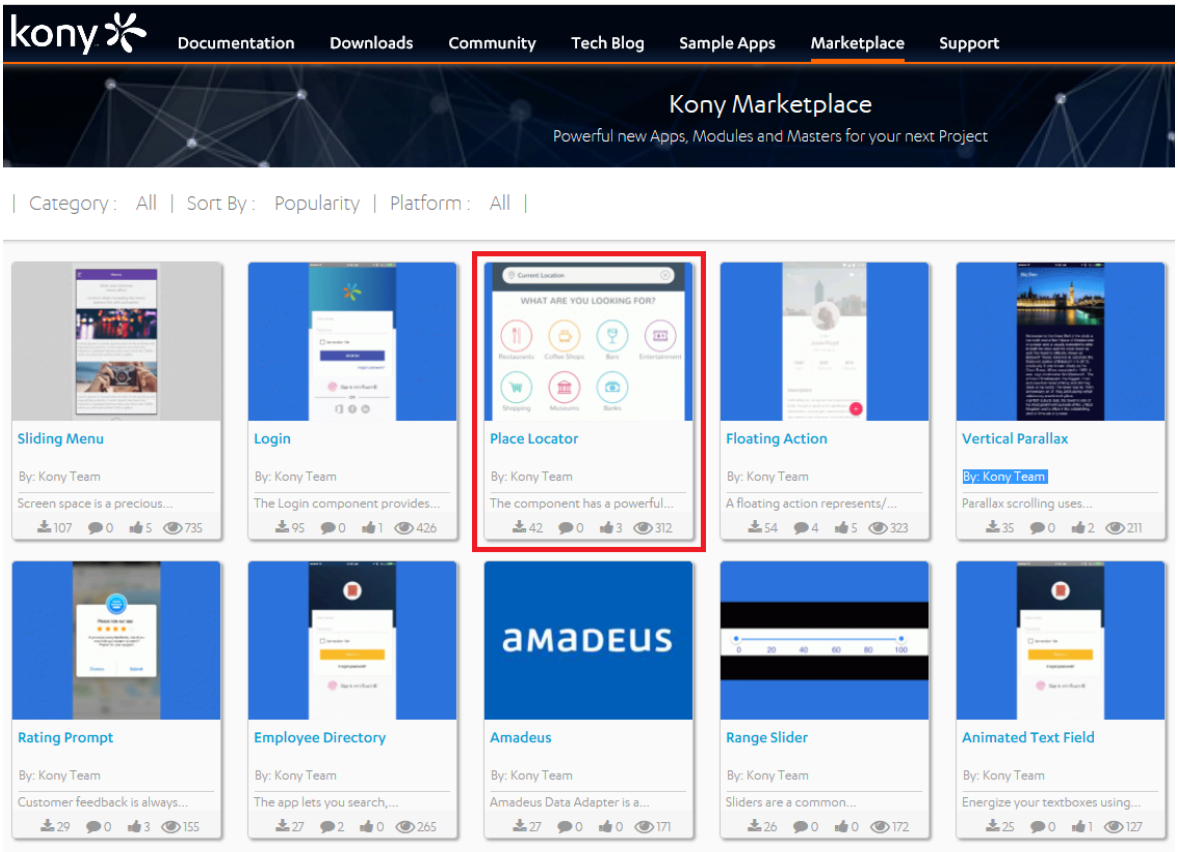
Download a Component from Kony Marketplace

Kony Marketplace gives you access to a rich assortment of components. You can download a component, and then import it into your project.

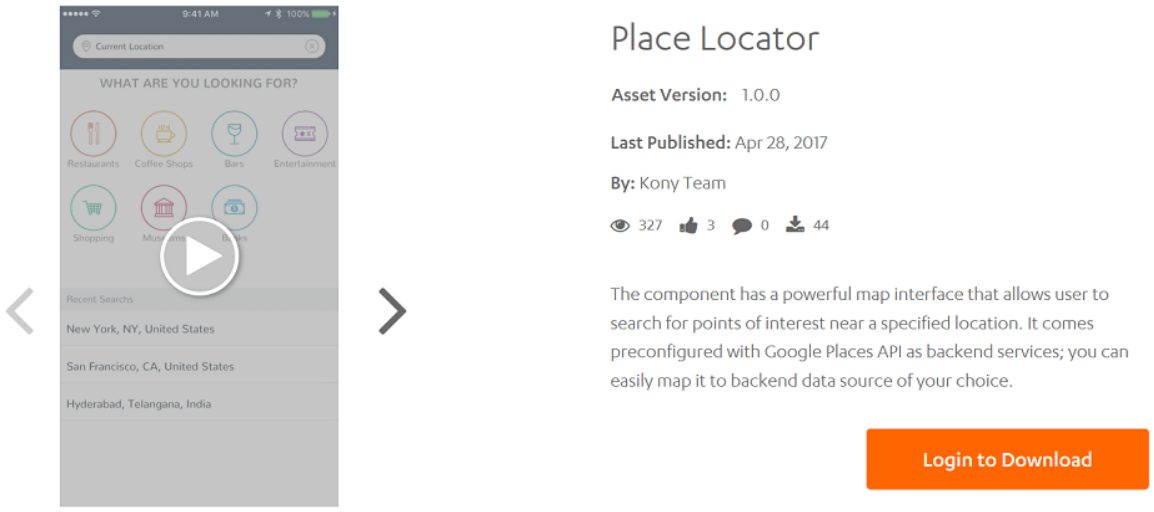
Click here to watch a video to learn more about the myriad features of Kony Marketplace, such as Data Adapters, Data Models, Components, and more.

To download a component from Kony Marketplace:

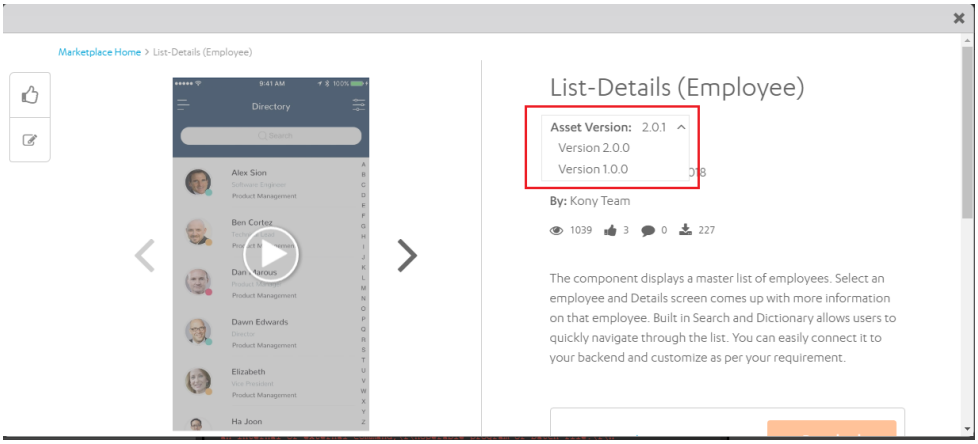
1. Select **Browse** from the **Marketplace** menu, or navigate to the Kony Marketplace website in your web browser. If you are not already logged into Kony Cloud from the Visualizer, a login window appears. Enter the Kony Cloud account credentials.



2. Select the component you want to download to open a web page that describes the component.

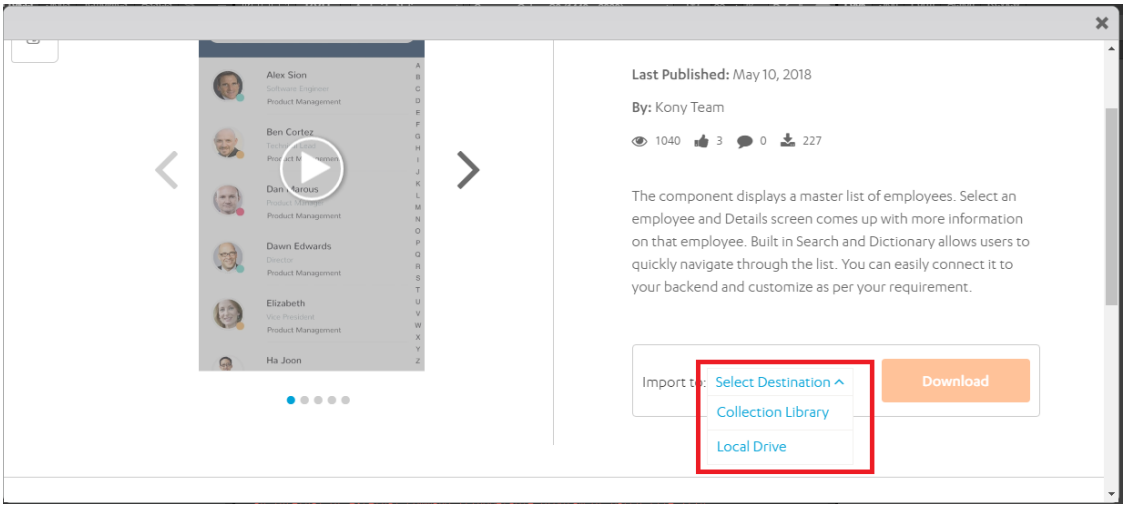


From Asset Version, select the version of the component to download. Once you are done selecting the asset version, scroll down.



3. Click the **Login to Download** button to log in to your Kony account and download the component.

4. From **Select Destination** list, select one of the options
   • Select **Collection Library** to add the component to your library in Visualizer
   • Select **Local Drive** to download the component to your local drive. You can import this component later in your Visualizer project.



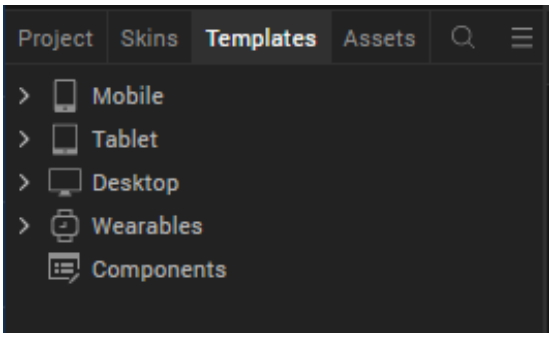
Import a Component into Your Project

You can import components to your Kony Visualizer project and reuse them at necessary places in the application. Components can be imported from the following sources:

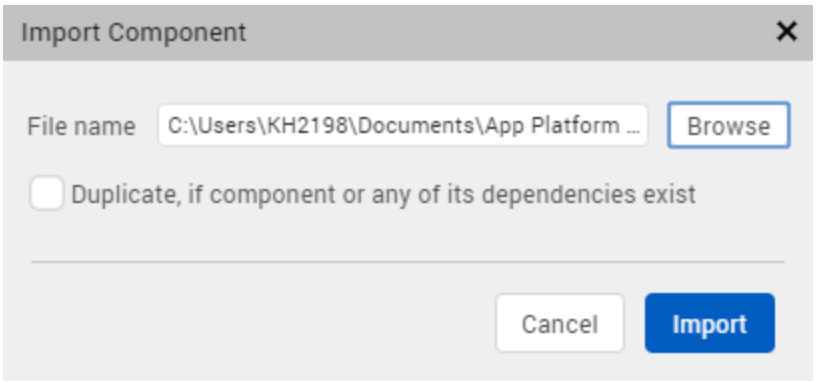
- Components that you download from Kony Marketplace.
- Components that you create and export to your computer, network, or the cloud.
- Components that others create and export to your computer, network, or the cloud.

To import a component to your project, follow these steps:

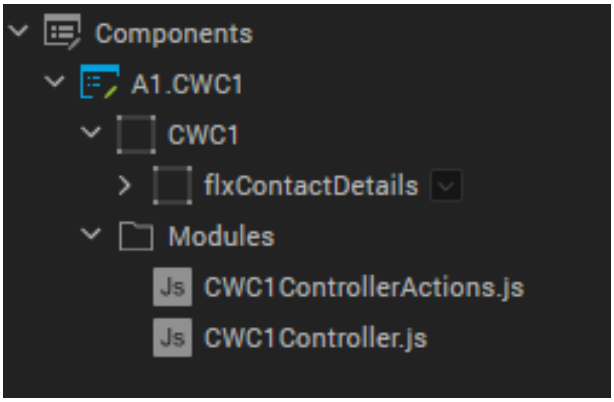
1. In the Project Explorer, click the **Templates** tab.



2. Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.



3. Click **Browse** to navigate to the location of the component, select the component, and then click **Import**. The component and its associated widgets and modules are added to your project.



Once you have imported a component to your project, you can easily add the component to a form. For more information, refer Add a Component to a Form.

You can also save the component to a collection. For more information, refer Add a Component to a Collection.

Add a Component to a Collection

To help you organize your components, you can group them in a collection within the Library Explorer. You can organize related components, or organize them however you like, in multiple collections within a library. You can then easily drag and drop the component from the Collections pane of Library Explorer onto a form in any of your digital application projects.

To add a component to a collection, follow these steps:

1. Click the Templates tab in either the Project Explorer or the Library Explorer.
2. If it is not expanded already, expand the Components node.
3. Click the context menu arrow of the component you want to add to a collection, and hover over **Add to Collection**. Then hover over the library that contains the collection, and click the collection where you want to save the component. If the library or collection does not exist, you can create the library or collection.

If the component uses any Kony Fabric services, Kony Visualizer Classic prompts you to select the services that you want to include with the component. Kony Visualizer creates a trial Kony Fabric account and includes any required services.

You can also add a master created in earlier versions of Kony Visualizer to a collection.

Add Dependent Assets to a Component

Starting with Kony Visualizer V 8.1, you can add skins, media assets, and images to components. In previous releases, when you create a component and export the component to the marketplace, associated dynamic skins, and media assets were not added to the collection automatically. These skins and media assets used to be accessed through the code that you customized within the component.

> ***Important:*** When you are importing a component into a project, ensure that the component assets do not have the same name as any of the assets in the project you are importing the component. Visualizer does not warn you of any duplication, and you may lose some functionality.

Add Skins to a Component

To add skins to a component, do the following:

1. In Project Explorer, click **Skins**.
2. Right-click on the skin you want to add to a component. A list of options appears.
3. Click **Add to Component**. The Add to Component dialog appears.
4. Select the component to which you want to add the skin. The skin is added to the selected component.

When you add a skin to a component, a live two-way dependency map is maintained to track skins associated to a component. If you delete a skin from a project, the skin is removed from associated components.

Add Media Assets to a Component

To add media assets to a component, do the following:

1. In Project Explorer, click **Assets**. The assets tab appears.
2. Browse to the asset you want to add to a component.
3. Right-click on the asset. A list of options appears.
4. Click **Add to Component**. The Add to Component dialog appears.
5. Select the component you to which you want to add the media asset. The asset is added to the selected component.

If you delete an asset from a project, the asset is removed from associated components.

Directly Import Images into a Component

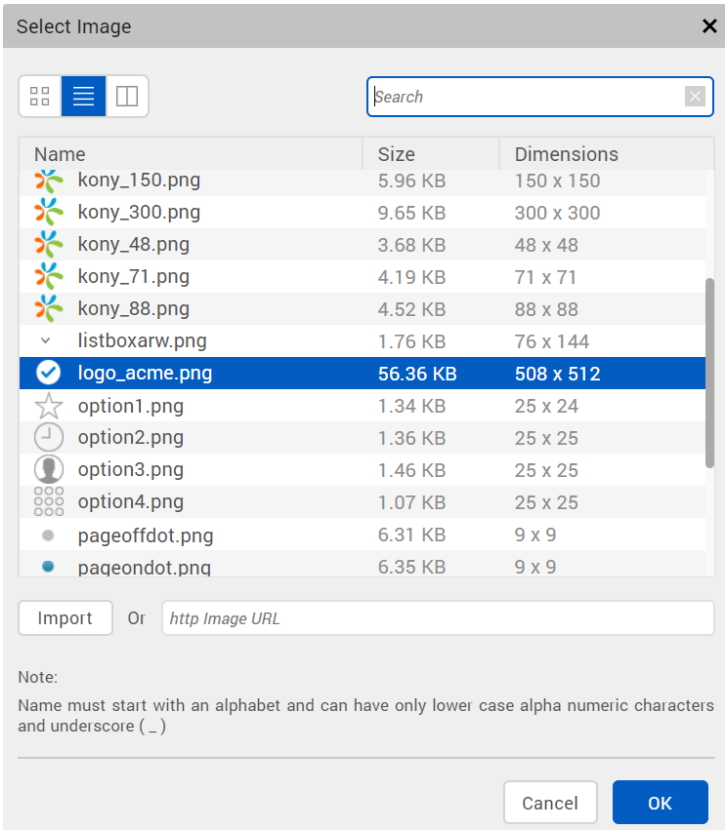
From Kony Visualizer V8 SP4 Fixpack 28 onwards, you can directly import an image from your local computer and add it to the widget in a component. Previously, to change the asscoiated image of a Login component, you had to first import the image to the Assets tab and only then you could add it to the component.

To directly import an image and add it to a component, follow these steps:

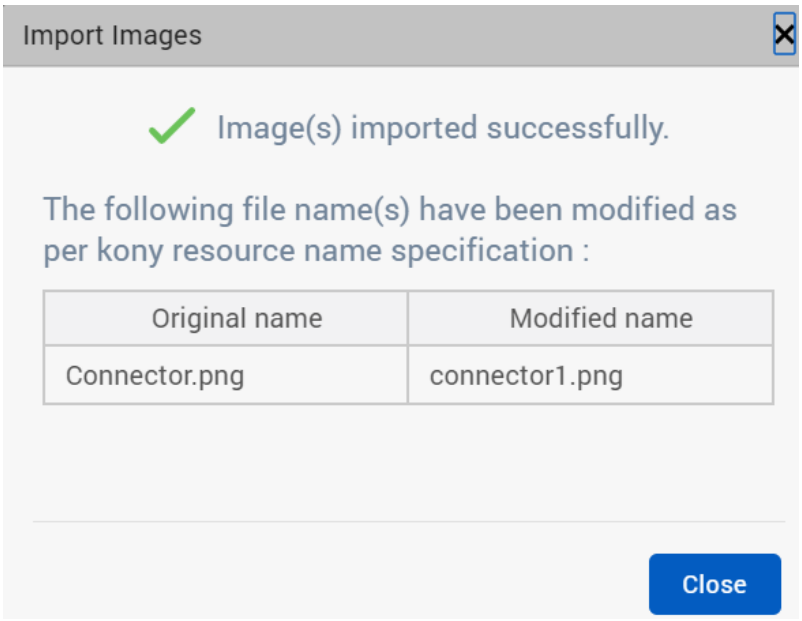
1. Drag and drop a component to your form. For instance, the **ACME Login** component from the **Default Library**.
2. You can perform any of the following actions:

   - On the Project Canvas, double-click the success image  .
   - In the Project Explorer, go to your form > ACMELogin, right-click **imgLogo**, and then click **Edit Image Source**.

The **Select Image** dialog box appears, containing the list of available imported images.



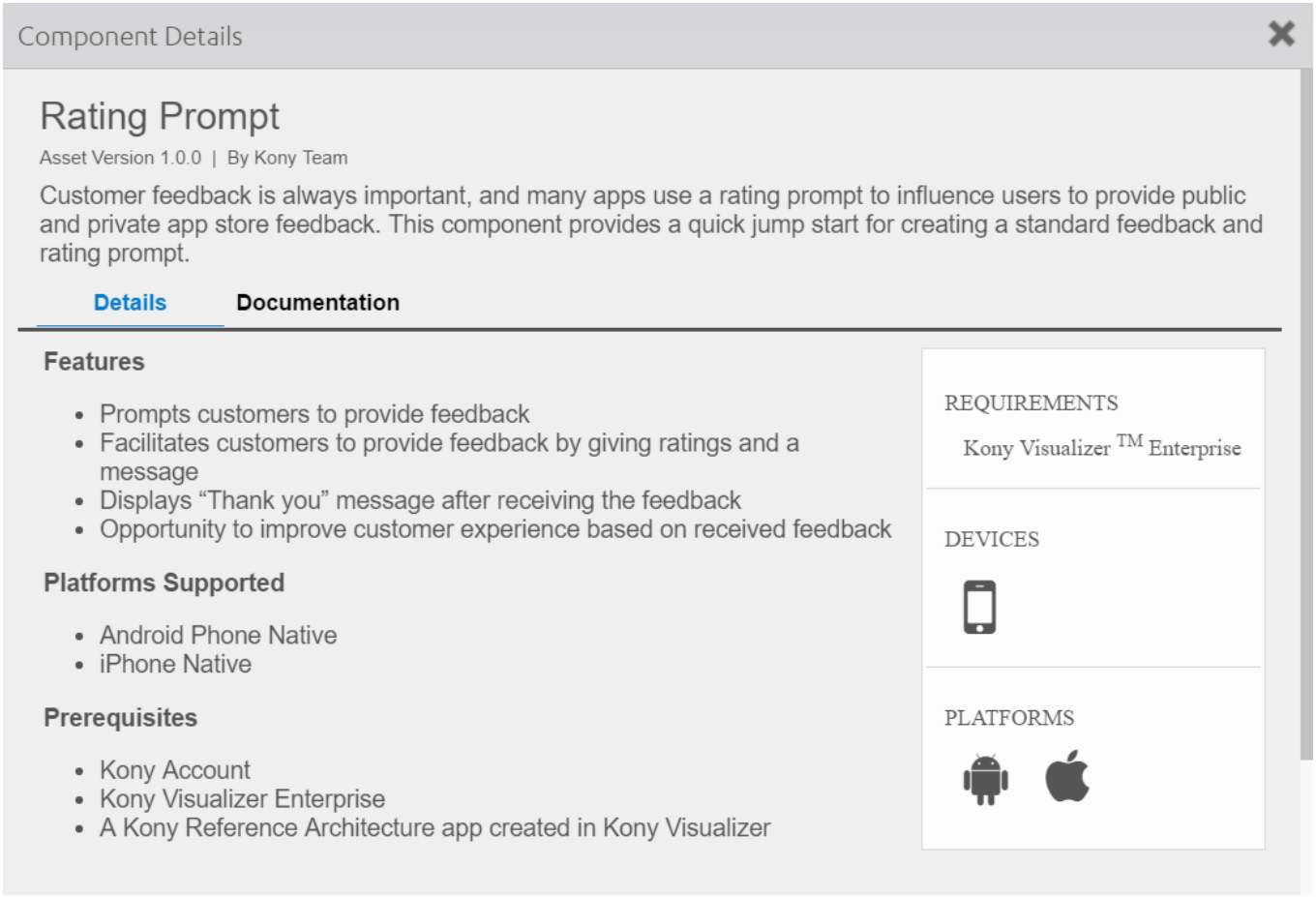
3. Click **Import**. The File Explorer of your local computer appears.
4. Select the required image, and then click **Open**. The Import Images dialog box appears, indicating that the image has been imported successfully. If required, Kony Visualizer automatically modifies the file name of the imported image according to the Kony resource name specification.

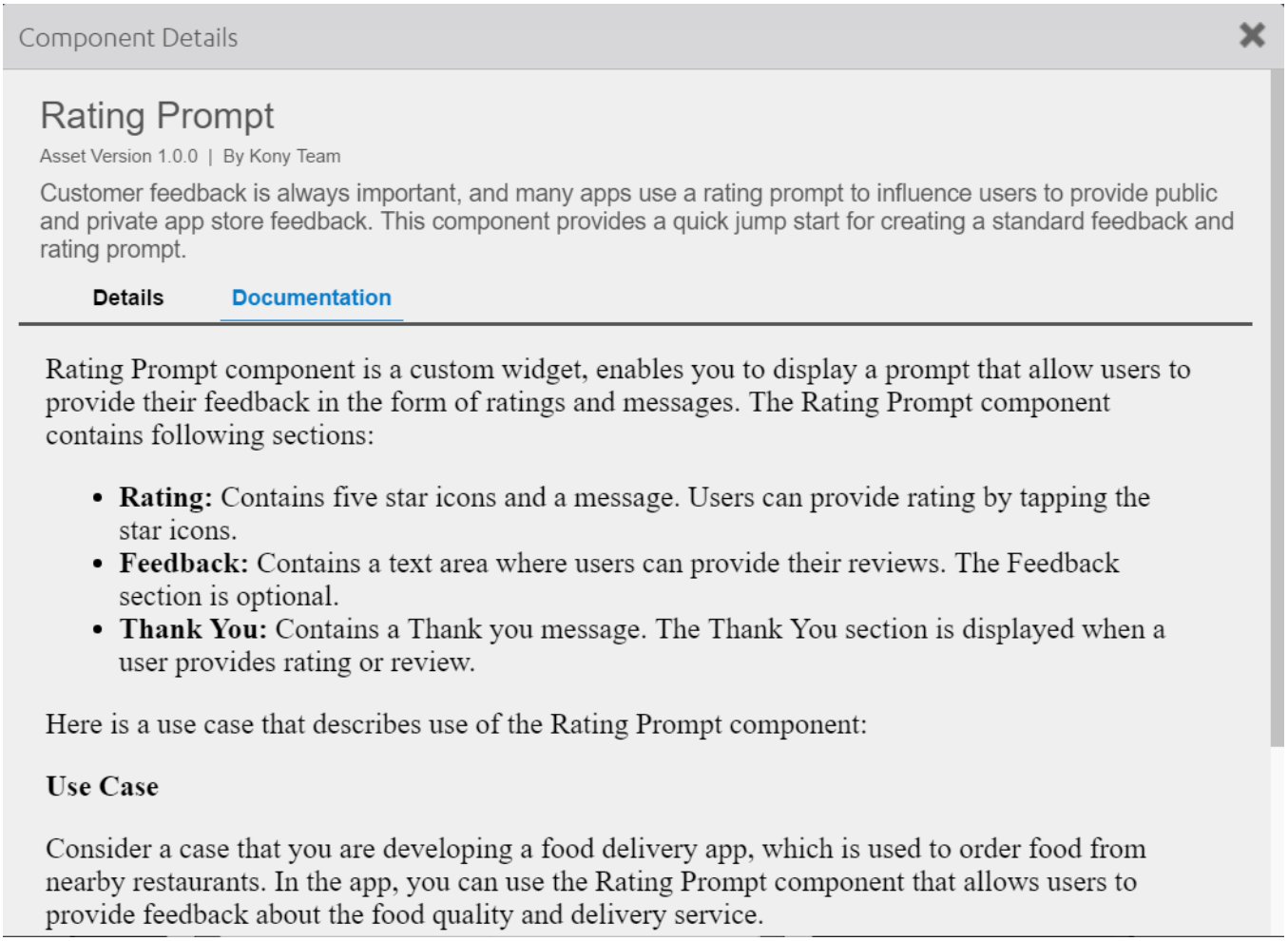


5. The imported image appears in the list of images of the **Select Image** dialog box. Select the image, and then click **OK**. The image is added to the ACMELogin component.

View Details and Documentation of a Component

Components downloaded from Kony Marketplace often include documentation supplied by the component creator. For example, the Rating Prompt component developed by the Kony team includes a summary of key details about the component, along with a more complete description of the component and how to use it.





You can view this documentation, if it exists, from within Library Explorer in the collection where you saved the component.

To view details and documentation about a component:

1. In Library Explorer, go to the **Default Library** and **Collection** where you saved the component.

2. Right-click the component name and select **Details**. Kony Visualizer opens a viewing pane and displays the documentation.

   To switch between details and documentation view, click **Details** or **Documentation**.

Add a Component to a Form

You can add a component to a form in a project in virtually any form scenario, including using components within another component. You can then drag and drop the component into your application to create sophisticated, full-featured digital applications without writing all of the code.

For example, rather than creating a log-in form from scratch for each of your digital applications, you can download the Login component from Kony Marketplace. You can add the Login component to a collection of components, and then add it to a form.

To add a component to a form:

1. On the Project tab of the Project Explorer, locate and open the form where you want to add the component. The form displays on the Visualizer Canvas and has the focus.
2. If you have added the component to a collection, open the library and collection that contains the component in Library Explorer. Then drag and drop the component onto the form.

> ***Note:*** From Kony Visualizer V8 SP4 Fixpack 28 onwards, the Default Library filter has been enhanced to segregate components according to their respective channels. As a result, mobile-only components are displayed in the Default Library when a Mobile form or a Tablet form is open at that time on the Project Canvas. Similarly, web-only components are displayed Default Library when a Web form is open on the Project Canvas. However, when there is no form open on the Project Canvas, all the available components are displayed in the Default Library.

If the component is added to the project and appears on the **Templates** tab, you can also do one of the following:

- Click the context menu arrow of the component you want to use, and then click **Insert Into**. An instance of the component is placed on the form you originally selected.
- Drag the component from the **Templates** tab to the Visualizer Canvas, and drop the component onto the form.

The instance appears in the Project Explorer as blue text with a different icon, making it easy to identify as a component instance.

Import a Master into Your Project

You can also use a master created in an earlier version of Kony Visualizer in your digital application by importing the master into a Free Form JavaScript project. For information on the types of projects you can create, see Types of Projects.

To import a master:

1. On the **File** menu, hover over **Import**. Then select **Masters**.
2. Click **Browse** to navigate to the location of the master, select the master, and then click **Open**. Kony Visualizer adds the master to the Masters node on the Templates tab in Project Explorer. If a Masters node does not exist, Kony Visualizer creates one.

You can then drag and drop the master onto a form just as you would drag and drop a component.

Export a Component

Just as you can import a component into your project, you can export a component to your computer, network, or the cloud. You can then import the component into another project or share it with others.

To prevent your component from being modified by users, you may want to lock the component. This prevents users from viewing or modifying the component's source code. For more information about locking a component, see Creating a Component.

> ***Note:*** Locales are not exported with a component. For more information about locales, see Appendix A • Internationalizing (i18n) Application Content.

To export a component, follow these steps:

1. If you have added the component to a collection, open the library and collection that contains the component in Library Explorer. Then select the component.

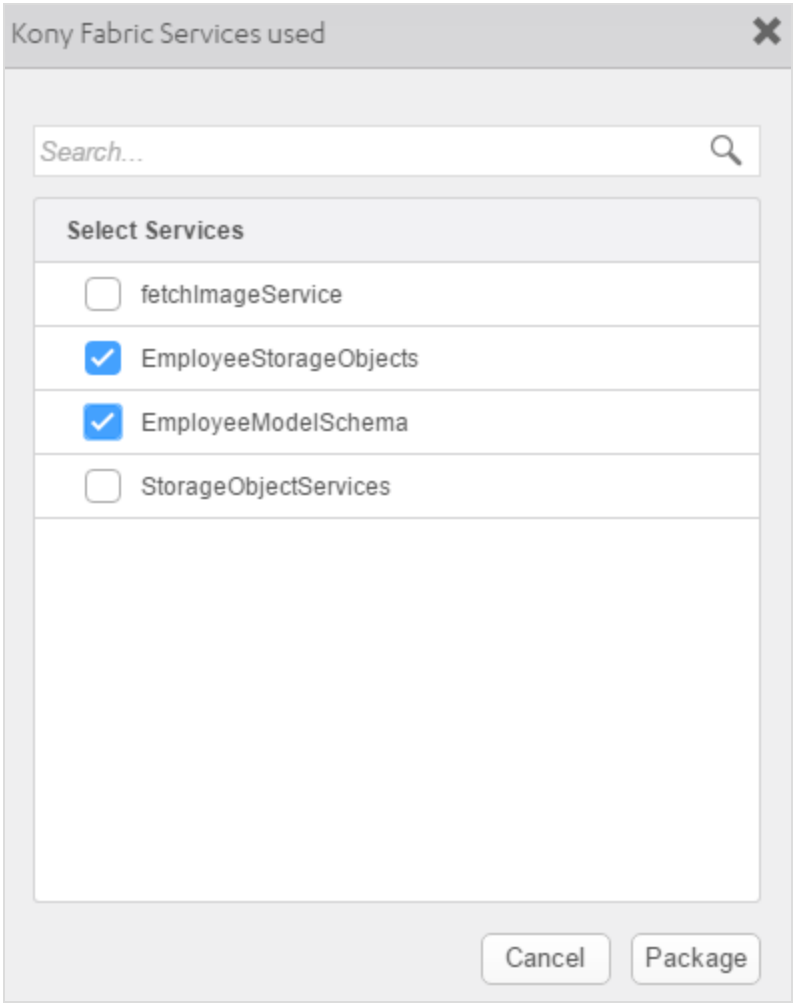
   If the component has been added to the project and appears on the **Templates** tab, select the component there.

2. Right-click the component, and then select **Export**. Kony Visualizer displays the **Save As** dialog box.

3. Navigate to the location where you want to export the component, and then click **Save**. Kony Visualizer exports the component to that location.

> ***Note:*** While exporting a component, if any project-level properties have been enabled, these properties are automatically applied to the exported component. To avoid this issue, you must first disable the required NFIs for either iOS or Android from Edit > Manage Native Function APIs > iOS/Android and then export the component.

If the component uses any Kony Fabric dependencies, including .jar files, service definitions, connectors, or node.js files, a pop-up is displayed with the list services used in the component.



Select the required services by clicking the checkbox and click **Package** to export (or) to add it to the library.

If you create a component that requires configuration of Kony Fabric service parameters, follow these steps:

1. Extract the exported component package file to your local file system.
2. Go to **Components** folder and extract the `services.zip` package.

3. Create a `Parameters.json` file with all the configurable parameters.

   The `JSON` file includes the information about all the configurable parameters for the services included in the component.

4. Place the `Parameters.json` file in the `Services` folder of the services package.

   Here is a sample `JSON` file with the default parameters.

```
[{
    "name": database_url,
                "displayName": < Display name of the parameter to
show in Properties Window > ,
                "defaultValue": < Mandatory
    if property is readOnly,
    Optional otherwise > ,
                "type": < Type of the parameter.Can be one of
String / Integer / Double > ,
                "serviceName": < Name of the MF service > ,
                "serviceType": < Type of the MF service.Can be
one of identity / integration / object / orchestration > ,
    "readOnly": true / false
}, { < configurable param 2 >
}]
```

5. Update the corresponding file located in `component_package/component/services/services/services/service_type/meta.json` with the placeholders. Example, endpoint file for service RDBMS123 is updated as:

```
<? xml version = "1.0"
encoding = "UTF-8"
standalone = "yes" ?> < endpoint dataAdapterId = "1"
encryptSecureInfo = "false"
name = "default"
```

```
type = "CustomDataAdapter" >     < apiThrottling / >     < config
>         < entry >         < entry >             < key > jdbcUrl
< /key>
            <value>${database_url }</value >         < /entry>
            <key>jdbcClass</key >             < value >
com.mysql.jdbc.Driver < /value>
        </entry >     < /config>
</endpoint >
```

6. Zip the services folder and the component folder and import the package to Visualizer. For more information on configuring the parameters in Visualizer, refer Configuring Kony Fabric service parameters in Visualizer.

Publish a Component to Kony Marketplace

To publish a component to Kony Marketplace, first add it to a collection in your project. For more information on adding a component to a collection, see Add a Component to a Collection.

Click here to watch a video to learn about how to submit Kony Visualizer components to Kony public Marketplace or to your private Marketplace.

> ***Note:*** You can only publish a component with a contract to Kony Marketplace. For more information about components with a contract, see Components Overview and Creating a Component.

To publish a component to Kony Marketplace:

1. Open the collection that contains the component that you want you publish.
2. Right-click the component that you want to publish, and then select **Publish**.
3. Supply log-in information to your Kony account, if necessary, and provide the necessary information to Kony Marketplace.

4. Supply log-in information to your Kony account, if necessary, and provide the following information to Kony Marketplace

    - The name of the component
    - The asset version
    - Description of the asset
    - The new feature added in the component in this version(optional)
    - Details such as code snippets(optional)
    - Asset Display view(optional)
    - Detail View Images/Video(optional)
    - Developer guide documentation (optional)
    - Kony documentation link (optional)
    - External links(optional)
    - Asset requirements (optional)
    - Related tags (optional)
    - Domain (optional)
    - Additional information (optional).

5. Click 'Submit for Review' to upload the component to the Marketplace.

> ***Note:*** While uploading a component again, you must change the asset version.

Work With a Private Section of Kony Marketplace

If you are an enterprise customer whose organization uses a private section of Kony Marketplace, you can download a component from that private section or upload a component to that private section using your organization's cloud account. First, configure Kony Visualizer or Kony Visualizer Classic to use the cloud account.

Click here to watch a video to learn about the various features of Private section of Kony Marketplace.

To work with a private section of Kony Marketplace in Kony Visualizer:

1. Close Kony Visualizer.
2. Open the *default.js* file in the *config* directory of your Kony Visualizer installation.
3. Add the following to the *default.js* file.

   ```
   mp:{privateMPAccountId: <MyAccountId>}
   ```

   <MyAccountId> is the account identifier of your cloud account. The account identifyer is displayed in the top-right corner of your cloud console on *manage.kony.com*.
4. Save and close the *default.js* file, and then reopen Kony Visualizer.

To work with a private section of Kony Marketplace in Kony Visualizer Classic:

1. Click **Project Settings** on the **Quick Launch Bar** or in **Project Explorer** to open the **Project Settings** dialog box, and then click the **Kony Fabric** tab.
2. In the **Cloud Account** box, select your cloud account.

To download a component, see Download a Component from Kony Marketplace. To upload a component, see Publish a Component to Kony Marketplace.

Moderator Capabilities for the Private Section of Kony Marketplace

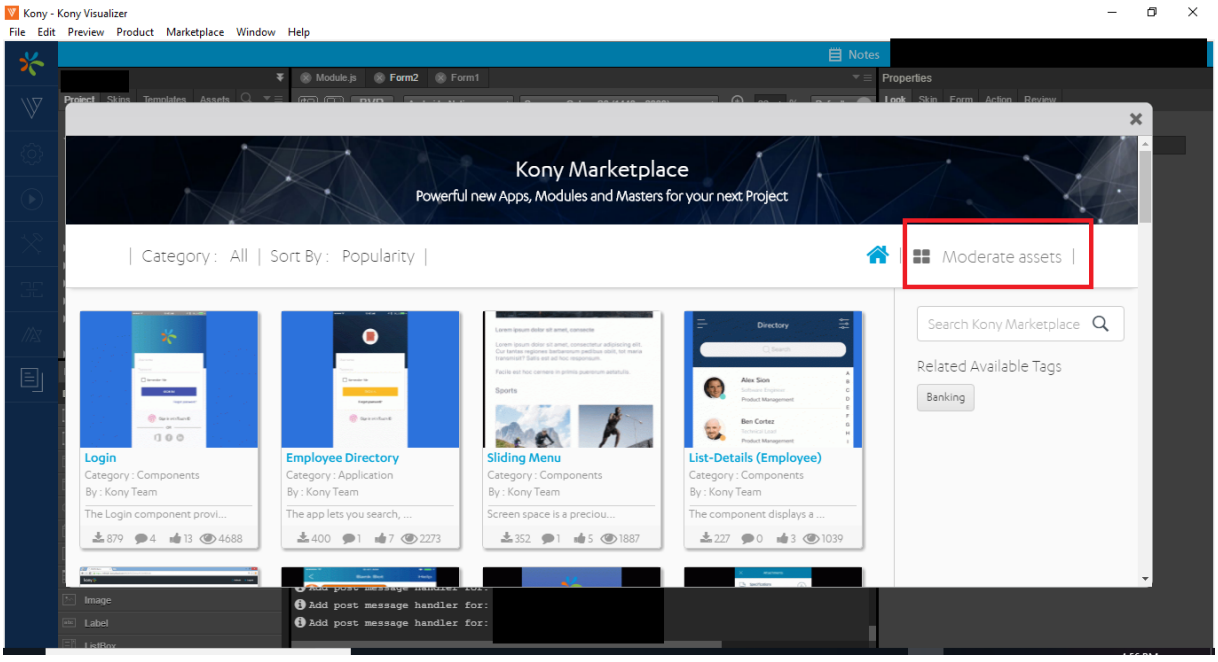
Kony Marketplace offers a private section where you can upload your own components, distribute them among your teams, download them, and use them.

Moderators manage the private section. As a moderator, you can review a component, approves them, or delete them.

If you want to be a moderator, you must have a Kony cloud account, and your cloud administrator must provide access to the private marketplace as a moderator.

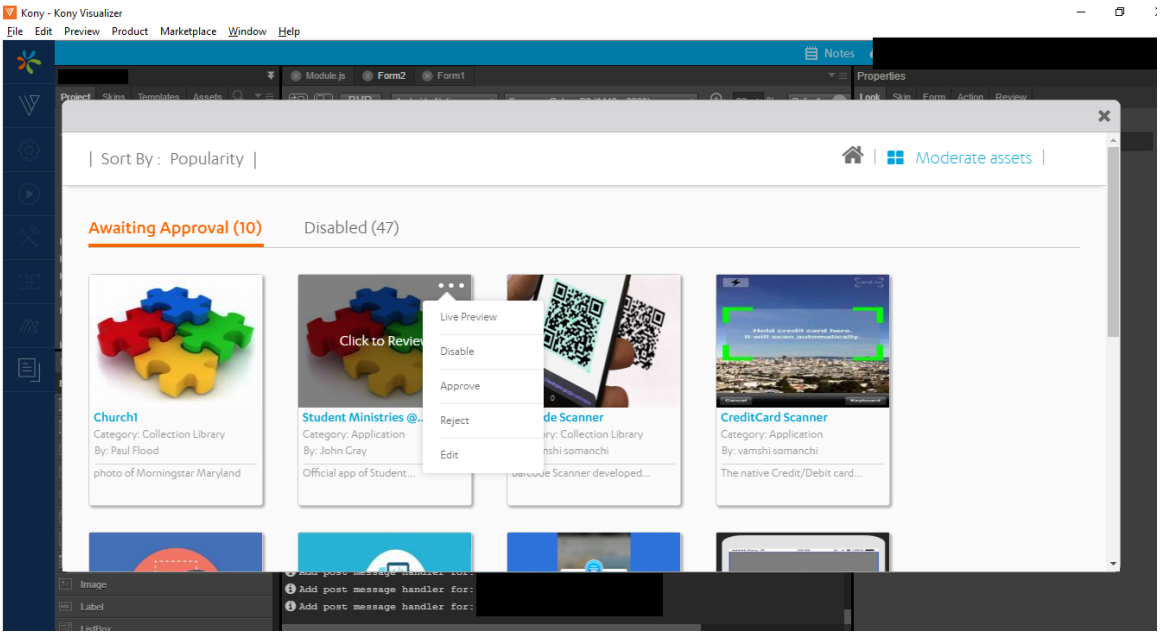
To Manage components as a moderator, follow these steps.

1. In Visualizer, navigate to Marketplace > Browse. Or Navigate to your private marketplace website using a web browser. In case you are not logged in to Kony Cloud, Visualizer displays a login window. Enter your Kony Cloud account credentials. Once you are logged in, the list of available assets appears.
2. Click Moderate Assets. The list of components appears.



The components are sorted under two tabs: Awaiting Approval and Disabled.

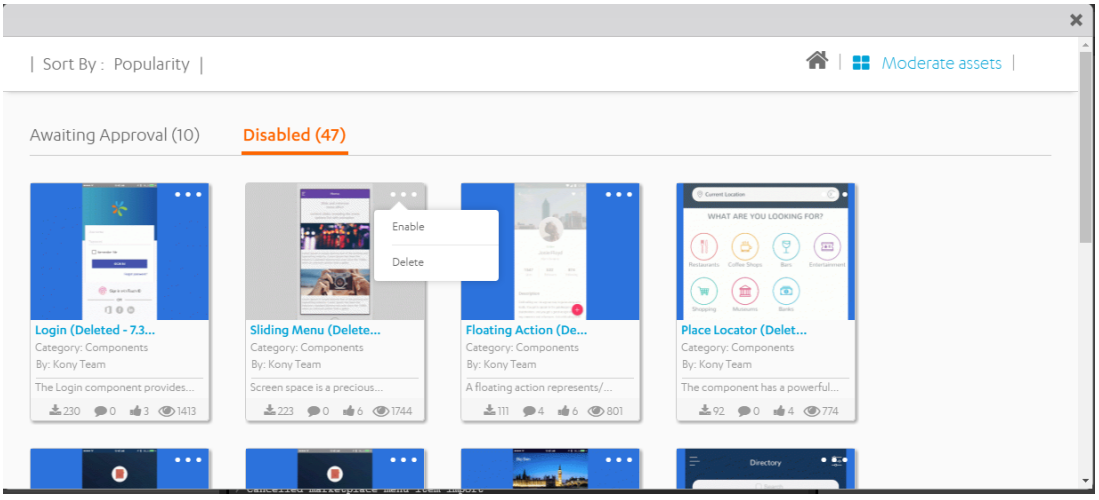
3. To modify a component, in the Awaiting Approval tab, click the ellipsis.



4. From the list, select an action. Options are:
   · Live Preview- Shows the moderator how the component will look when approved
   · Disable- Disables the component and move it to the Disabled tab
   · Approve- Approves the component for publishing it
   · Reject- Rejects the component. It will not be visible even in the Disabled tab
   · Edit- Gives option to edit the whole component details such as Description, title, version, etc.

5. In the Disabled tab, click on the ellipsis of the component.

6. From the list, select an option. Options are:
   • Enable- Enables the component and moves it to the Awaiting Approval tab
   • Delete- Deletes the component from the list



Configure Snap Mode for Components

When a user wants to create a component, if the user wants to select an option to switch between the default value for Drag and drop experience for a component, he can do that by the snap to drag position or snap to the layout.

In Kony Visualizer V8 SP1, Kony introduced **Snap Mode** feature which enables a user to snap all components and collections to the layout. A user can configure this in the preferences section of Kony Visualizer.

If Snap is OFF, all components and collections are placed where the user drags them. When a user selects Alt + Drag, the function will work opposite to what is configured.

To turn on the Snap Mode, do the following:

1. In your Kony Visualizer project, from the file menu, navigate to **Window > Preferences**. The Preferences window appears.
2. Navigate to **Kony Visualizer**. The Kony Visualizer section appears.
3. Select **Snap Mode**.
4. Click **OK**. The Preferences window closes. The Snap mode is turned on.

Customizing a Component

With V8 SP3 GA, a new feature is provided to help you to get the UI elements of a component on to a form without affecting the parent component and reuse the component's UI elements. You can use Component Flattening to:

- Use UI elements without the pre-bundled logic
- Modify Skin or the UI
- Modify Widget properties

You can further bundle your modified form to a new component.

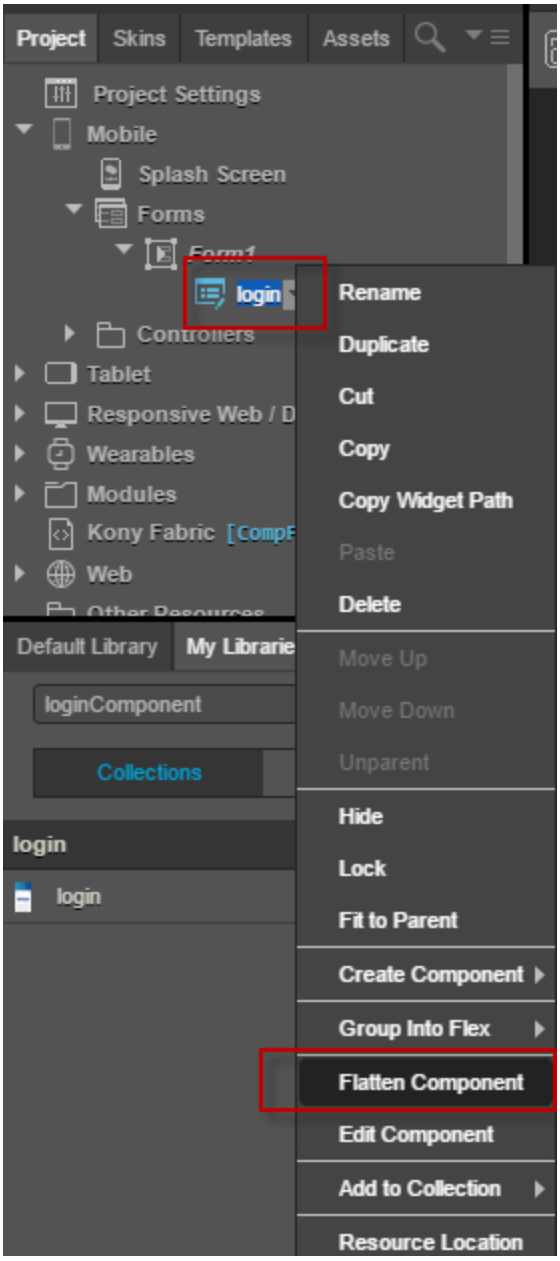
The feature, **Flatten Component** is applicable for components with contracts and components without contracts. It is also applicable to pre-built components.

To flatten a component, perform these steps:

1. In Kony Visualizer, open your project in which you have a component that you want to flatten. If you do not have a component already in the project, see Import a Component into Your Project.
2. In the project explorer, navigate to the form on which you have the component added.
3. Right-click on the component and select **Flatten Component**.
   A confirmation message appears.
4. Click **OK**.
   All widgets that are in the component are unpacked and are available on a new flexContainer and you can edit those widgets. Actions associated with widgets in the component are carried forward to the widgets.

> ***Important:*** All associated controllers and module files in component are lost after flattening. This might result in loss of functionality of your component.

Any actions associated with the source model widgets present in the form are copied to the newly created flexContainer. All actions also are moved to a channel specific directory.



Limitations:

- During the flattening process, nested instances are not further unpacked.
- You cannot flatten locked components.

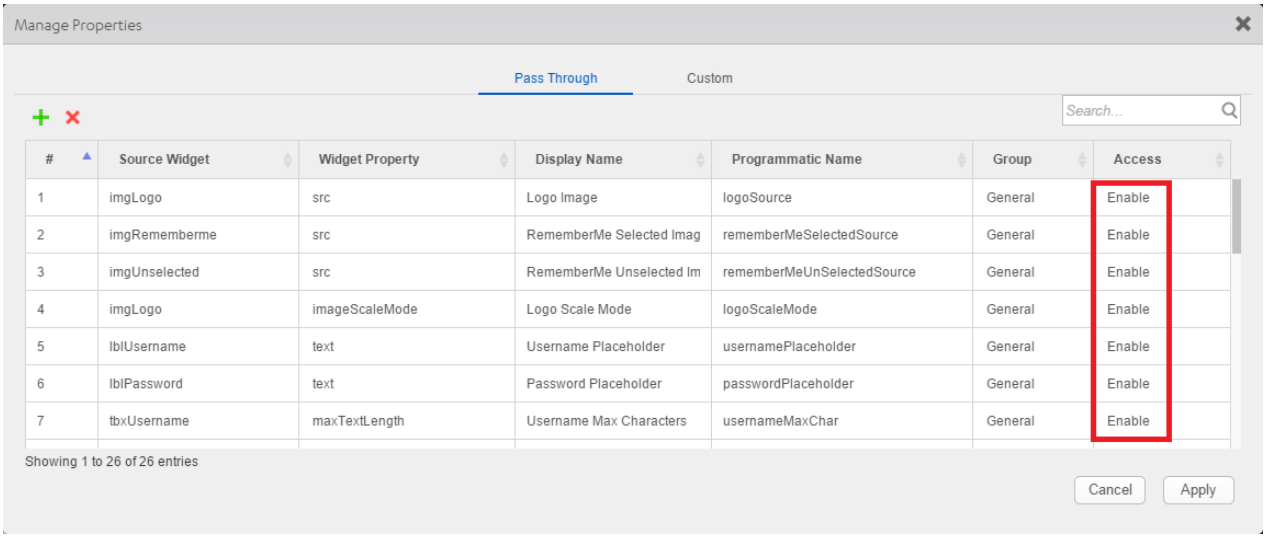
- During the flattening of the UserWidgetInstance, exposed properties for children inside a target container will not be retained.

Create a Component

The most powerful way to create reusable components is to create a component that includes a defined *contract*. The contract is a JSON file that specifies the component's exposed properties, events, and methods. You can specify those properties, events, and methods within the Visualizer interface, and then define the behavior of custom properties, events, and methods by adding code to the component's controller module.

When you create a component, you have a choice of creating a component with or without a contract. A component without a contract is similar to a master in earlier versions of Kony Visualizer, except that it includes a controller module. You can also convert a component that does not have a contract, or an imported master, to a component with a contract.

Once you create a component with a contract, you can control what is exposed to users of the component. You expose only the properties, events, and methods that you want to make available to your target audience, enhancing your ability to rapidly create and deploy digitalapplications for different audiences. The exposed properties, events, and methods are part of the contract.



You can then use the component in other applications, or share your custom components with other digitalapplication developers by publishing them to Kony Marketplace. For more information about using components, see Using Components.

> *Important:* Parent value for components will always be null. In component controller code, **this.view.parent** will always return a null value.

The following topics provide additional information about creating components:

- Create a Component without Contract
- Create a Component with Contract
- Expose a Component's Widgets
- Data & Services Panel Support for Components
- Specify a Container Widget as a Target Container
- Expose a Component's Skins
- Lock a Component
- Manage Properties of a Component with Contract
- Manage Events of a Component with Contract
- Manage Methods of a Component with Contract
- Group Properties, Events, and Methods of a Component with Contract
- Set Data for Components with Contract by using Mapping Editor
- Map Service Parameters to the Segment in a Component
- Define the Behavior of a Custom Property in Code
- Define a Custom Event in Code

Create a Component Without Contract

A component without contract is similar to a Master in earlier versions of Kony Visualizer, except that a component includes a Controller module. You can reuse the component within your application and distribute it via Kony Marketplace. However, you cannot control which of the component's properties, events, and methods are exposed, or create custom properties, events, and methods. To control the exposed properties, events, and methods of a component, create a component with contract.

To create a component without contract, follow these steps:

1. In the Project Explorer, click the **Templates** tab.
2. Right-click **Components**, point to **New**, and then select **w/o Contract**. The **Create new Component without Contract** dialog box appears.
3. Enter a **Namespace** and **Name** for the component. The component should follow this naming convention: *<first part>.<second part>*; for example, *my.namespace*. The new component without contract is created. The component includes a FlexContainer to contain any widgets that you add to the component and a Modules node, comprising the component's *Controller* and *Actions Controller* JavaScript files. The Actions Controller module is auto-generated and contains any defined action sequences.
4. Add widgets to the FlexContainer, just as you would for a standard form.
5. Add code to the component's controller module or to actions for widgets on the form. To add code to widget actions, select the FlexContainer. On the **Properties** panel, on the **Action** tab, click **Edit** for the event to which you want to add code. For more information, refer Add Actions.

Create a Component With Contract

To create a reusable component that can be published to Kony Marketplace, create a component with contract. Components with a contract use the Kony Reference Architecture: a structured, modular framework based on the Model-View-Controller (MVC) architecture. For more information on how to create a Kony Reference Architecture project, refer Create a Kony Reference Architecture Project.

To create a component with contract, follow these steps:

1. In the Project Explorer, click the **Templates** tab.
2. Right-click **Components**, point to **New**, and then select **with Contract**. The **Create new Component with Contract** dialog box appears.
3. Enter a **Namespace** and **Name** for the component. The component should follow this naming convention: *<first part>.<second part>*; for example, *my.namespace*. The new component with contract is created. The component includes a FlexContainer to contain any widgets that you add to the component and a Modules node, comprising the component's *Controller* and *Actions Controller* JavaScript files. The Actions Controller module is auto-generated and contains any defined action sequences.
4. Add widgets to the FlexContainer, just as you would for a standard form.
5. Add code to the component's controller module or to actions for widgets on the form. To add code to widget actions, select the FlexContainer. On the **Properties** panel, on the **Action** tab, click **Edit** for the event to which you want to add code. For more information, refer Add Actions.
6. Manage properties, events, and methods for the component with contract. For more information, refer Manage Properties of a Component with a Contract, Manage Events of a Component with a Contract, and Manage Methods of a Component with a Contract.

Each component's controller module contains the JavaScript code associated with the component. You can add additional modules containing any supporting code to the Modules node. The actions Controller module is auto-generated and should not be modified.

Expose a Component's Widgets

Typically, a component comprises multiple widgets. Once you create a component with a contract, you can specify which of the component's widgets to expose to users.

To expose a component's widgets:

1. Select a component's widget on the Visualizer canvas or on the **Templates** tab of Project Explorer.

   You can select a container widget or a child widget of a container widget.

2. Right-click the widget and select **Expose Widget**, or click on the **Look** tab in the **Properties** pane and set the **Expose Widget** property to *On*.

   Kony Visualizer displays the **Programmatic Key** property. Use this value to refer to the exposed widget in code. You can use the default value or specify a different programmatic key.

   If you select a container widget, all widgets within the container will be exposed. If you do not want to expose an individual widget within the container widget, select the widget and set its **Expose Widget** property to *Off*.

   > ***Note:*** You can set the **Expose Widget** property of a container widget to *Off* but expose an individual widget within the container by setting its **Expose Widget** property to *On*.

3. Repeat the process for each widget that you want to expose.

Data & Services Panel Support for Components

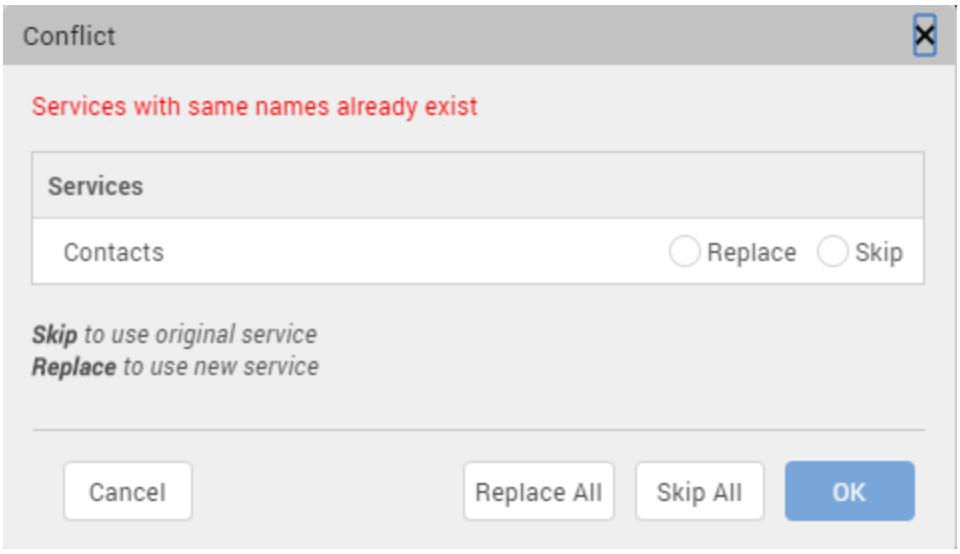
Earlier, the ability to leverage Data & Services panel features for components was not supported in Kony Visualizer. From V8 SP4 Fixpack 20, you can directly drag and drop services from the Data & Services panel to a component. In addition, you can drag and drop individual parameters of a sample service to a component. This enhancement will enable you to quickly customize components according to your requirement, and reuse them at various places in your Kony Visualizer Classic and Kony Visualizerprojects. Furthermore, you can publish customized components to Kony Marketplace. You can leverage Data Panel support for both components with contract and components without contract. For more information on the Data & Services panel, click here.

Once you drag and drop a sample service from the Data & Services panel to a component, a corresponding project service is added. That particular project service and its associated parameters are auto-highlighted in the Data & Services panel. So if multiple components exist and when you select a specific component, only those services which have been added to that component are highlighted in the Project Services of the Data & Services panel. The Data & Services panel also allows you to view the mappings for each project service operation in the component. The **onMapping** Event of the operations in the component is not generated separately in the CodeGen. You can view the onMapping Event in the **preShow** Event of the component

> ***Note:*** Currently, only the Details form and List form UI are supported for the drag and drop of services into a component; Entry form UI is not supported.

If the sample service that you are trying to add to a component has already been added to another component, a **Conflict** window appears. In such a scenario, you must perform any one of the following actions to resolve this issue:

- Select the Skip radio button, and then click **OK** to use the original service.
    - If multiple instances of the same service exist and you want to use the original service, click Skip All.
- Select the Replace radio button, and then click **OK** to use a new service.
    - If multiple instances of the same service exist and you want to use a new service, click Replace All.



This section contains the following topics:

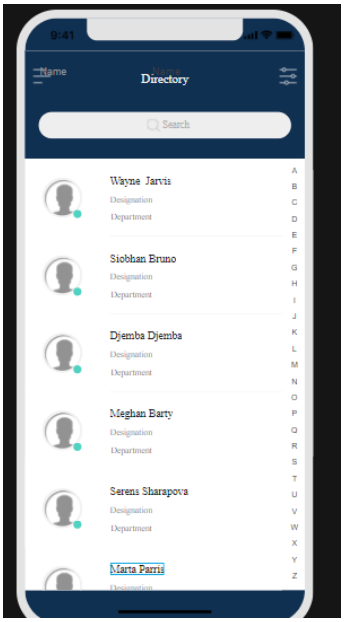
- How to Create a Component with Services
- How to Use a Component with Services

How to Create a Component with Services

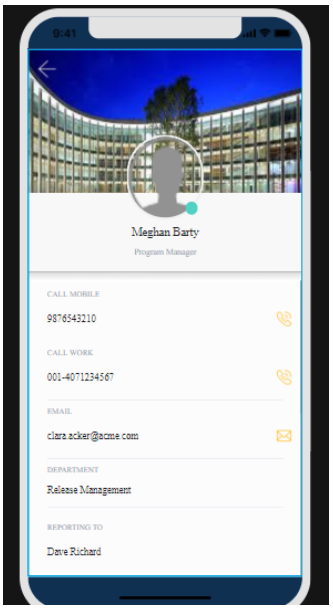
Create a component and leverage sample services from the Data & Services panel. You can bundle these services to the required widgets of a component. You can directly drag and drop services into a component and customize it according to your requirement.

In this scenario, we will xthe List-Details (Employee) component from Kony Marketplace and add services to the widgets of the component. You can add parameters (for example, **Name**) to specific widgets (for example, **empname** Label) of the component. This component contains the following two screens:

- **Employee List**



- Employee Details

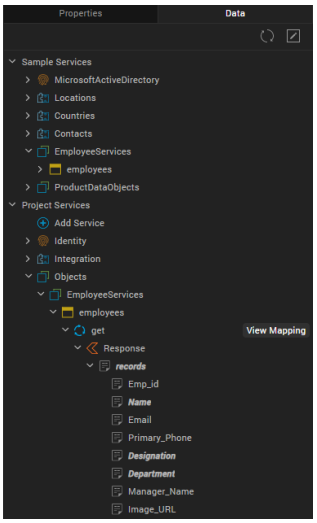


To import the List-Details (Employee) component and add services to the component, follow these steps:

1. In Kony Visualizer, create a new **Sample App** project , and then import the List-Details (Employee) component from Kony Marketplace.
   The landing page of the List-Details (Employee) component is displayed on the Project Canvas and its associated widgets are displayed in the Project Explorer.

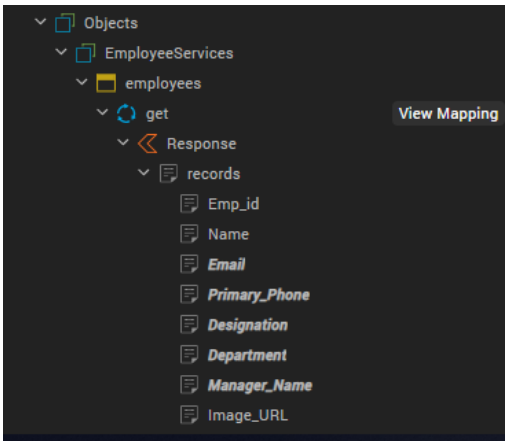
2. To add services to the Employee List screen, follow these steps:
   a. In the Project Explorer, go to **Mobile** > **Forms**, and then click **frmList**. The list screen is displayed on the Project Canvas.
   b. Go to **Data & Services** panel > **Sample Services**, and then expand **Employee Services**.
   c. Under the **employees** > **get** > **Response** > **records** operation, drag and drop the following parameters to the respective widgets:

- Name > empname
- Designation > designation
- Department > department



3. To add services to the Employee Details screen, follow these steps:
    a. In the Project Explorer, go to **Mobile** > **Forms**, and then click **frmDetails**. The details screen is displayed on the Project Canvas.
    b. Go to **Data & Services** panel > **Sample Services**, and then expand **Employee Services**.
    c. Under the **employees** > **get** > **Response** > **records** operation, drag and drop the following parameters to the respective widgets:
        - Designation > lblEmployeeDesignation
        - Primary_Phone > callMobileFlex
        - Email > emailFlex

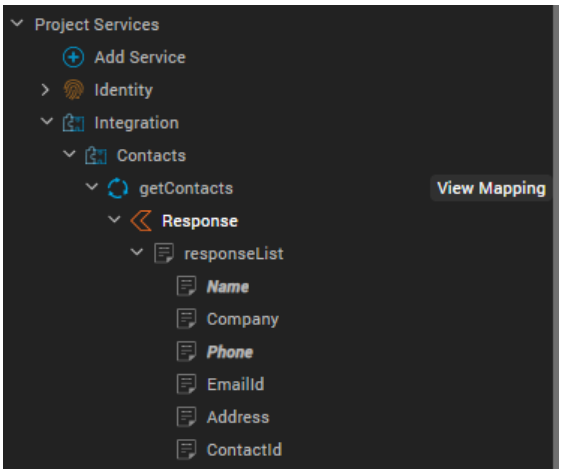
- Department > departmentFlex
- Manager_Name > reportingToFlex



d. Go to **Data & Services** panel > **Sample Services**, and then expand **Contacts**.

e. Under the **getContacts** > **Response** > **responseList** operation, drag and drop the following parameters to the respective widgets:

- Name > lblEmployeeName
- Phone > callWorkFlex



The parameters are mapped to the corresponding component widgets, and the services are created in your Kony Fabric instance. In addition, the mappings are highlighted and displayed under **Project Services** on the Data & Services panel.

4. Click **View Mapping** for a service, and then click **Generate Code** to view the respective mappings of the operations in the component. Alternatively, you can go to **Properties panel** > **Component**, click **Edit** for the **onMapping** Event, and then click **Generate Code**.

5. You can then reuse the updated component across your Kony Visualizer projects or publish this customized component to Kony Marketplace.

How to Use a Component with Services

Users can import the component with bundled services that you published on Kony Marketplace. They can then modify the component as per their requirement.

Consider a scenario where Valerie (a user who wants to create a CRM app) imports your Employee List-Detail component to Kony Visualizer. However, she does not want to view employee data. Instead, Valerie wants to view the data of her customers. She can use the Data Panel support for Components feature to map data from another service that fetches customer data to the widgets, wherever required. Valerie can leverage an Object service to fetch the data, and she can directly map the response parameters from the Data & Services panel.

Valerie can follow these steps to import the Employee List-detail component from Kony Marketplace and customize the services bundled with it:

1. Create a new project in Kony Visualizer, and then import the component to Collection Library.

2. Create a new form, say **frm1**.

3. Drag and drop the component from Collection Library to frm1. The component is added to the form and the bundled services are displayed under Project Services of the Data & Services panel.

4. For the List and Details screen, add response parameters of the required Object service (that fetches customer data) to the corresponding widgets of the component. Response parameters

of other services (such as **Contacts**) can also be used wherever necessary. The new parameters are mapped to the corresponding component widgets, and the services are created in Kony Fabric. Furthermore, the mappings are highlighted and displayed under **Project Services** on the Data & Services panel.

5. Click **View Mapping** for the service, and then click **Generate Code** to view all the mappings of the parameters in the component.
   Alternatively, you can go to **Properties panel** > **Component**, click **Edit** for the **onMapping** Event, and then click **Generate Code**.
6. This component can now be used in the CRM app to display the list of customers on the Customer List screen and specific customers' information on the Customer Details screen.

Specify a Container Widget as a Target Container

A *target container* is a Container widget that can contain other child widgets. To specify a Container widget as a target container, set both its **Expose Widget** and **Set As Target Container** properties to *On*.

To specify a Container widget as a target container:

1. Select the container widget on the Visualizer canvas or on the **Templates** tab of Project Explorer.
2. Click on the **Look** tab in the **Properties** pane.
3. Set the **Expose Widget** property to *On*, and then set the **Set As Target Container** property to *On*.

> ***Note:*** If the **Expose Widget** property is set to *Off*, setting the **Set As Target Container** property to *On* will set both the **Expose Widget** and **Set As Target Container** properties to *On*.

Kony Visualizer displays the **Placeholder** property. The Placeholder property lets you specify further information or direction to a user; for example "Add Content Here" or "Drop Image Here."

Expose a Component's Skins

In addition to specifying which of a component's widgets to expose to users, you can specify the skins to expose.

To expose a component's skins:

1. Select the component on the Visualizer canvas or on the **Templates** tab of Project Explorer.
2. Select the **Skin** tab in the **Properties** pane and select the skin that you want to expose.
3. Set the **Expose Skins** property to *On*.
4. Repeat the process for each skin that you want to expose.

Lock a Component

To prevent your component from being modified by users, you can lock the component. Users will not be able to view or modify the component's source code.

To lock a component:

1. Select the component on the Visualizer canvas or on the **Templates** tab of Project Explorer.
2. Right-click the component and select **Lock**.

Manage Properties of a Component with a Contract

Once you create a component with a contract, you can specify which properties of the component to expose to users. The specified properties are called *pass-through* properties. You can also define custom properties.

To manage properties of a component with a contract:

1. Click the **Templates** tab in the Project Explorer or the Library Explorer.
2. Expand the components node, if necessary, and then select the component.
3. In the **Properties** pane, click the **Component** tab.
4. Click **Manage Properties**. Kony Visualizer displays the **Manage Properties** dialog box.
5. To define pass through properties, select the **Pass Through** tab on the **Manage Properties** dialog box.

   Click the plus sign (+) for each pass-through property you want to define. Click the delete symbol (X) to delete an existing property. To define a new pass-through property, specify the following for each property:

   - Source Widget – The widget that contains the property. Click in the **Source Widget** field to display a hierarchical list of the component's widgets, and then select the widget.
   - Widget Property – The property that you want to define as a pass-through property. Click in the **Widget Property** field to display a list of the source widget's properties, and then select the property.
   - Display Name – The name to display in the **Component** tab of the **Properties** pane. Click in the **Display Name** field and enter a name. The display name should start with a non-numeric character, and can contain only alphanumeric characters and spaces.
   - Programmatic Name – The name to identify the property in code. Click in the **Programmatic Name** field, and enter a name. The programmatic name can contain only alphanumeric characters, and cannot start with a number or contain spaces.
   - Group – The group to which the property belongs. Use groups to display related properties in their own section in the **Properties** pane.
   - Access – Whether the property is enabled or disabled. Click in the **Access** field, and select either *Enable* or *Disable*.

   Click **Apply** to add the pass-through property.

> ***Note:*** You can also designate a widget's property as a pass-through property by navigating to the widget and right-clicking on the property name.

6. To define custom properties, select the **Custom** tab on the **Manage Properties** dialog box.

   Click the plus sign (+) for each custom property you want to define. Click the delete symbol (X) to delete an existing property. To define a new custom property, specify the following for each property:

   - Property Name – The name you want to give the property. Click in the **Property Name** field and enter a name. The property name can contain only alphanumeric characters, and cannot start with a number or contain spaces. The property name is used to refer to the property in code.
   - Display Name – The name to display in the **Component** tab of the **Properties** pane. Click in the **Display Name** field and enter a name. The display name should start with a non-numeric character, and can contain only alphanumeric characters and spaces.
   - Property Type – The data type of the property. Click in the **Property Type** field, and select a value, either *boolean*, *List Selector*, *String*, or *HTML*, *Data Grid*, or *Integer*.
   - Value – For a property with a List Selector data type, the key-value pairs that make up the list. If the property type value is *List Selector* and you click in the **Value** field, Kony Visualizer displays the **Key Value Popup** dialog box. Enter the values and click **OK**. For a *boolean* property type, the **Value** field is automatically set to *true/false*. For other property types, the value field is not applicable.
   - Default Value – The default property value. Click in the **Default Value** field, and enter a default value.
   - Group – The group to which the property belongs. Use groups to display related properties in their own section in the **Properties** pane.
   - Read/Write – Whether the property is read-only or read-write. Click in the **Read/Write** field, and select either *Read* or *Write*.

   Click **Apply** to add the custom property.

> *Important:* Unlike a pass-through property, which is based on an existing property of the component, a custom property has no built-in behavior. You must define the property's behavior programmatically. On the Templates tab, expand the component's **Modules** node, open the controller module, and add code defining the property's behavior to the controller code. For more information, see Define the Behavior of a Custom Property, Event, or Method in Code.

Manage Events of a Component with a Contract

In addition to specifying pass-through properties, you can specify which of the component's events to expose to users. You can also define custom events.

To manage events of a component with a contract:

1. Click the **Templates** tab in the Project Explorer or the Library Explorer.
2. Expand the components node, if necessary, and then select the component.
3. In the **Properties** pane, click the **Action** tab.
4. Click **Manage Events**. Kony Visualizer displays the **Manage Events** dialog box.
5. To define pass-through events, select the **Pass Through** tab on the **Manage Events** dialog box.

   Click the plus sign (+) for each pass-through event you want to define. To delete an existing event, select the event and click the delete symbol (X). To define a new pass-through event, specify the following for each event:

   - Source Widget – The widget that contains the event. Click in the **Source Widget** field to display a hierarchical list of the component's widgets, and then select the widget.
   - Event – The event that you want to define as a pass-through event. Click in the **Event** field to display a list of the source widget's events, and then select the event.

- Programmatic Name – The name to identify the event in code. Click in the **Programmatic Name** field and enter a name. The programmatic name can contain only alphanumeric characters, and cannot start with a number or contain spaces.
- Group – The group to which the event belongs. Use groups to display related events in their own section on the **Action** tab of the **Properties** pane.

Click **Apply** to add the pass through event. Kony Visualizer adds the event to the **Pass Through** section of the **Properties** pane's **Action** tab.

> ***Note:*** You can also designate a widget's event as a pass-through event by navigating to the widget and right-clicking on the event name.

6. To define custom events, select the **Custom** tab on the **Manage Events** dialog box.

   Click the plus sign (+) for each custom event you want to define. Click the delete symbol (X) to delete an existing event. To define a new custom event, specify the following for each event:

   - Raised Event – The name you want to give the event. Click in the **Raised Events** field and enter a name. The event name can contain only alphanumeric characters, and cannot start with a number or contain spaces.
   - Group – The group to which the event belongs. Use groups to display related events in their own section on the **Action** tab of the **Properties** pane.

   To add parameters to the custom event, click the **Manage Events** button, and specify the parameter information in the dialog box.

   Click **Apply** to add the custom event. Kony Visualizer adds the event to the **Custom** section of the **Properties** pane's **Action** tab.

> *Important:* Unlike a pass-through event, which is based on an existing event of the component, a custom event has no built-in behavior. You must define the event's behavior programmatically. On the Templates tab, expand the component's **Modules** node, open the controller module, and add code defining the event's behavior to the controller code. For more information, see Define the Behavior of a Custom Property, Event, or Method in Code.

Manage Methods of a Component with a Contract

In addition to specifying pass through properties and events, you can specify which of the component's methods to expose to users. You can also define custom methods.

To manage methods of a component with a contract:

1. Click the **Templates** tab in the Project Explorer or the Library Explorer.
2. Expand the components node, if necessary, and then select the component.
3. In the **Properties** pane, click the **Action** tab.
4. Click **Manage Methods**. Kony Visualizer displays the **Manage Methods** dialog box.
5. To define pass-through methods, select the **Pass Through** tab on the **Manage Methods** dialog box.

   Click the plus sign (+) for each pass-through method you want to define. Click the delete symbol (X) to delete an existing method. To define a new pass-through method, specify the following for each method:

   - Source Widget – The widget that contains the method. Click in the **Source Widget** field to display a hierarchical list of the component's widgets, and then select the widget.
   - Method – The method that you want to define as a pass-through method. Click in the **Method** field to display a list of the source widget's methods, and then select the method.

- Programmatic Name – The name to identify the method in code. Click in the **Programmatic Name** field and enter a name. The programmatic name can contain only alphanumeric characters, and cannot start with a number or contain spaces.
- Group – The group to which the event belongs. Use groups to display related events in their own section on the **Action** tab of the **Properties** pane.

Click **Apply** to add the pass-through method.

6. To define custom properties, select the **Custom** tab on the **Manage Methods** dialog box.

   Click the plus sign (+) for each custom method you want to define. Click the delete symbol (X) to delete an existing method. To define a new custom method, specify the following for each method:

   - Method – The name you want to give the method. Click in the **Method** field and enter a name. The method name can contain only alphanumeric characters, and cannot start with a number or contain spaces.
   - Group – The group to which the event belongs. Use groups to display related events in their own section on the **Action** tab of the **Properties** pane.

   To add parameters to the custom method, click the **Manage Methods** button, and specify the parameter information in the dialog box.

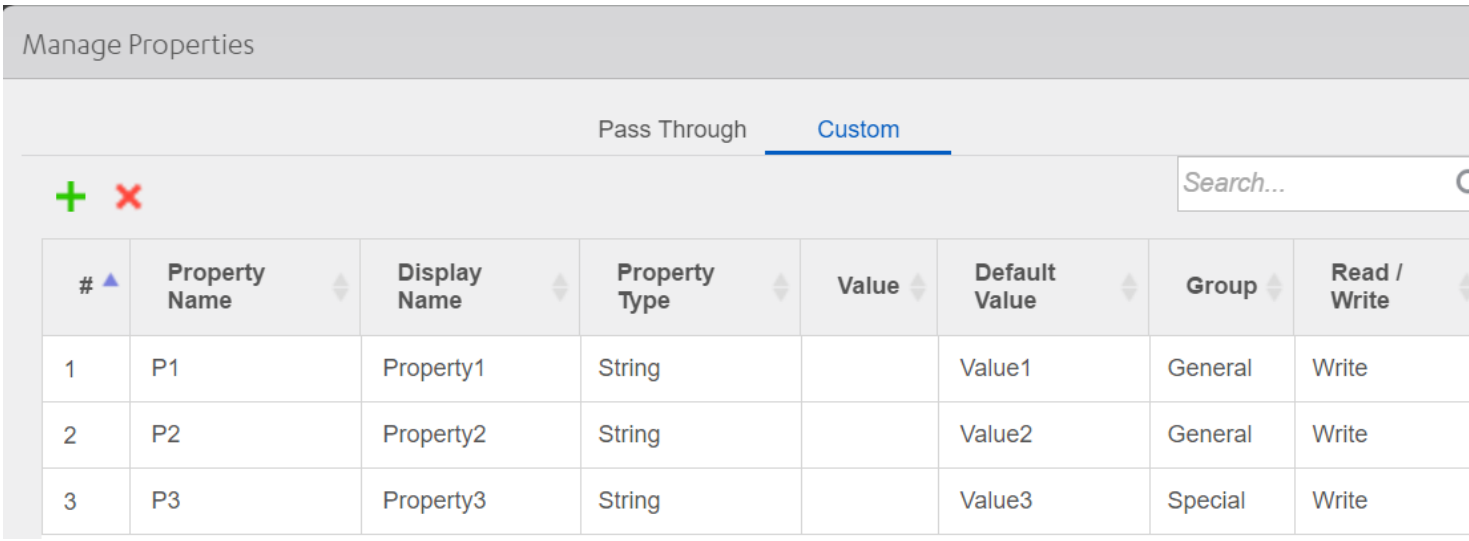
   Click **Apply** to add the custom method.

   > ***Important:*** Unlike a pass-through method, which is based on an existing method of the component, a custom method has no built-in behavior. You must define the method's behavior programmatically. On the Templates tab, expand the component's **Modules** node, open the controller module, and add code defining the method's behavior to the controller code. For more information, see Define the Behavior of a Custom Property, Event, or Method in Code.

Group Properties, Events and Methods of a Component with a Contract

When you define pass-through or custom properties, events, and methods for a component, you can specify how they are displayed in the **Properties** pane by organizing them in groups.

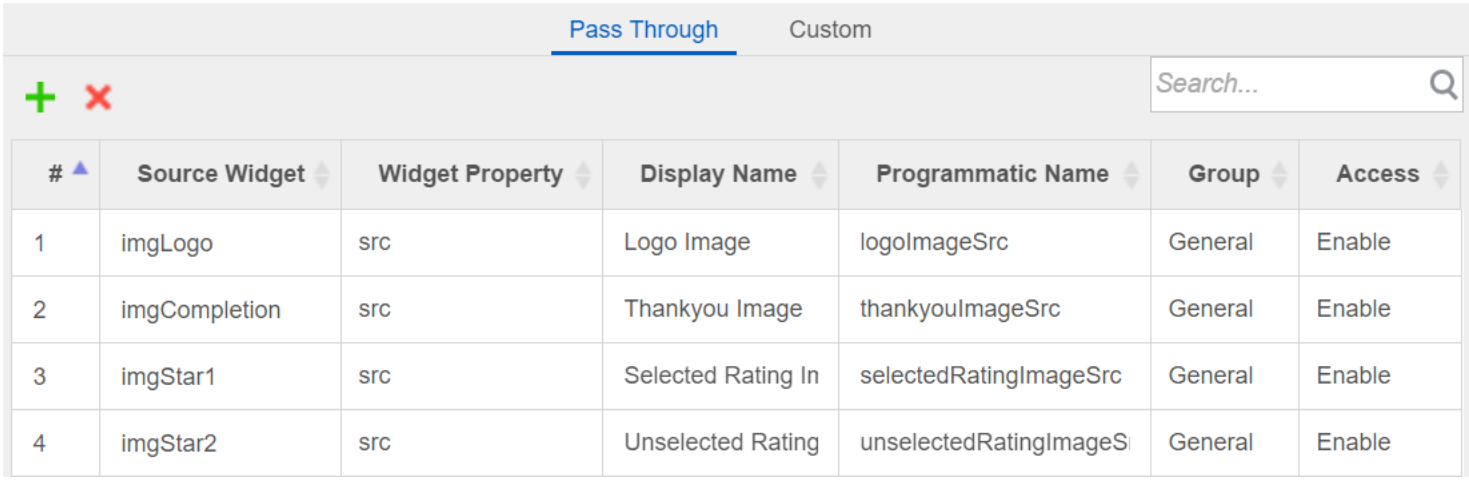
For example, you can specify that two custom properties are displayed in a group under the heading, "General," and that a third property is displayed under the heading "Special" by defining them as follows in the **Manage Properties** dialog box.



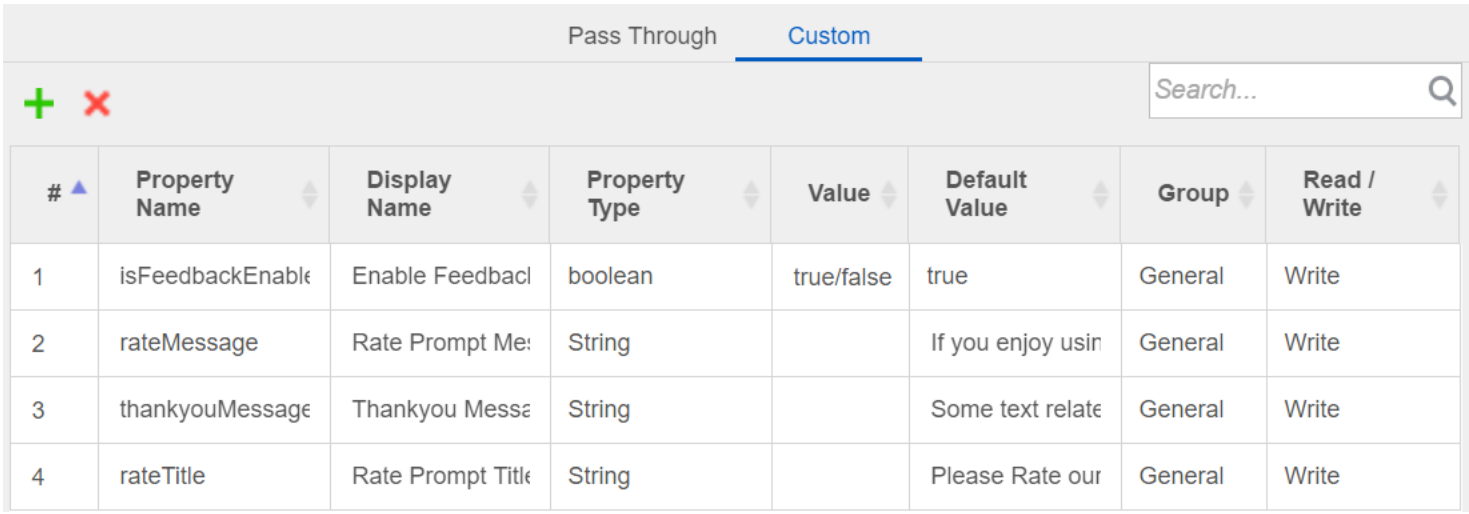
#	Property Name	Display Name	Property Type	Value	Default Value	Group	Read / Write
1	P1	Property1	String		Value1	General	Write
2	P2	Property2	String		Value2	General	Write
3	P3	Property3	String		Value3	Special	Write

The properties will be displayed in the **Properties** pane.

You can also group both pass-through and custom properties under the same heading. For example, the following General properties of the Rating Prompt component include a mixture of pass-through and custom properties:



#	Source Widget	Widget Property	Display Name	Programmatic Name	Group	Access
1	imgLogo	src	Logo Image	logoImageSrc	General	Enable
2	imgCompletion	src	Thankyou Image	thankyouImageSrc	General	Enable
3	imgStar1	src	Selected Rating In	selectedRatingImageSrc	General	Enable
4	imgStar2	src	Unselected Rating	unselectedRatingImageS	General	Enable



#	Property Name	Display Name	Property Type	Value	Default Value	Group	Read / Write
1	isFeedbackEnable	Enable Feedbacl	boolean	true/false	true	General	Write
2	rateMessage	Rate Prompt Mes	String		If you enjoy usin	General	Write
3	thankyouMessage	Thankyou Messa	String		Some text relate	General	Write
4	rateTitle	Rate Prompt Title	String		Please Rate our	General	Write

To group properties of a component with a contract:

1. Follow the steps in Manage Properties of a Component with a Contract to define a pass-through or custom property.
2. Under **Group**, specify the group where you want the property to be displayed.

   If the group name does not exist, click in the **Group** field, select **Manage Group** to open the **Manage Groups** dialog box, and then add the group name.
3. Click **Apply** to add the custom property.

To group events of a component with a contract:

1. Follow the steps in Manage Events of a Component with a Contract to define a pass-through or custom event.
2. Under **Group**, specify the group where you want the event to be displayed.

   If the group name does not exist, click in the **Group** field, select **Manage Group** to open the **Manage Groups** dialog box, and then add the group name.
3. Click **Apply** to add the custom event.

To group methods of a component with a contract:

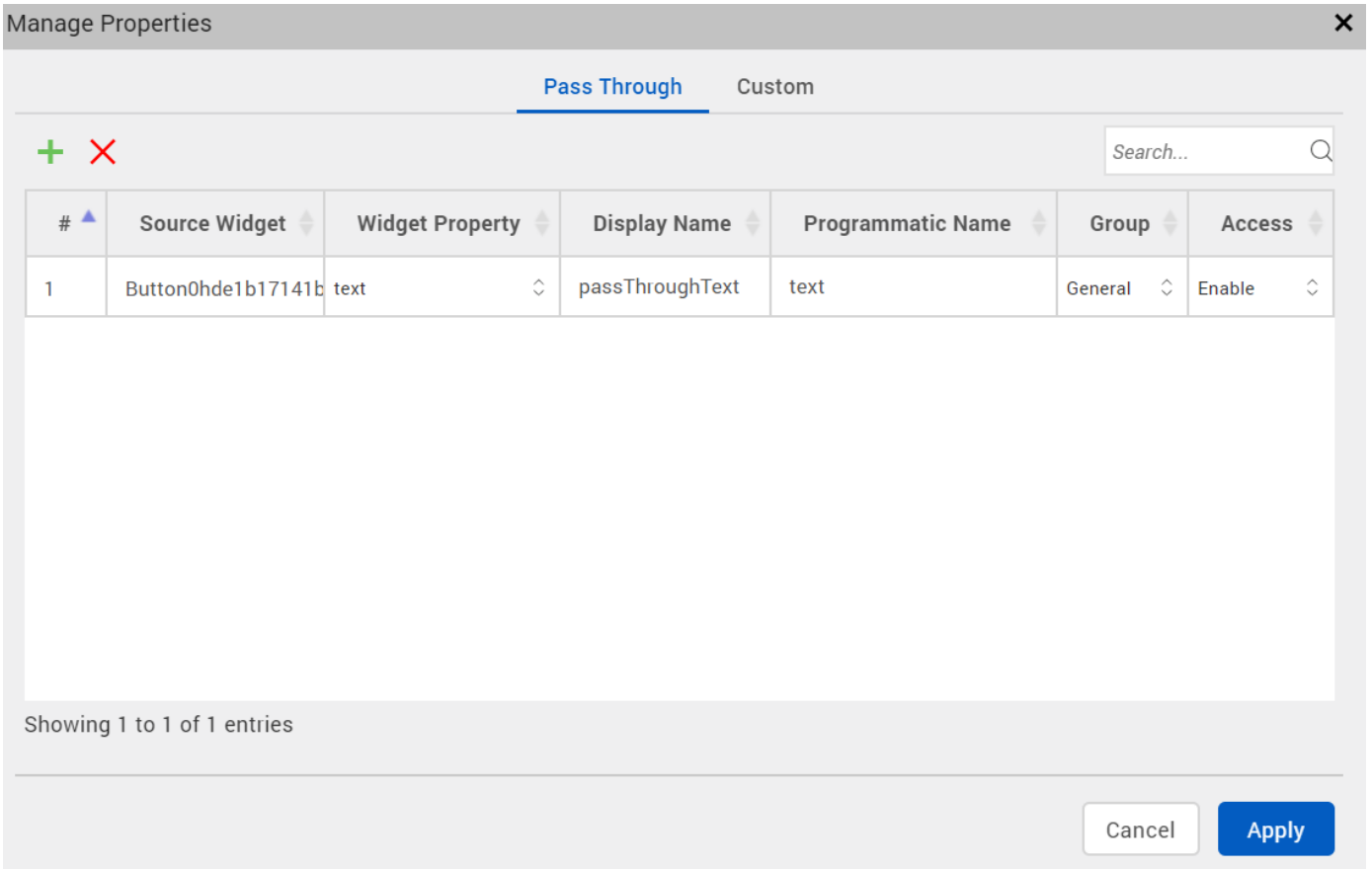
1. Follow the steps in Manage Methods of a Component with a Contract to define a pass-through or custom method.
2. Under **Group**, specify the group where you want the method to be displayed.

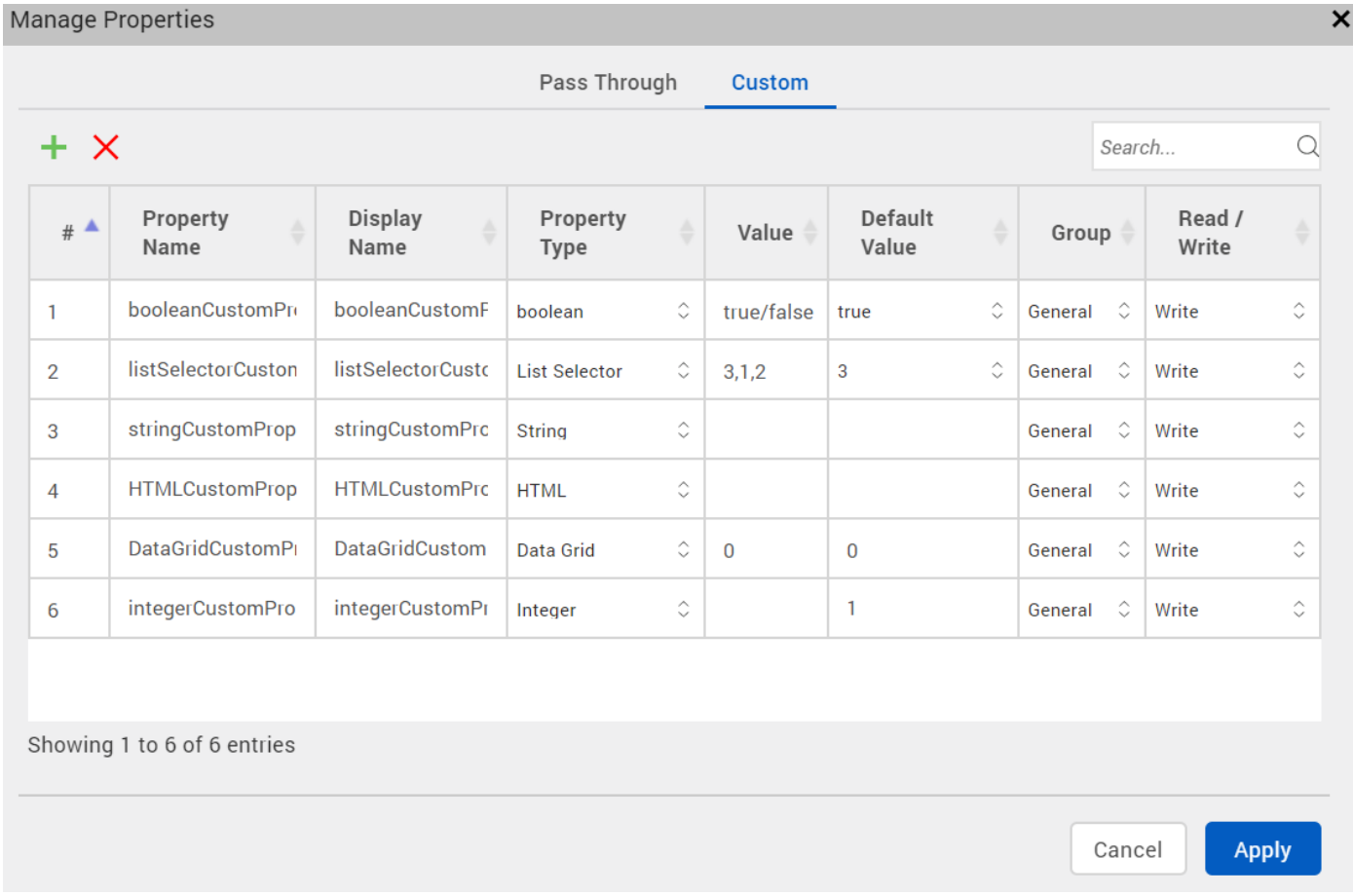
   If the group name does not exist, click in the **Group** field, select **Manage Group** to open the **Manage Groups** dialog box, and then add the group name.
3. Click **Apply** to add the custom method.

Set Data for Components with Contract by using Mapping Editor

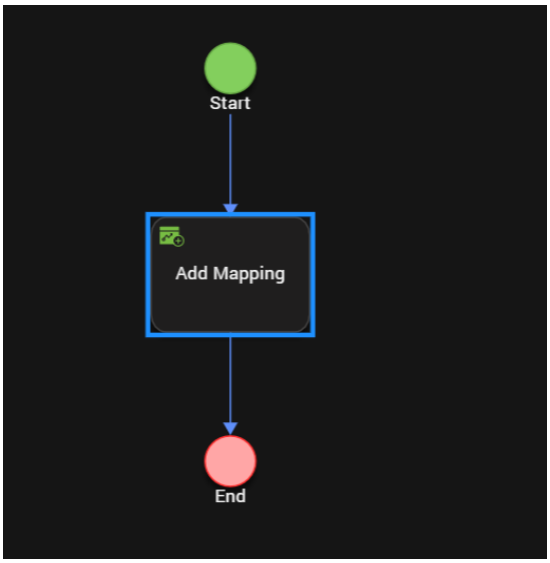
To set data for a component with contract by using Mapping Editor, follow these steps:

1. In Kony Visualizer, create a component with contract, add the required widgets, define the pass through and custom properties of the component, and then click **Apply**.



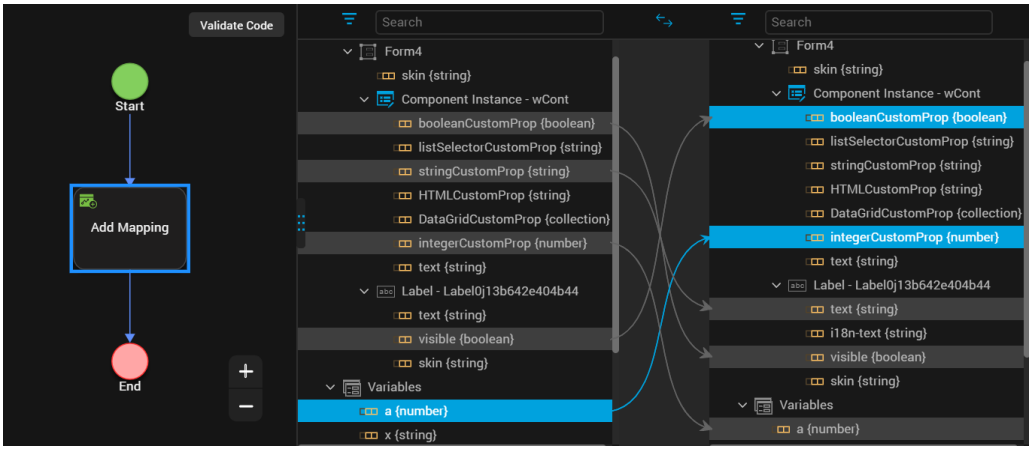


2. In Project Explorer, select the form into which you want to insert the component with contract.
3. Go back to **Templates** > **Components**, and right-click the newly created component. A list of options appears.
4. Click **Insert Into**. The component with contract is inserted into the selected form.
5. Select the form, and then go to **Properties** panel > **Action**.
6. For any Event (for example, *onMapping*), click **Edit**. The **Action Editor** window appears, with **Diagram View** open by default.
7. On the left pane of Action Editor, locate and click the **Add Mapping** action. The Add Mapping action is added to the flow diagram, as shown here.

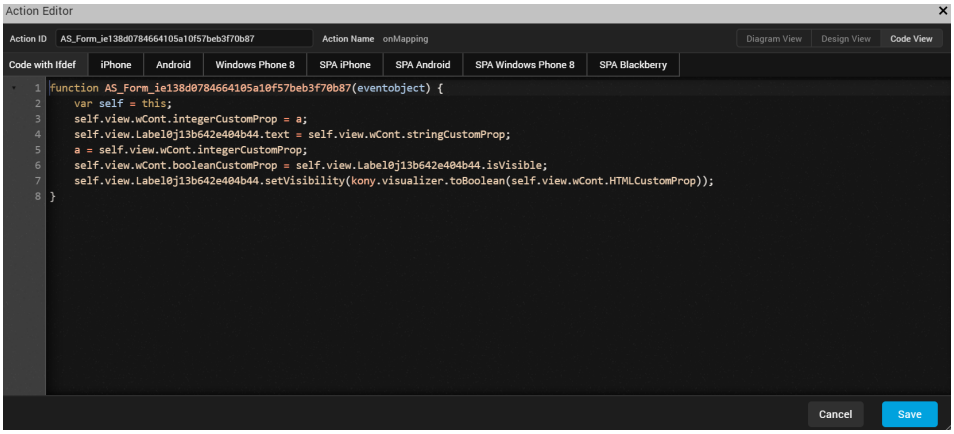


8. Select **Add Mapping** from the flow diagram. Mapping Editor opens on the right pane of Action Editor.

9. You can use Mapping Editor to expand and create mappings among various form, component, and widget elements as well as global variables.



10. Click **Code View** to see the generated sample code details of the data mappings.



11. Click **Save**. You have successfully set data for the component with contract by using Mapping Editor.

Define the Behavior of a Custom Property in Code

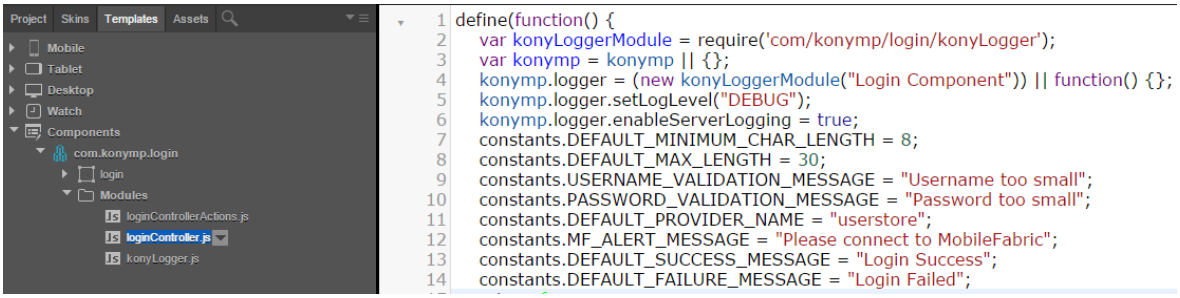
When you create a component with a contract, you can define pass-through and custom properties, events, and methods for the component. Defining pass-through properties, events, and methods does not require that you write code. You simply specify the built-in properties, events, and methods that you want to expose. However, a custom property, event, or method does not have any built-in behavior. You must define its behavior in code.

First, specify a name for the custom property, event, or method using the **Manage Properties** dialog box. For a custom property, you can also specify a display name, data type, and default value, and whether the property is read-only or read-write. If the property has a list selector, you can specify the key value pairs that make up the list. For more information on working with the **Manage Properties** dialog box, see Manage Properties of a Component with a Contract, Manage Events of a Component with a Contract, or Manage Methods of a Component with a Contract.

Once you specify information in the **Manage Properties** dialog box, you can add code to the component's controller module to define the behavior of the custom property, event, or method.

To define the behavior of a custom property, event, or method in code (Beta):

1. Specify the required information in the **Manage Properties** dialog box.
2. On the Templates tab, expand the **Modules** node of the component that contains the property, event, or method.
3. Open the component's controller module; for example, *loginController.js*.



4. Add JavaScript code that defines the behavior of the property, event, or method.

For example, the following code defines custom properties for the Login component, which can be downloaded from Kony Marketplace.

```
/**
 * @function initGettersSetters
 * @description contains getters/setters for the usernameMinimumChar
and usernameValidationMsg custom properties
 */
initGettersSetters: function() {
    defineGetter(this, "usernameMinimumChar", function() {
        konymp.logger.trace("----------Entering
usernameMinimumCharacter Getter---------", konymp.logger.FUNCTION_
ENTRY);
        return this._usernameMinimumChar;
    });
    defineSetter(this, "usernameMinimumChar", function(val) {
        konymp.logger.trace("----------Entering
usernameMinimumCharacter Setter---------", konymp.logger.FUNCTION_
ENTRY);
```

```
        try {
            if (val == null || val == undefined) {
                konymp.logger.warn("Username Min Char is undefined");
                throw {
                    "Error": "LoginComponent",
                    "message": "Username Min Char is undefined"
                };
            }
            if (isNaN(val)) {
                konymp.logger.warn("Invalid datatype for Username Min
Characters Property");
                throw {
                    "Error": "LoginComponent",
                    "message": "Invalid datatype for Username Min
Characters Property"
                };
            }
            if (this.usernameMaxChar & amp; lt; val) {
                konymp.logger.warn("usernameMaxChar is less than
usernameMinimumChar");
                throw {
                    "Error": "LoginComponent",
                    "message": "username Max Char is less than
Username Min Character propert"
                };
            }
            this._usernameMinimumChar = val;
        } catch (exception) {
            if (exception["Error"] === "LoginComponent")
                alert(JSON.stringify(exception));
        }
    });
    defineGetter(this, "usernameValidationMsg", function() {
```

```
        konymp.logger.trace("----------Entering usernameValidationMsg
Getter---------", konymp.logger.FUNCTION_ENTRY);
        return this._usernameValidationMsg;
    });
    defineSetter(this, "usernameValidationMsg", function(val) {
        konymp.logger.trace("----------Entering usernameValidationMsg
Setter---------", konymp.logger.FUNCTION_ENTRY);
        this._usernameValidationMsg = val;
    });
}
```

Define a Custom Event

Creating and using a Custom Event involves three stages, creating a custom event, invoking the custom event, and consuming the custom event.

To create, consume, and invoke a custom event, do the following:

1. In your Visualizer Project, from the **Project Explorer** section, click **Templates** tab.
2. From **Components**, select **Create new Component with Contract**.
3. Enter the **Namespace**
4. Enter **Name**
5. Click **OK**.
6. Navigate to **Project**
7. **Navigate to Mobile > Forms > New Form.** A new form is created.
8. Navigate to the **Templates** tab, and drag and drop the new template you created earlier onto this form. Your component is created, and it is added to the form. Now, let us define a custom event.
9. Navigate to the component you created.

10. In the **Properties** pane, click **Action**.
11. Click **Manage Events**. The Manage Events pane displays.
12. Click **Custom** tab.
13. Click on the **+** sign to add a new custom event.
14. In the **RaisedEvents** column, enter a name for your event. For example, testEvent.
15. Click **Apply**. The event is created, and you can view it in the **Action** tab under **General**. Now, the custom event is defined. Let us now invoke the custom event.
16. In your component, add a button.
17. Name the button. For example, Trigger Event.
18. In the **Properties** pane of the button, navigate to the **Action** tab and click onClick event **Edit** button. The Action Editor opens.
19. From the list of functions available, select **Raise Event**.
20. From the **Function Name** dropdown, select **testEvent**.
21. Close the Action Editor. Now, let us consume the event.
22. Navigate to your form on your Mobile channel.
23. On the form, select the component and from the **Properties** pane, click **Action** tab.
24. Under **General**, you will notice the testEvent you created. Click **Edit**. The Action Editor opens.
25. From the Functions list, select **Add Snippet**.
26. In the code pane, enter `alert("Yay! Custom event");`
27. Close the action editor.
28. From the File Explorer menu of Visualizer, from **Run**, select **Run**.
29. The Building pane appears.

30. Navigate to your local preview. For more information on how to preview your app locally, click here. You will see your form in local preview in chrome.
31. Click **Trigger Event**.
    An alert Yay! custom event is displayed.
32. Click **OK** to close the alert.

Adding Functionality

Applies to *Kony Visualizer Classic*.

You can add functionality to your application by adding actions using the Action Editor, by using services exposed by a URL, and by developing offline applications.

Add Actions

Add Services and Data Sources

Developing Offline Applications

Connect to Services

A *Service* is an application component that represents the application's interaction with the external service data source. A service definition comprises the metadata or the configurations required to exchange data with the external data source. For example, the configurations can be: service type, service ID, input parameters, output parameters, pre and post processors, target URL, authentication credentials (if required), and type (HTTP/HTTPS).

You can define the services for the following connectors:

- XML over HTTP
- Simple Object Access Protocol (SOAP)
- Java - Custom connector for any other data source
- Composite - A combination of all the above

The following services are supported in Kony Fabric:

- XML
- SOAP
- JSON
- Java
- JavaScript
- API Proxy
- Mock Data
- Kony SAP Gateway
- MuleSoft
- AWS API Gateway

- Database
- MongoDB
- RAML
- SAP JCo
- IBM MQ
- Salesforce
- Open API (Swagger)

Preprocessors and Postprocessors

Applies to *Kony Visualizer Classic*.

A preprocessor is a component that is invoked before passing the data to the external data source. This enables the developer to include any business logic on the data before forwarding the request to the external data source.

A postprocessor is a component invoked after the data is received from the external data source but before that data is returned to the mobile device. This enables the developer to include any business logic on the data before sending the response to the mobile device.

The logic, in both cases, may include formatting the data elements, removing them, or logging them (for diagnostics purposes). As custom Java classes, preprocessor and postprocessor implement the `com.konylabs.middleware.common.DataPreProcessor` and `com.konylabs.middleware.common.DataPostProcessor` Java interfaces respectively. You need to implement and load these libraries before using them.

> ***Important:*** The preprocessor or postprocessor name must start with `com.kony` or `com.konylabs`.

Both preprocessors and postprocessors can perform the following tasks on the data that is passed to a service, or data that is retrieved from a service call:

- Filter the data.
- Change the format of the data.
- Add or delete data.

If you have a preprocessor defined in a service call, the preprocessor library is invoked first before making the service call.

You can browse and select the preprocessor or postprocessor libraries in the Service Definition Editor for all the types of services.

> ***Note:*** Invoking a preprocessor or a postprocessor within any type of service is optional, depending on the needs of your application.

Best Practices

In writing preprocessors and postprocessors, it is recommended that you follow these best practices:

- For logging, use log4j statements instead of System.out.println() statements.
- Wrap the debug statements with the log.isDebugEnabled() method. For example,

```
if (logger.isDebugEnabled()) {
        ...logger.debug("printing debug info")
}
```

Sample Preprocessor

The following is a sample preprocessor file.

```
import java.util.HashMap;
import org.apache.log4j.Logger;
import com.konylabs.middleware.common.DataPreProcessor;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.dataobject.Result;
import com.konylabs.middleware.session.Session;
public class TestPreProcessor implements DataPreProcessor {
        ...public boolean execute(HashMap hm, DataControllerRequest
dcRequest, Result result) 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 Postprocessor

The following is a sample postprocessor file.

```
import java.util.ArrayList;
import org.apache.log4j.Logger;
import com.konylabs.middleware.common.DataPostProcessor;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.dataobject.Param;
import com.konylabs.middleware.dataobject.Result;
public class TestPostProcessor implements DataPostProcessor {
        ...public Object execute(Result results, DataControllerRequest
dcRequest)
        ... {
                ...... //Write application logic here to modify the
results returned from the service
            ......
                ...... // return Result object here
            ...
            return results;
                ...
    }
}
```

Data Provider

Applies to *Kony Visualizer Classic*.

With the DataProvider interface, you can customize how you establish a connection and generate a response to a third party service. By reading the service definition, DataProvider forms the request metadata and then parses the response as per the service definition. The *DataProvider.execute()* method is implemented such that after execution, the *DataControllerResponse object* is populated with the response, CharacterEncoding, and status code.

Sample Data Provider

The following is a sample DataProvider file.

```
import java.io.File;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import com.konylabs.middleware.common.DataProvider;
import com.konylabs.middleware.controller.DataControllerRequest;
import com.konylabs.middleware.controller.DataControllerResponse;
import com.konylabs.middleware.exceptions.ConnectorException;
public class SampleDataProvider implements DataProvider {@
    Override
    public void execute(String arg0, DataControllerRequest arg1,
        DataControllerResponse DCresponse, String arg3, Map < String,
String > arg4,
        List < File > arg5, HashMap arg6) throws ConnectorException {
        System.out.println("::::::::entered:::SAMPLE DATAPROVIDER:");
        System.out.println("::::::::DataController Request::::" +
arg1.toString());
        System.out.println("::::::::DataController Request::::" +
arg0.toString());
        //System.out.println("::::::::DataController
Request::::"+arg3.toString());
```

```
        //Respose generation
        String xmlResult = "" +
            "<soap:Envelope
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
            "<soap:Body>" +
            "<AllPlayerNamesResponse
xmlns:m=\"http://footballpool.dataaccess.eu\">" +
            "<AllPlayerNamesResult>" +
            "<tPlayerNames>" +
            "<iId>1434</iId>" +
            "<sName>Lucas Barrios</sName>" +
            "<sCountryName>Winner SF-II</sCountryName>" +

"<sCountryFlag>http://footballpool.dataaccess.eu/images/flags/</sCoun
tryFlag>" +
            "</tPlayerNames>" +
            "<tPlayerNames>" +
            "<iId>715</iId>" +
            "<sName>Alexander Frei</sName>" +
            "<sCountryName>Winner SF-II</sCountryName>" +

"<sCountryFlag>http://footballpool.dataaccess.eu/images/flags/</sCoun
tryFlag>" +
            "</tPlayerNames>" +
            "</AllPlayerNamesResult>" +
            "</AllPlayerNamesResponse>" +
            "</soap:Body>" +
            "</soap:Envelope>";
        //Data Response setting
        DCresponse.setResponse(xmlResult);

        DCresponse.setCharsetEncoding("UTF-8");
```

```
        DCresponse.setStatusCode(200);
        //System.out.println("::::::::DataController
Request::::"+DCresponse.toString());
    }
}
```

Response Encoding Schemes

Applies to *Kony Visualizer Classic*.

The response fetched from a Web Service or an XML Service from the external data source follows a certain encoding scheme to support different languages or character sets. The response can be encoded using the following formats:

Encoding Algorithm	Description
UTF-8	UCS (Unicode Code Space) Transformation Format - 8 bit. Represents every character in the Unicode character set. This is the default encoding format.
ISO-8859-1	Encoding scheme to support the first set of Latin alphabet, which contains 191 characters
US-ASCII	Encoding scheme based on the sequence of characters in the English alphabet
UTF-16	16-bit UTF encoding scheme
UTF-16BE	UTF 16 bit Big Endian. Encoding scheme that maps Unicode character set to a sequence of 2 bytes (16 bits)
UTF-16LE	UTF 16 bit Little Endian. Encoding scheme that maps Unicode character set to a sequence of 2 bytes (16 bits)
Windows-1250	Encoding scheme (under Microsoft Windows) for languages that use Latin script
Windows-1251	8-bit encoding scheme for languages that use Cyrillic alphabet
Windows-1252	Encoding scheme for the Latin alphabet

Encoding Algorithm	Description
Windows-1253	Encoding scheme for modern Greek language
Windows-1254	Encoding scheme for Turkish language
Windows-1255	Encoding scheme for Hebrew language
Windows-1256	Encoding scheme for Arabic language
Windows-1257	Encoding scheme for Estonian, Latvian, and Lithuanian languages under Microsoft Windows
Windows-1258	Encoding scheme for Vietnamese texts

Input and Output Parameters

Applies to *Kony Visualizer Classic*.

Input parameters are the parameters that you pass to the service for use during a service call. Some of the services may not have input parameters. Input parameters are used when dynamic content is passed to the external data source.

Output parameters are the parameters that are fetched from the response of a service call. These are formatted according to the attributes you configure for the output before displaying on the device.

The service parameters have a scope and data type attached to them. While defining the input and output parameters for a service, you need to define the scope and the data type.

Input Parameters

The following attributes are defined for **Input Parameters**:

1. **ID**- Unique identifier for a parameter.
2. **Test Value** - The value to be used to test the service within the Service Definition Tool. This value is not the part of the final Service Definition file.
3. **Scope** - The scope can be *request* or *session.*
    - **request** - Indicates that the value has to be retrieved from the http request received from the mobile device.
    - **session** - Indicates that the value has to be retrieved from the http session stored on the Kony Application Server.
4. **Datatype** - Type of the data. These datatypes are defined here and are used for mapping appropriately in the Mapping editor. It can belong to the following data types:
    - **string** - Represents a combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit.
    - **boolean** - Represents a value that can be either true or false.

- **number** - Represents an integer or a floating number.
- **collection** - Represents a group of data, also referred to as data set.

  This value is received from the client as a JSON string.

  > ***Important:*** Datatypes Boolean, number, and collection are not available in Java Services.

5. **Encode** - *True* or *False*. Specifies if the URL needs to be encoded or not. URL encoding converts the characters into a format that can be safely transmitted over the Internet. For example, if the word **New York Times** is to be passed as part of the URL, it would be encoded as **New%20York%20Time** when the encoding is set to True adhering to the HTML URL encoding standards.

Output Parameters

The following attributes are defined for **Output Parameters**:

1. **ID** - Unique name that we use in our code to access this value.
2. **Xpath** - Xpath is applied for extracting the required elements from the response of the service call. XPath is used to navigate through elements and attributes in an XML document. For more information about *XPath*, see http://www.w3schools.com/xpath/.
3. **Scope** - The scope can be *response* or *session.*

   - **response** - Indicates that the parameter is included in the final response that is sent to the mobile device.

     > ***Note:*** Mark only those parameters that are represented on the device UI as response scope.

   - **session** - Indicates that the parameter gets stored in the session on the Kony Application Server, that is, this parameter need not be included in the response that is sent to the

mobile device.

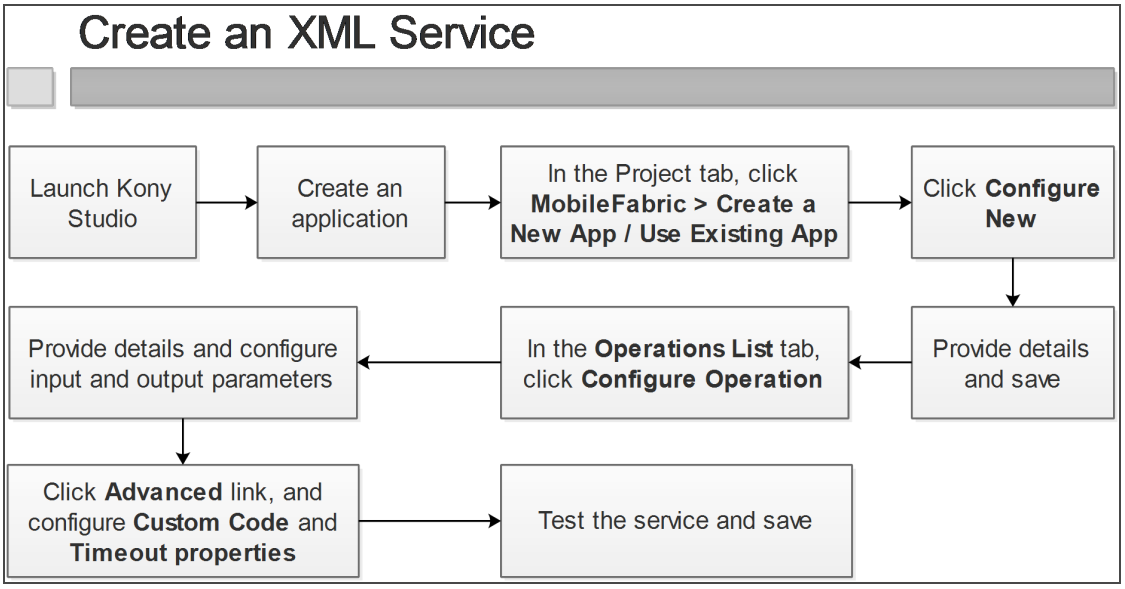
> ***Note:*** It is a best practice to mark the session tokens, apart from cookies returned from the data source, as session scope.

4. **Datatype** - The following data types are available:
    - **string** - Represents a combination of alpha-numeric and special characters. Supports all formats including UTF-8 and UTF-16 with no maximum size limit.
    - **Boolean** - Represents a value that can be either true or false.
    - **number** - Represents an integer or a floating number.
5. **Collection ID** - Groups data elements under the specified parameter as a collection. Collection is created to assign grouped data to segment or a table in the UI.
6. **Record ID** - Groups data elements under the specified parameter as a record. Typically we use this to provide metadata to the segment.
7. **Format**- Can be set to None, currency, number or date.
8. **Format Value** - Standard for converting to the specified format.
    - **Number** - for number format refer to **java.text.DecimalFormat**.
    - **Currency** - for currency format refer to **java.text.DecimalFormat**
    - **Date** - for date format refer to **java.text.SimpleDateFormat**

Create an XML Service

Applies to *Kony Visualizer Classic*.

A service that communicates with an external data source using an XML data connector over the HTTP protocol is known as an XML service, and you can construct your application to use multiple XML web services from various sources that work together. Configured using the Kony Fabric Console, XML web services manages the interface between the requests of your app to an XML based data source and the data source's responses to those requests.



An XML Service Scenario

The remainder of this topic describes how you would use an XML service to retrieve data from a web site, such as a news feed. In this scenario, the following takes place:

- In the digital application, the user enters a search term (i.e. a keyword) directed to the URL of a news site.
- The application invokes the designated XML service and hits the URL defined in the service with the keyword as an input parameter.

- The service fetches the response and the response is sent back to the application.
- The response received is displayed on a form to the user.

The following procedures describe how you would configure an XML service to carry out this scenario.

Create an XML Service

Publish the Service

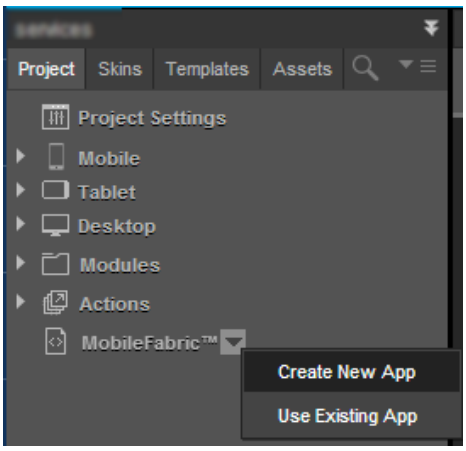
Map the Service Output to Widgets on a Form.

Create an XML Service

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

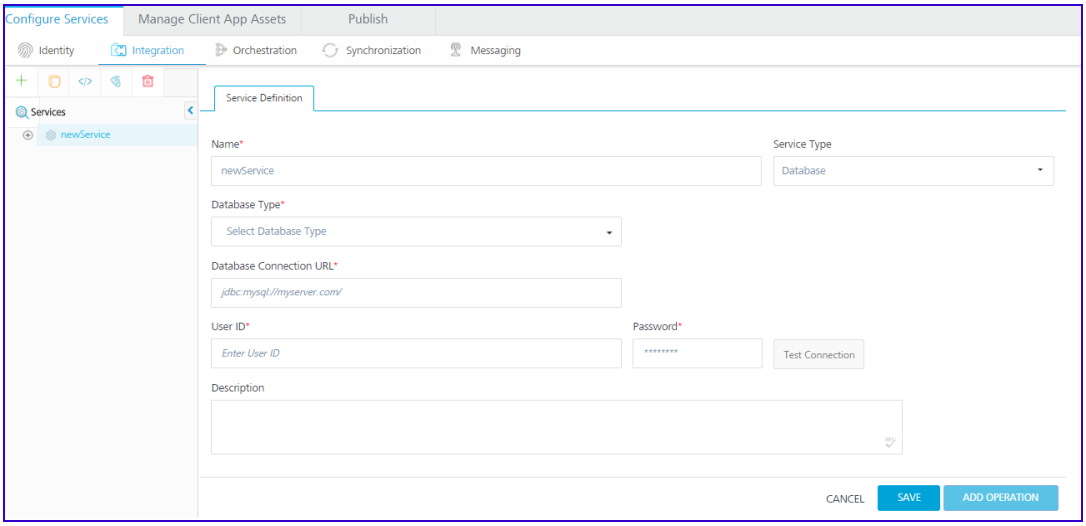
To create an XML service, do the following:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.



> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.



5. In the **Name** box, enter a unique name for your service.
6. From the **Service Type** drop-down list, click **XML**.
7. In the **Base URL** text box, type the URL from which the XML services are extracted.

8. From the **Client Authentication** drop-down list, select a value.
9. In the **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 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.

   > ***Important:*** If you configure an integration service with Basic web service authentication, ensure that reserved IDs are not used as input or header IDs since some key words such as userid, pwd and password are reserved by middleware services.

10. Click the **Advanced** tab.
11. If you want to specify a JAR file to associate with this 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.
12. If you want other identity services associated with your app to allow this service to access the identity session of the end-user at runtime, click the **Select Dependent Identity Services** drop-down menu and click the check box next to each identity service you want. This enables any operation to retrieve backend security tokens or other user profile data received during authentication and use it as part of the request sent to the backend target.
13. If you want to use **API throttling** 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.

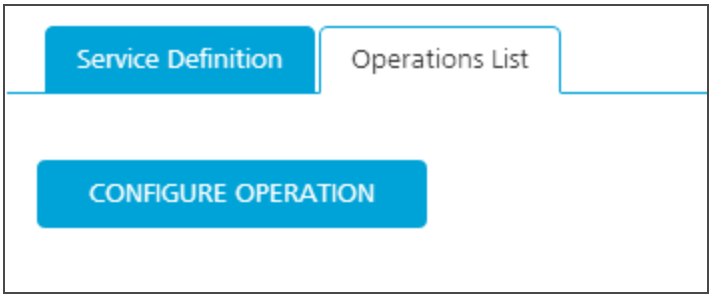
    ii. In the **Rate Limit Per IP** field, enter a required value. With this value, you can limit the

number of IP address requests configured in your Kony Fabric console in terms of Per IP Rate Limit.

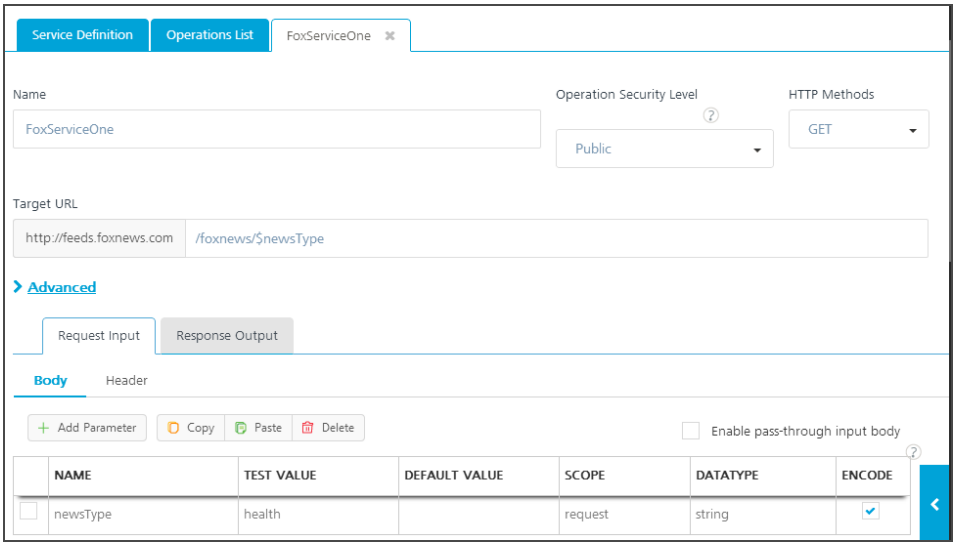
To override throttling, refer to Override API Throttling Configuration.

> ***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.

14. Enter the qualified name of the URL Provider Class. For more information, refer URL Provider Support for XML, JSON, SOAP, and API Proxy.
15. Click **Save**. A new tab titled **Operations List** appears. Click it.



16. Click **CONFIGURE OPERATION**. The **New Operation** tab appears.



17. In the **Name** box, provide a name for the operation.
18. Select a security level from the **Operation Security Level** list. By default, this field is set to **Authenticated App User.**

    You can restrict access to this operation based on the following levels:

    - **Authenticated App User.** Indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.
    - **Anonymous App User.** Indicates that a user must have the app key and app secret to access this operation.
    - **Public.** Indicates that this operation requires no special security.
19. From the HTTP Methods drop-down list, select a method for the operation.
20. If necessary, in the **Target URL** text box, modify the URL by adding any needed suffix.
21. On the **Request Input** tab, add parameters and provide values for them, ensuring that they correspond to the form elements on your digital app that the user has entered or selected. For example, in the case of a news feed scenario, these values might be as follows:

Name	Test Value	Data Type
newsType	health	string

> ***Note:*** These values are specific to the example given here. You must do the mapping between the service parameters and the form elements to capture the values that the user enters and pass them to the service.

22. On the **Request Output** tab, add parameters and provide the following details:

Name	Path	Scope	Data Type	Collection ID
channel1	//channel	response	collection	
title	item/title	response	string	channel1
description	item/description	response	string	channel1
pubDate	item/pubDate	response	string	channel1
link	item/link	response	string	channel1

23. Click **Advanced**, and then enter the necessary values for the following parameters:

    Custom Code Invocation: Upload a JAR file containing the pre-processor class name and post-processor class name. This step allows you to further filter the data received from a service call.

    > ***Note:*** For more information on Pre-processor and Post-processor, see the section Preprocessor/Post Processor. Invoking a pre-processor or a post-processor within any type of service is optional. The need to invoke these libraries depends on your requirement.

    HTTP Headers: You can provide the HTTP Headers for the call.

    Properties: You can configure various advanced service properties.

Timeout (in ms) - Specify the maximum time (in milli-seconds) the service waits for a response, before terminating the connection to the external data source.

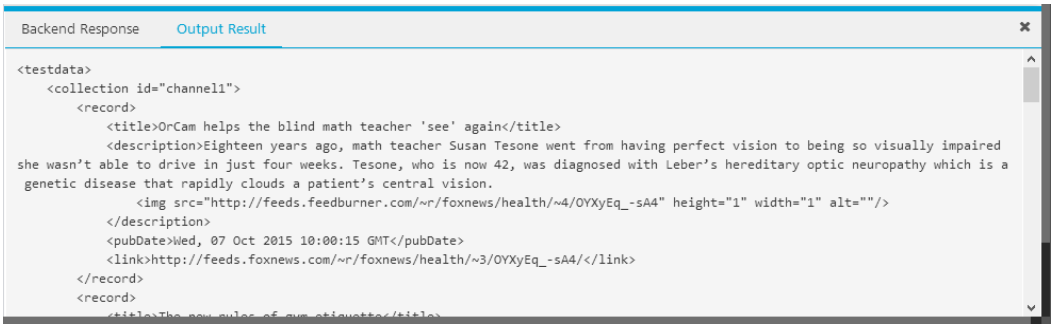
Cacheable - Specify if the service response must be stored in the cache.

Select appropriate **Response encoding** scheme. The default value is UTF-8. For more information about different encoding schemes supported by Kony Visualizer, see Response Encoding Schemes.

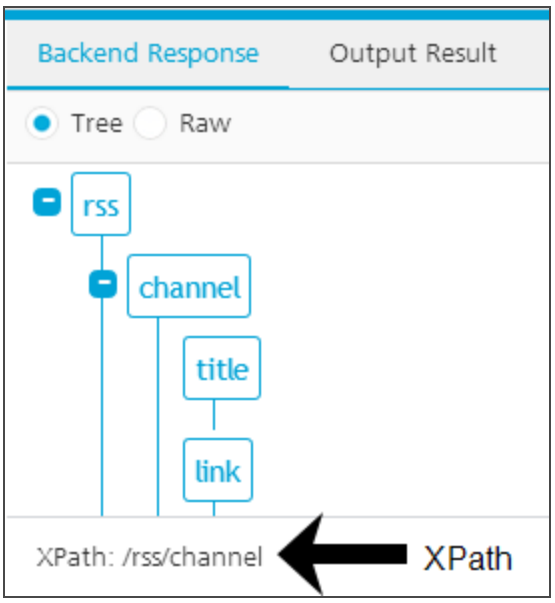
Cacheable Duration - This option is enabled if you select the **Cacheable** property. This property specifies the time duration in seconds.

> ***Note:*** If this service is **Cacheable** and a call is made before the specified time duration, the service response is fetched from the Cache, else a fresh service call is made to fetch response from the server.

24. Scroll down to the bottom of the window, and then click **Save Operation**.
25. To view the results of the operation, click **Test**. The results are displayed in the **Output Result** section.



In the **Backend Response** section, you can view the back-end response as either raw data or in a tree format. Clicking an element on the tree displays the Xpath of that tag.



26. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Create a Web (SOAP)/SharePoint Service

Applies to *Kony Visualizer Classic*.

A service that communicates with an external data source using a SOAP data connector over the HTTP protocol is known as an SOAP service, and you can construct your application to use multiple SOAP web services from various sources that work together. Configured using the Kony Fabric Console, SOAP web services manage the interface between the requests of your app to an SOAP based data source and the data source's responses to those requests.

To create the service definitions for an external data source providing SOAP interface, use the Web Service (SOAP) service definition tool. You need to have a WSDL URL or file to create the service definition.

A SOAP Service Scenario

The remainder of this topic describes how you would use an SOAP service to retrieve data from a web site, such as a weather report. In this scenario, the following take place:

1. In the digital application, the user enters a zip code (a keyword) directed to the URL of a weather website. The device initiates the service request.
2. The response is fetched from the SharePoint service and is sent to the application.
3. The response is displayed on a form to the user.

To view a video on using web services, see Using Web Services.

> ***Notes:***
> - After you provide the SharePoint server's IP address and port, Kony Visualizer loads all WSDLs exposed by SharePoint server. You can browse the WSDLs and their services associated with WSDLs. Users can view the WSDL service input and output parameters in the Web service definition files.
> - The input and output parameters can be classes or primitive types.

> - If you have defined the credentials of the SharePoint server, the credentials will show up within the **Web Services** folder. Expand the Web Services/SharePoint server folder and click the service you want to add.

The following procedures describe how you would configure a SOAP service to carry out this scenario.

Create a SOAP Service

Publish the Service

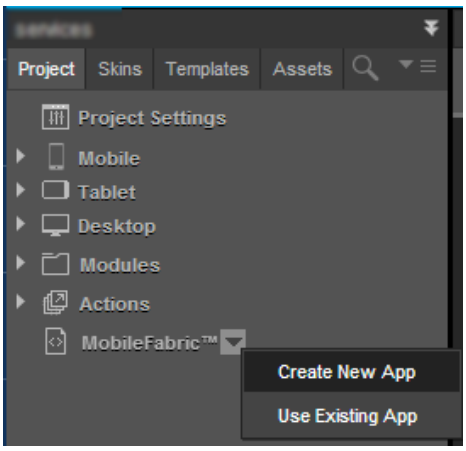
Map the Service Output to Widgets on a Form.

Create a SOAP Service

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

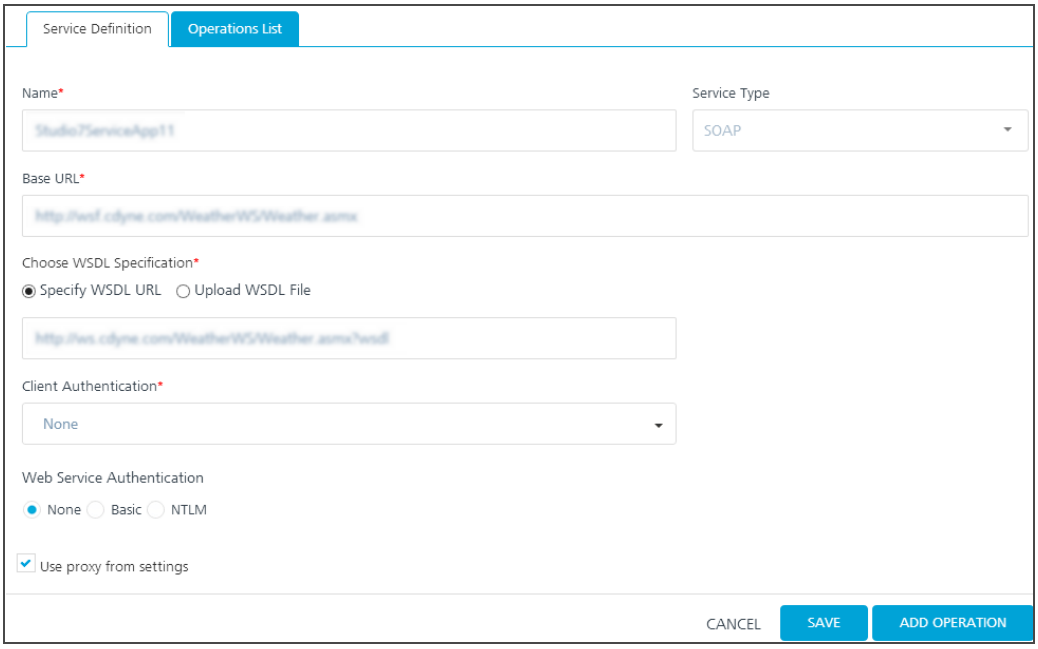
To create a SOAP service, follow these steps:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.



> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.



5. In the **Name** box, provide a unique name for your service.

6. Select **SOAP** from the Service Type list.

7. In the **Base URL**, type or paste the URL from which the SOAP services are extracted.
8. In the **Client Authentication** list, select a value.
9. Click the **Advanced** tab to specify dependent JAR and API throttling. All options in the Advanced section are optional.
    - **To specify dependent JAR, follow these steps**:

      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. The 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.

    - **API throttling** enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, the API will not return the service response.
      To specify throttling, 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, refer to Override API Throttling Configuration.

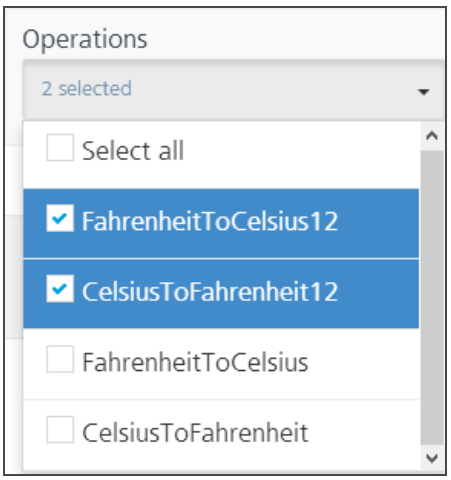
> ***Note:*** In case of On-premises, the number of nodes in a clustered environment is set by configuring the `KONY_SERVER_NUMBER_OF_NODES` property in the Admin Console. This property indicates the number of nodes configured in the cluster. The default value is 1.
> Refer to The Runtime Configuration 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.

- Enter the qualified name of the URL Provider Class. For more information, refer URL Provider Support for XML, JSON, SOAP, and API Proxy.

10. In the **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 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.

    > ***Important:*** If you configure an integration service with Basic web service authentication, ensure that reserved IDs are not used as input or header IDs since some key words such as userid, pwd and password are reserved by middleware services.

11. Click **Save**. A new tab titled **Operations List** appears. Click it.
12. Select an operation from the list of operations.



13. Click **Add Operation**. The operation is added to the service.

> ***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.

14. Click the added service.
15. Select one of the following security operations in the Operation Security Level field. By default, this field is set to Authenticated App User. You can restrict access to this operation based on the following levels:
    - Authenticated App User - indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.
    - Anonymous App User - indicates that a user must have the app key and app secret to access this operation.
    - Public - indicates that this operation requires no special security.
16. In the **Operation Path** box, modify the path if required.
17. Select a method for the operation from **HTTP Methods** list.

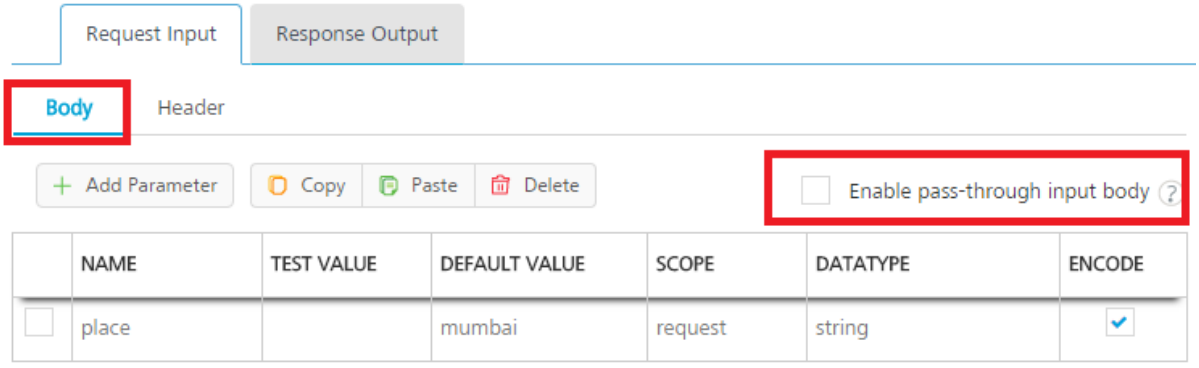
18. Click **Request Input** and do the following.

    Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

    > ***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.

    a. Under the **Body** tab, do the following:

    - 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.



    - To configure parameters in the client's body, do the following.

        i. In the **NAME** field, enter the name for the request input parameter.

        ii. **VALUE:** select request or session. By default, this field is set to **Request.** three different options are available in Kony Fabric under **Request Input >**

Body > **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.

- **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.

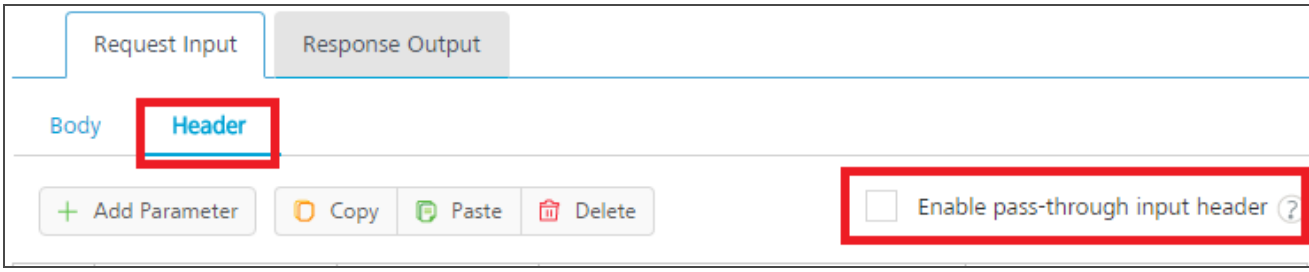
iii. **TEST VALUE**: Enter a value. A test value is used for testing the service.

iv. **DEFAULT VALUE:** enter the value if required. The default value will be used if the test value is empty.

v. Select the **ENCODE** check box 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 to the HTML URL encoding standards.

b. Click the **Header** tab to provide the following customer headers:

   Based on the operation - for example, post or get -, provide custom HTTP headers. To provide customer headers, click **Header**.



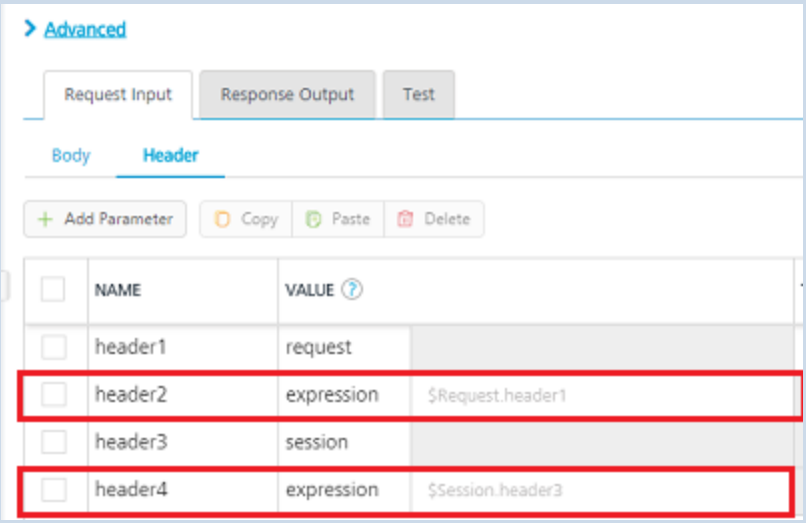
- To forward the header of the client's request to backend as it is, select the **Enable pass-through input header** check box. For more details on API Proxy service, refer to How to Enable Pass-through Proxy for Operations.
- To configure parameters in the client's header, do the following.
    i. In the **NAME** field, enter the name for the request input parameter.
    ii. **VALUE:** select request or session. 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.



iii. **TEST VALUE**: Enter a value. A test value is used for testing the service.

iv. **DEFAULT VALUE:** enter the value if required. The default value will be used if the test value is empty.

v. **DESCRIPTION**: Enter a proper description.

To validate the details, click **Fetch Response**. The result of the operation appears.

19. In the **Response Output** tab, provide the following details.

Output parameters are the parameters that are fetched from the response of a service call and return to the calling device. You can configure as many or few output parameters as you need. Trimming the dataset saves bandwidth. You can define output parameters as collections or single values. Collections are useful when you have repeating rows of data like the images that appear in your application

- Click **Add**. A row gets added to the table.
- Enter values for **ID**. It is advisable to create an ID that easily identifies the elements that is likely to capture from the response.
- Apply **XPath** for extracting the required elements from the response of the service call. XPath is used to navigate through elements and attributes in an XML document. For more information about *XPath*, see https://www.w3schools.com/xml/xml_xpath.asp.

  In this example, XPath is configured as:

  > ***Note:*** Kony Visualizer supports all functions of XPath 1.0, except the translate function.

ID	XPath
State	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult/State
City	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult/City

ID	XPath
Temperature	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /Temperature
Humidity	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /RelativeHumidity
Wind	//GetCityWeatherByZIPResponse/GetCityWeatherByZIPResult /Wind

- Select a value for **Scope** from the drop-down list. **Scope** denotes the scope of the parameter, if it is limited to *response* or *session*.
- Select a value for **Datatype** from the drop-down list. Datatype denotes the type of the parameter.
- **Collection ID** - Groups data elements under the specified parameter as a collection. Collection is created to assign grouped data to segment or a table in the UI.
- **Record ID** - Groups data elements under the specified parameter as a record. Typically this is used to provide metadata to the segment.
- **Format** - Can be set to None, currency, number or date.
- **Format Value** - Provide the conversion standard for the selected format.
- If a parameter row needs to be deleted, select the parameter row to be deleted and click **Delete**. This action removes the parameter row from the list of service parameters.

20. Click **Advanced**.

    - **Custom Code Invocation**: Upload a JAR file containing the pre-processor class name and post-processor class name. This step allows you to further filter the data received from a service call.

      > ***Note:*** For more information on Pre-processor and Post-processor, see the section Preprocessor/Post Processor.

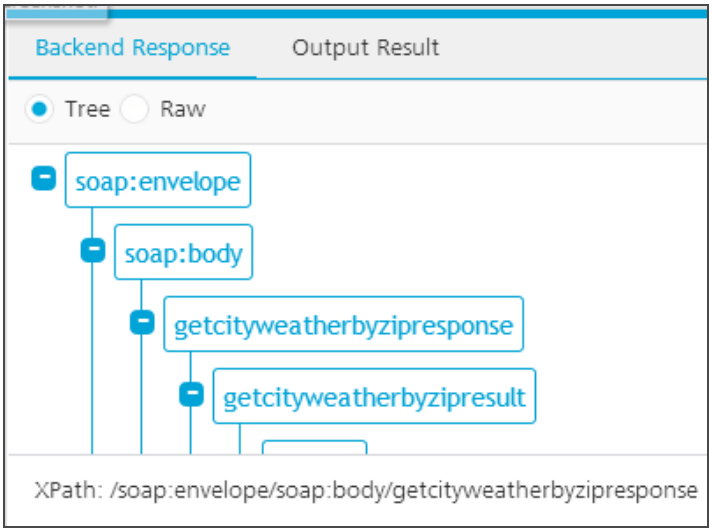
    - **HTTP Headers**: You can provide the HTTP Headers for the call.
    - **Properties**: You can configure various advanced service properties.
    - **Timeout** (in ms) - Specify the maximum time (in milli-seconds) the service waits for a response, before terminating the connection to the external data source.
    - **Cacheable** - Specify if the service response must be stored in the cache.
    - **Cacheable Duration** - This option is enabled if you select the **Cacheable** property. This property specifies the time duration in seconds.

      > ***Note:*** If this service is **Cacheable** and a call is made before the specified time duration, the service response is fetched from the Cache, else a fresh service call is made to fetch response from the server.

    - Click **Save Operation** to the save the operation.

21. Click **Test** to view the result of the operation. The results are displayed in Output Result area.

    - You can view the back-end response in Backend Response tab.
    - In Backend Response, you can view the raw response or view the response in a tree format. Clicking on a element in the tree displays the Xpath of that tag.



22. Click **Test**. The Result of the operation appears.
23. Click **Save Operation** to the save the operation.
24. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Create a JSON Service

Applies to *Kony Visualizer Classic*.

A JSON service communicates with an external data source using JSON connector over the HTTP protocol and returns a response in JSON format. JSON is popular over XML and returns a response in JSON format.

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.

In this topic, you will learn about:

JSON Connector Basics

Create a JSON Service

Publish the Service

Map the Service Output to Widgets on a Form.

JSON Connector Basics

Notations

- JSON Object - {}
- JSON Array - []

Important Considerations

- JSON Array consists 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 does not contain attributes.
- JSON path does not provide Axes like Xpath.

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.

Sample Code

```
{
"bookstore": {
"book": [
{ "category": "reference",
"author": "Nigel Rees",
"title": "Sayings of the Century",
"price": 8.95
},
{ "category": "fiction",
"author": "Evelyn Waugh",
"title": "Sword of Honour",
"price": 12.99
},
{ "category": "fiction",
"author": "Herman Melville",
"title": "Moby Dick",
"isbn": "0-553-21311-3",
"price": 8.99
},
{ "category": "fiction",
"author": "J. R. R. Tolkien",
```

```
"title": "The Lord of the Rings",
"isbn": "0-395-19395-8",
"price": 22.99
}
]
}
}
```

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[1]	Selects the first book element that is the child of the bookstore element

Path Expression	Result
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
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

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

Operator	Description	Example	Result
<	Less than	price<9.80	• True if price is 9.00 • False if price is 9.80
!=	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

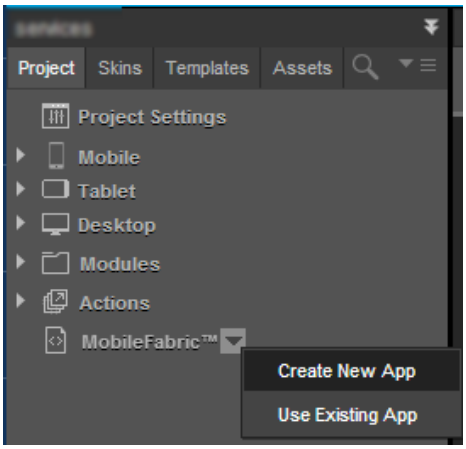
Creating a JSON Service

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

To create a JSON service, do the following:

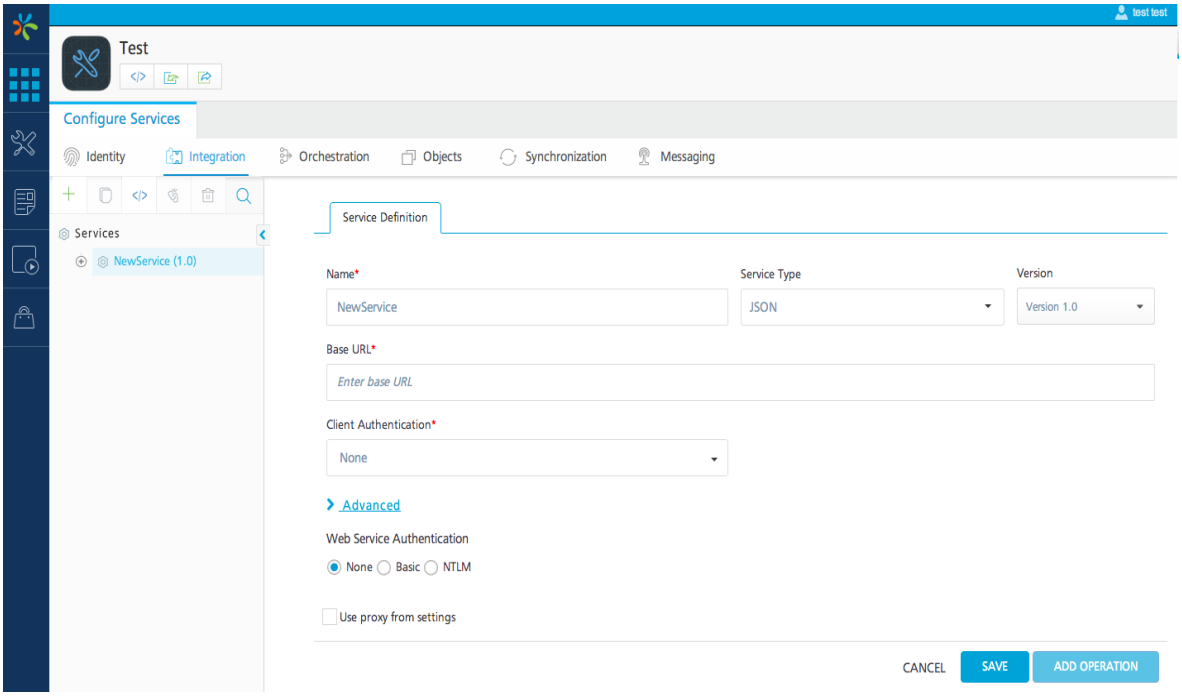
1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create**

New App, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.

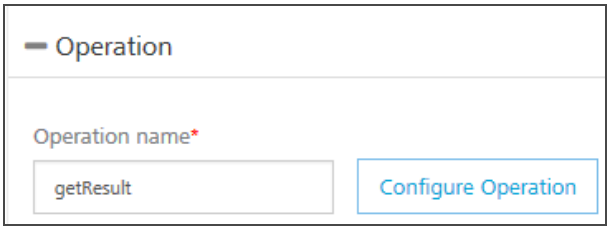


> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.

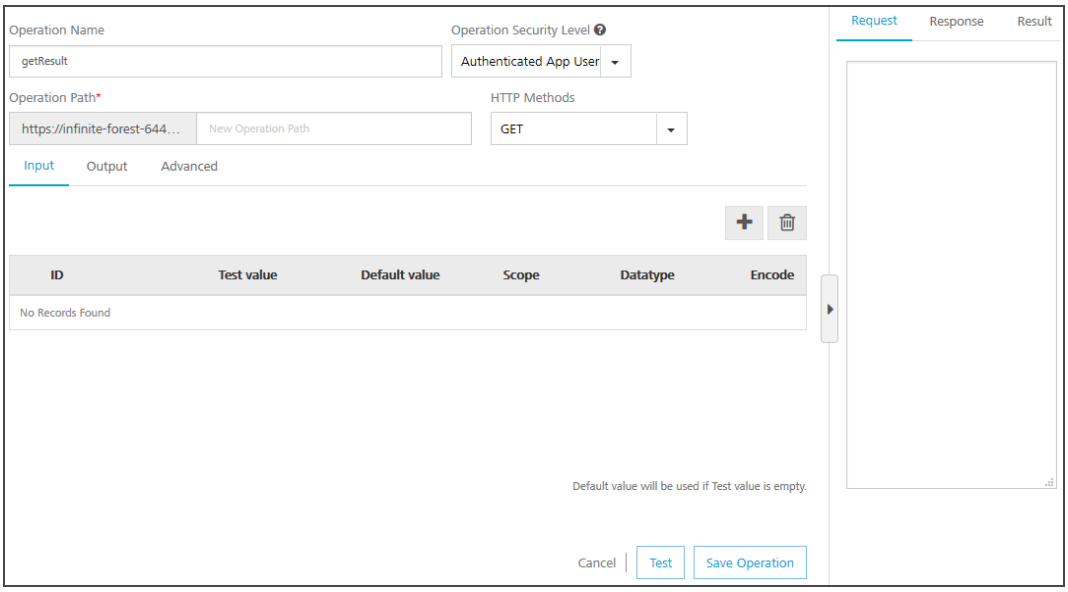


5. In the **Base URL**, type the URL.
6. Under the **Web Service Authentication**, select one of the following modes:
    - **None**: Select this option if you do not want to provide any authentication for the service.
    - **Basic**: Provide User ID and Password if the external Web service requires 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.
    - Click **Save and Continue**.
    - On the **Operation** tab, type a new name for the operation in the **Operation Name** box.



7. Click **CONFIGURE OPERATION**. The New Operation tab appears.

> ***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.



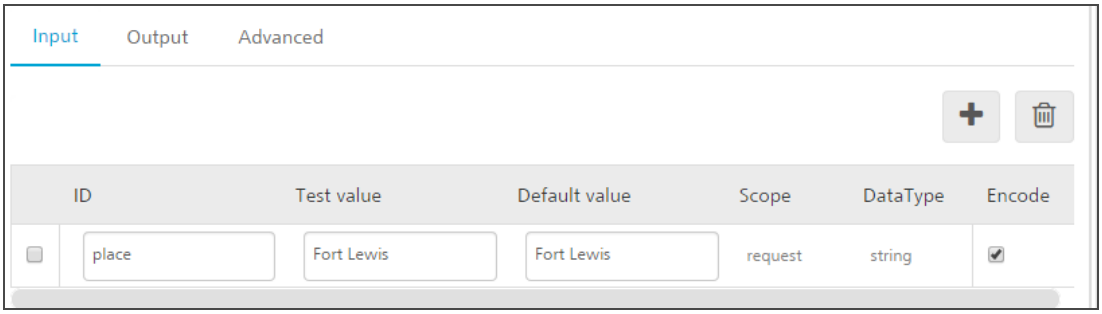
8. In the **Operation Name** text box, modify the name if required.
9. Select one of the following security operations in the **Operation Security Level** field. By default, this field is set to **Authenticated App User.**

   You can restrict access to this operation based on the following levels:

   - **Authenticated App User** - indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.
   - **Anonymous App User** - indicates that a user must have the app key and app secret to access this operation.
   - **Public** - indicates that this operation requires no special security.
10. In the **Operation Path** text box, modify the path if required.

11. Select the required method for this operation from the **HTTP Methods** field.

12. Click the **Request Input** tab to provide the required details.



13. Click **Response Output** and provide the required details.

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

14. Click **Advanced** to configure the following:

    Custom Code Invocation: Upload the JAR file containing the preprocessor class name and postprocessor class name. This step allows you to further filter the data received from a service call.

    HTTP Headers: You can provide the HTTP Headers for the call.

    Properties: You can configure various advanced service properties.

15. Click **Test** to view the result of the operation.

16. Click **Save Operation** to the save the operation.

17. Click **Done** to save the service.

18. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Create a Java Service

Applies to *Kony Visualizer Classic*.

A service that uses a custom Java connector is called a Java service. The Java connector is a custom Java Class that implements the **com.konylabs.middleware.common.JavaService** interface or **com.konylabs.middleware.common.JavaService2** interface. It is recommended that you 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. JAR files contain Java classes. The Java classes contain Java methods. These methods have the logic defined that is required for a service. Java services are mostly used in conjunction 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.

Prerequisites

The prerequisites for creating a Java Service are:

Conversion of JSON Data to a Kony Object

How to write a Java Class for a Java Adapter

In this topic, you will learn about:

Creating a Java Service

Publishing the Service

Mapping the Service Output to Widgets on a Form.

Creating a Java Service

With Java service, you can interact with your software application that does not support RESTful APIs. A service that uses a custom Java connector is a Java service. The Java connector is a custom Java class that implements the `com.konylabs.middleware.common.JavaService` interface.

Adding a Java service involves the following steps:

Configure a New Java Connector - Integration Service

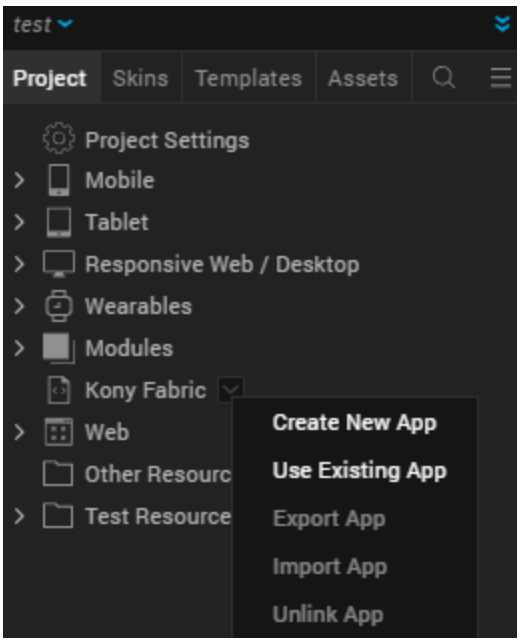
Edit an Existing Java Connector - Integration Service

Configure a New Java Connector

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

To configure your Java Adapter, follow these steps:

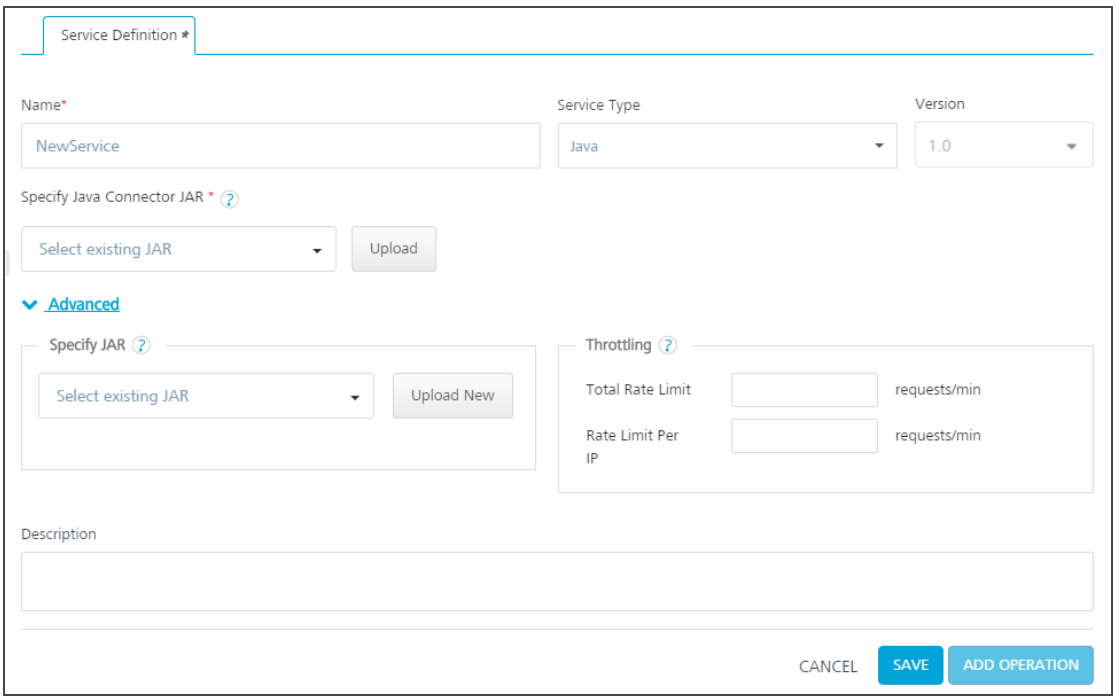
1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the Kony Fabric application that you want to publish. The Kony Fabric Console opens.



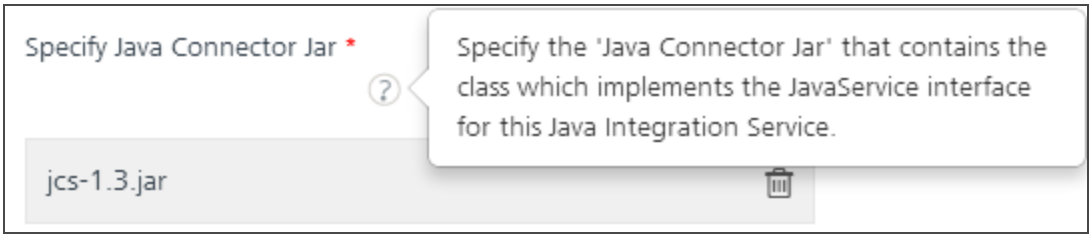
> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabricapp, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

The Kony Fabric Console opens. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.

4. Click the **Integration** tab, click **CONFIGURE NEW** to create an integration service.
5. In the **Name** box, type a unique name for the service.
6. From the **Service Type** list, select **Java**.
   By default, XML is selected. If you select **Java**, the **Specify Java Connector Jar** section appears.



7. From the **Specify Java Connector Jar** list, select a JAR file, or click **Upload New** to select the JARs from your local machine.
    a. The system adds your main JAR file to the console. The system displays the added JAR file's name under the **Specify Java Connector Jar** field.

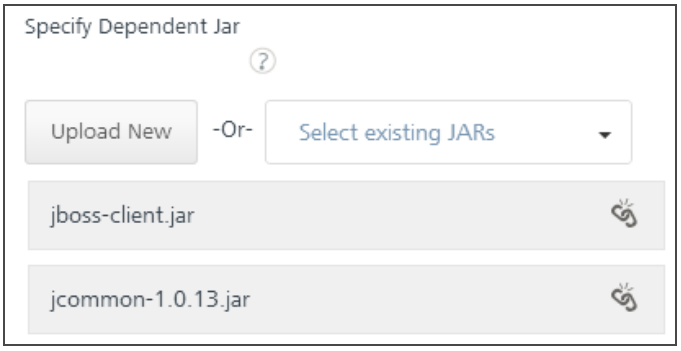


You can select and upload only one JAR file at **Specify Java Connector Jar**. The main JAR file contains Kony Integration Java services.

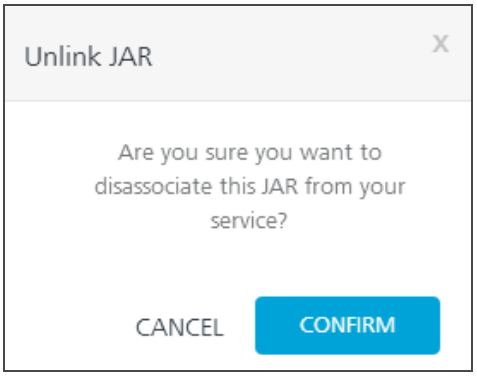
You can delete an uploaded JAR file by clicking the **Delete** button.

> ***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.

8. Under the **Specify Dependent Jar** list, select one or more JAR files if the main JAR depends only on external JARs. You can also click **Upload New** to select the JARs from your local machine. The system adds JAR files to the console.



To remove an existing dependent JAR file from the console, click the **Unlink** button. An **Unlink JAR** confirmation message window appears. Click **CONFIRM**. The JAR file is removed from the **Specify Dependent Jar** section.



> ***Important:*** To update a JAR file, you must upload the JAR file with name matching your previous JAR file. The existing JAR file gets replaced by the new JAR file.

9. Click the **Advanced** tab to specify dependent JAR and API throttling. All configurable

parameters in the Advanced section are optional.

- **To specify dependent JAR, follow these steps**:

  Select the JAR containing preprocessor or post processor libraries from the drop-down list, or click **Upload New** to browse the JAR file from your local system. The 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.

- **API throttling** enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, the API will not return the service response.
  To specify throttling, 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, refer to Override API Throttling Configuration.

10. Click **SAVE** to save your service definition. The system displays the success message: Service Saved Successfully.
    The **Operations List** tab appears only after the **Service Definition** is saved. The **ADD OPERATION** button in the Service Definition page is active only after you click the **SAVE** button.

11. Click **ADD OPERATION**. The Operations List tab appears.
    Based on your JAR files, the system loads all Java classes into the console.

12. In the **Operation List** section, follow these steps to configure operations for your Java service:

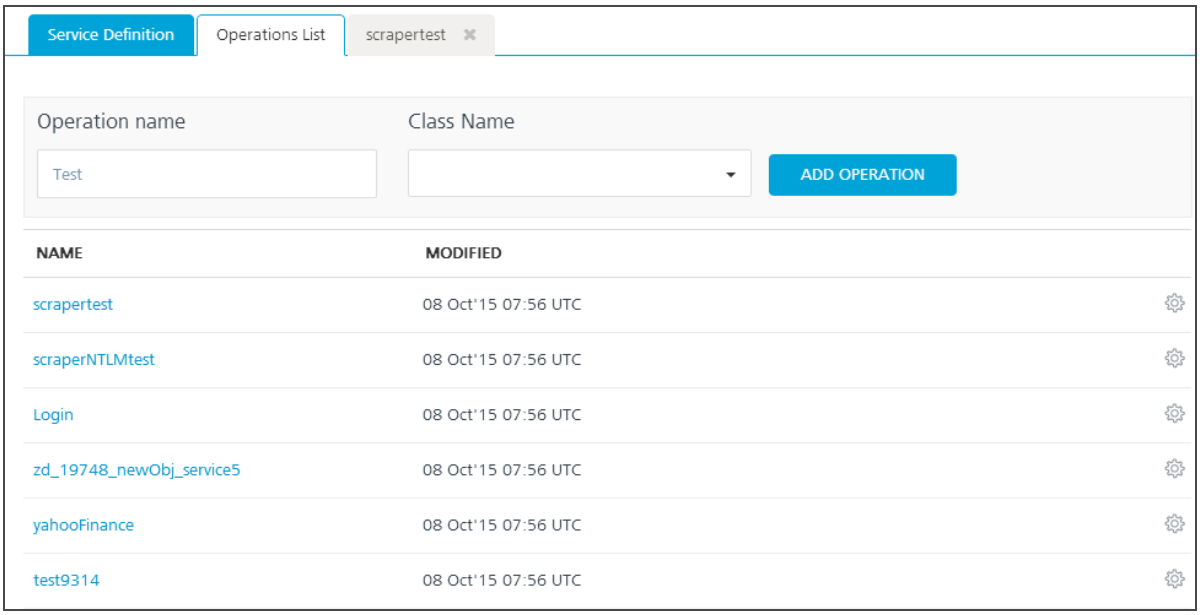
    In the **Operation** name text box, enter a name. For example, Java_operation.

    From the **Class Name** drop-down list, select the Java class that has the method defined.

    Click **ADD OPERATION** to display the Operation Modal tab.

13. In the **Operation Modal** tab, follow these steps:

    The following dialog contains the request input, response output, and advanced tabs. 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.

a. In the **Operation Name** box, modify the name if required.

b. Select a class name from the list.

c. Click **Add Operation**. The operation gets added and appears in the Operations List tab.

d. Click the operation name.

e. Select one of the following security operations in the **Operation Security Level** field. By default, this field is set to **Authenticated App User.**

   You can restrict access to this operation based on the following levels:

   Authenticated App User - indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.

   Anonymous App User - indicates that a user must have the app key and app secret to access this operation.

   Public - indicates that this operation require no special security.

14. In the **Request Input** > **Body** tab, to configure parameters in the clients body, provide the following details:

    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.

Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

> ***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.

i. The **NAME** field contains a unique identifier for a parameter. Change the identifier if required.

ii. In the **VALUE** field, select request or session. By default, this field is set to **Request.** Three different options are available in Kony Fabric under **Request Input > Body** > **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.

- **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.

iii. **TEST VALUE**: Enter a value. A test value is used for testing the service.

iv. **DEFAULT VALUE:** enter the value if required. The default value will be used if the test value is empty.

v. 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.
- **Boolean** - A value that is true or false.
- **Number** - An integer or a floating number.
- **Collection** - A group of data or data set.

vi. Select the **Encode** check box 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 to the HTML URL encoding standards.

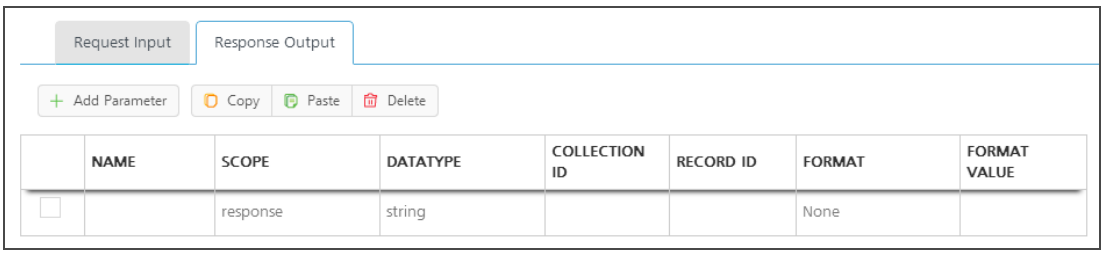
vii. In the **Description**, provide the description.

15. 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.



i. The **NAME** field contains a unique identifier for a parameter. Change the identifier if required.

ii. Select request or session in the **SCOPE** field. By default, this field is set to **Request**.

Request - Indicates that the value must be retrieved from the HTTP request received from a mobile device.

Session - Indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.

iii. 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.

Boolean - A value that is true or false.

Number - An integer or a floating number.

Collection - A group of data, also referred to as data set. A collection contains only records, and a record contains 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.

iv. Select a format type in the **FORMAT** list.

- **None** - No format.
- **Currency**-Currency format.

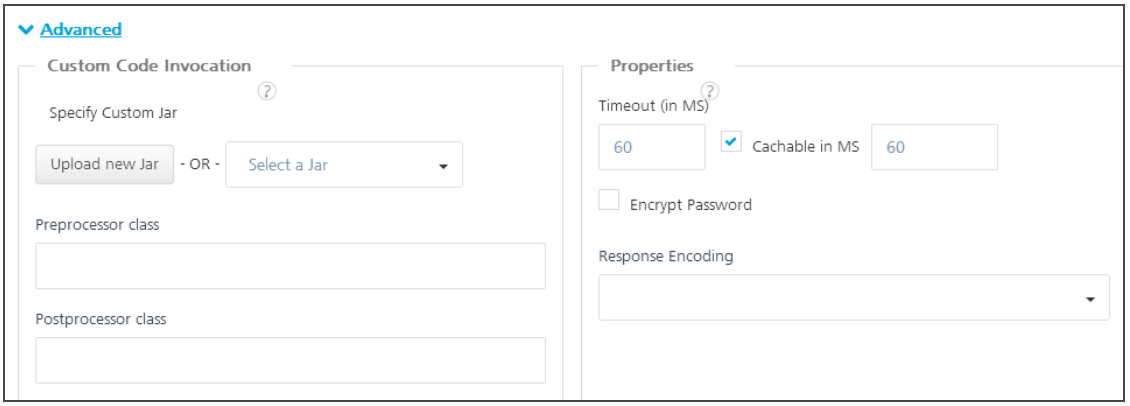
- **Number** - Number format.
- **Date**- Date format.
  - If datatype is String, then the options in the Format Type are Currency, Number and Date.
  - If the datatype is Number, then the options in the Format Type are Currency and Date.
  - If the datatype is Boolean, then the options in the Format Type and Format Value text box are disabled.

v. In the **FORMAT VALUE** field, provide the standard for converting the specified format. For example, enter a date format as **MMDDYY** to display the date in Month/Date/Year.

- **Currency** - For currency format, refer to `java.text.DecimalFormat`.
- **Number** - For number format, refer to `java.text.DecimalFormat`.
- **Date** - For date format, refer to `java.text.SimpleDateFormat`.

16. Click the **Advanced** tab to configure the following options for request or response operations.

> ***Note:*** All options in the **Advanced** section for operations are optional.



- Custom Code Invocation - Preprocessor and Postprocessor
  You can add preprocessing 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.
  Upload the JAR file containing the preprocessor class name and post processor class name. This step allows you to further filter the data received from a service call.

- Additional configuration properties allow you to configure service call time out cache response:

  - **Specify Custom Jar** - Browse and select the JAR containing preprocessor or post processor libraries.

  - **Preprocessor class** - Enables a developer to include any business logic on the data before forwarding the request to the external data source. Select the JAR file from the list.

  - **Postprocessor class** - Enables a developer to include any business logic on the data before sending the response to a mobile device. Select the JAR file from the list.

  - 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.

    Cachable(in sec) - the duration in seconds within which the service response is fetched from the cache. Select the **Cachable(in sec)** check box and provide the details in the text box.

- Front End API allows you map your endpoint or back-end URL of an operation to a front-end URL.

17. Click **Save Operation** to save the operation. The system adds your operation under the **Configured Operations** section. The system also adds your new Java service to the **Integration**

page.

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 steps for Configure New.

18. From the **Operation** section, click **Done** to complete the configuration and go to the **Integration** page.
19. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon  . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

How to Edit an Existing Java Connector Integration Service

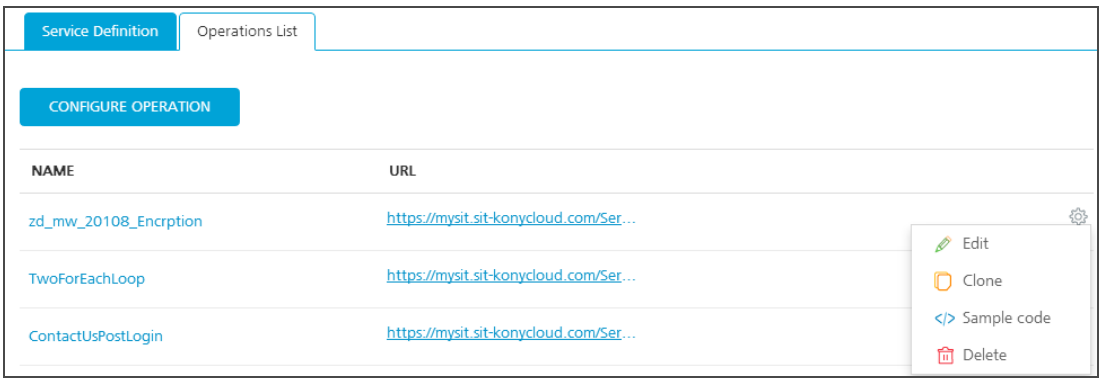
If you want to edit an existing Java service, you can edit details such as service name, JAR files, operation modal details. While editing a Java service, you can change the Java service type. A Java service must be available in the Integration home screen.

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

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** dialog.
   To modify Java operations, refer to How to Configure and Edit Operation Modal.

5. Click **Done** to update your Java service. The Integration page appears.

6. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Create a Database Service

Applies to *Kony Visualizer Classic*.

With Kony Fabric database connector, you can connect to your own database as an endpoint. After you configure the database connector in Kony FabricConsole, you can perform create, read, update, and delete (CRUD) operations on data in the tables.

For example, banks maintain a store of users and their details. With Kony Fabric database connector, banks can connect to their own databases and manage customers data.

In this topic, you will learn about:

Advantages of Kony Fabric Database Connector

Limitations of Kony Fabric Database Connector

Publishing the Service

Mapping the Service Output to Widgets on a Form.

Advantages of Kony Fabric Database Connector

- Admins can connect to the given database.
- Admins can mange the databases using CRUD operations.
- When an admin is creating CRUD operations, the admin can access the configured schema.

Limitations of Kony Fabric Database Connector

- Currently Kony Fabric database connector supports MySQL database, Oracle, MS SQL, and PostgreSQL databases.
- Plain SQL commands are 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.
- Using the update operation, an admin can update multiple records at a time based on primary value.
- Using the delete operation, an admin can delete one record at a time based on primary value.
- Support for a single operation on multiple tables is not available.
  For example, the system does not allow the `<Update>` operation for `Table1` and `Table2`.
- For on-premises, the primary key should be auto-incrementing in MySQL database. Otherwise, the system throws an error for the create operation.

- Data types: All major data types are supported. In Date data type, the YEAR() data type is not supported.
- Binary Support is available in RDBMS DataAdapter with Range header.
- Kony supports six ODATA parameters for the read operation such as `$filter`, `$orderby`, `$top`, `$skip`, `$select`, and `$expand`.
- Kony supports `eq`, `ne`, `lt`, `le`, `gt`, `ge`, and `or` operators with the `$filter` ODATA parameter.
    - For `$expand`, MySQL version must be 5.7.7+.
    - The `$expand` fails, if any related table has binary (bolb (or) bit) columns where MySQL itself fails to convert the binary data into a valid JSON.
    - If the child payload is higher than 1024 bytes, MySQL truncates the excess data with its default properties on `group_concat()`. To resolve the issue, you must configure a higher value than the max payload size for `group_concat_max_len` property.
    - For SQL Server, Oracle, and PostgreSQL databases, `$expand` is limited to one level.

> ***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/

Create a Database Service

Adding a Database Service involves the following steps:

Configure a Service Definition for a Database Service

Create CRUD Operations for a Database Service

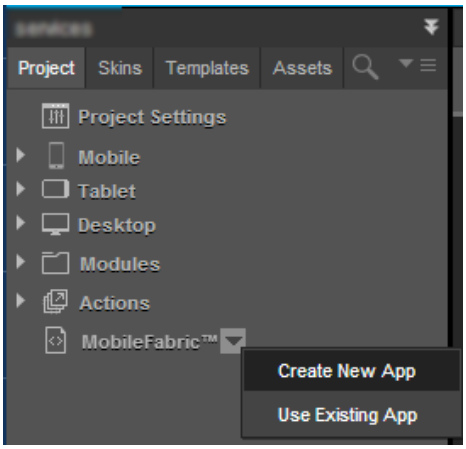
Configure CRUD Operations for a Database Service

Configure a Service Definition for a Database Service

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

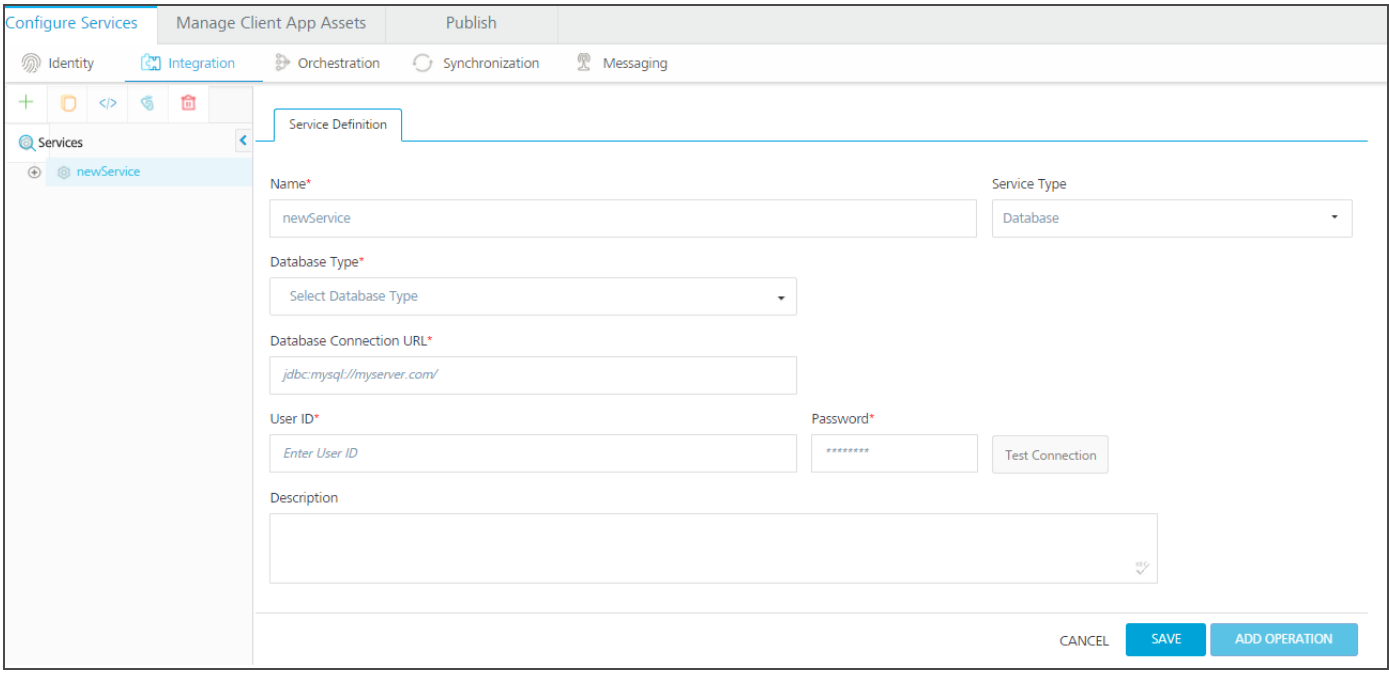
To configure a service definition, follow these steps:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.



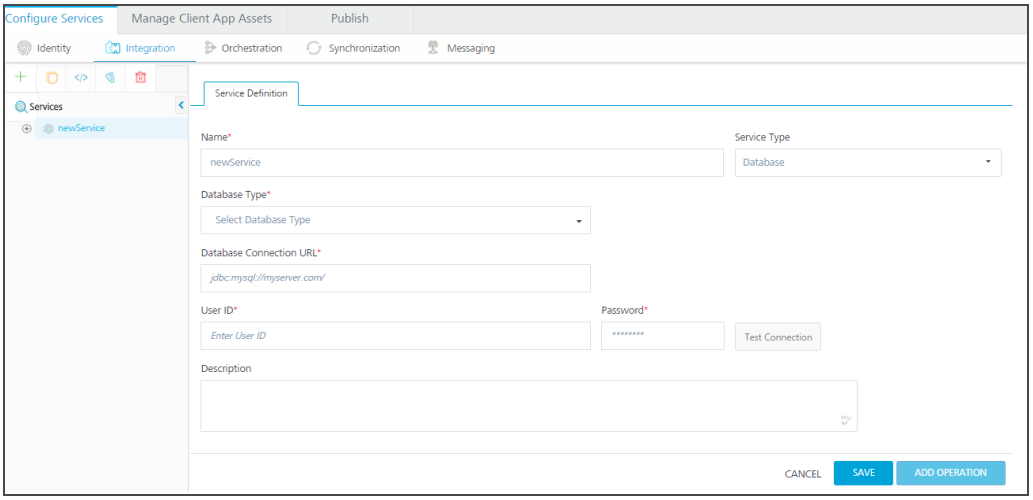
> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.



5. In the **Service Definition** section, follow these steps:

   - In the **Name** text box, enter 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.
   - From the **Service Type** list, select **Relational Database**.
     By default, XML is selected. If you select **Database**, the **Database Type, Database Connection URL**, and other details are displayed, shown below.



- In the **Database Type** list, select **MYSQL**.
- In the **Database Connection URL**, enter the database connection URL - for example, `jdbc:mysql://<ip_address>:<port>/`
- Under the **User ID**, type a valid user name for your database.
- Under the **Password**, type a valid password for your database.

  To test your database connection details, click **Test Connection**. If the entered details are correct, the system displays the message: Valid Database connection details.
- In the **Description**, type the appropriate description about the service.

> ***Note:*** For on-premises, proxy support is not available for database service.

- Click the **Advanced** tab to specify dependent JAR and API throttling. All options in the Advanced section are optional.
  - **To specify dependent JAR, follow these steps**:

    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. The 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.

- **API throttling** enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, the API will not return the service response.
  To specify throttling, 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, 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

> umber of configured nodes. For example, a throttling limit of 600
> equests/minute with three nodes will be calculated to be 200
> equests/minute per node.
> his is applicable for Cloud and On-premises.

- Click **SAVE** to save your service definition. The system displays the success message. The **Operations List** tab appears only after the **Service Definition** is saved. For creating operations for a database service, refer to How To Create CRUD Operations for Database Service.

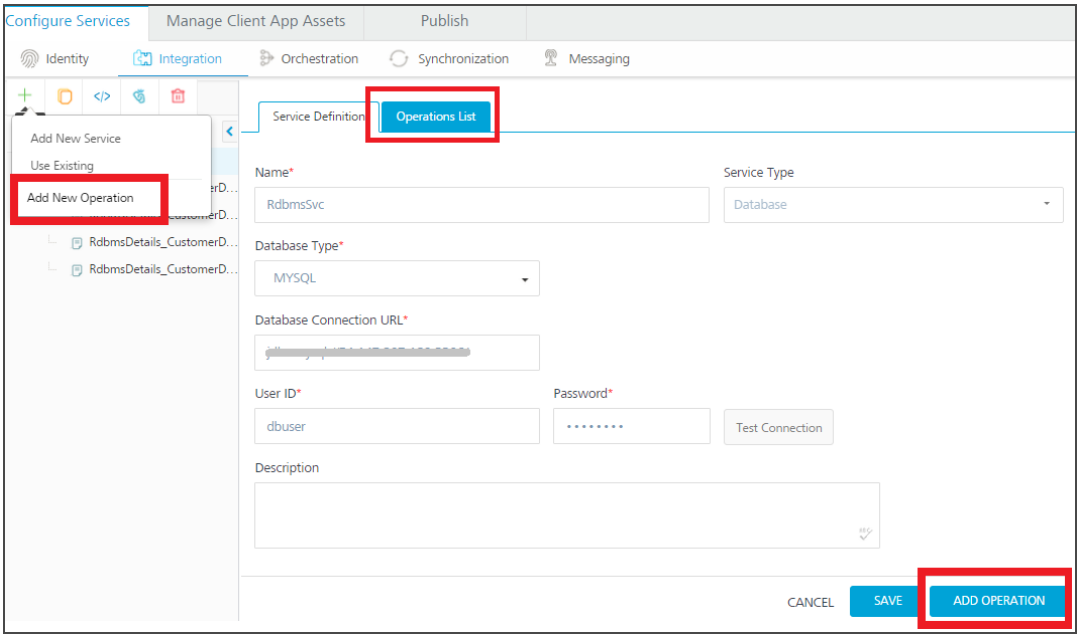
Create CRUD Operations for a Database Service

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

To create CRUD operations, follow these steps:

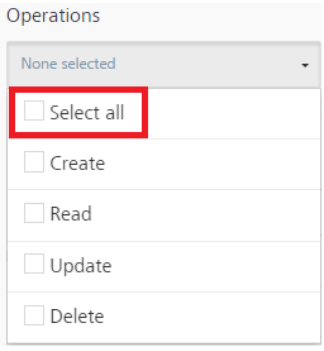
1. After you configure the service definition for a database service, click **ADD OPERATION** to display the **Operations List** tab.

> ***Note:*** You can also display the **Operations List** tab, by following steps:
> - Click the **Operations List** tab.
> - From the tree in the left pane, click **Add > Add New Operation**, shown below:

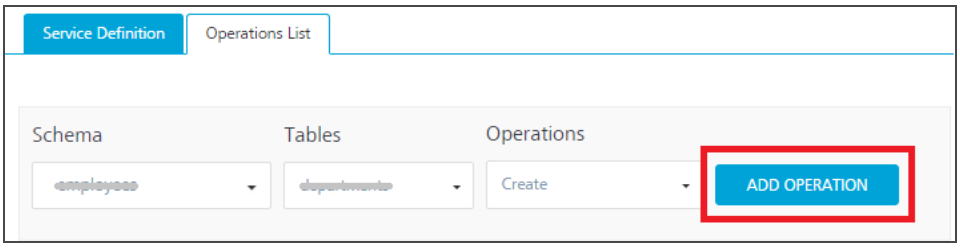


2. In the **Operations List** tab, follow these steps to configure operations:
3. From the **Schema** list, select the schema that is loaded based on your database configuration.

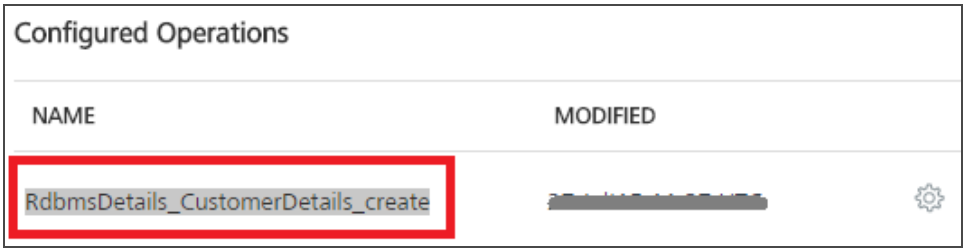
   Based on the selected schema, the **Tables** list is loaded with the tables in the schema. You can select one or more tables.
4. From the **Tables** list, select the required boxes. You can click **Select all** box to select all the tables in the list.
5. From the **Operations** list, select the required check boxes for CRUD operations. You can click **Select all** box to select four operations.



6. Click the **ADD OPERATION** button. The new operations are created under the **Configured Operations** section.



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 connector, 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 Service

After you create CRUD operations for database service, you can configure CRUD operations as follows:

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

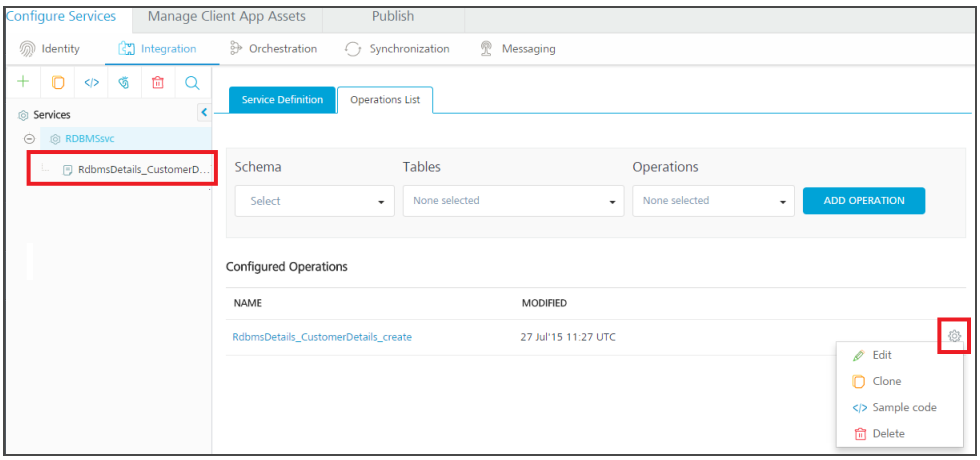
How to Create a Database Record with Create Operation

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

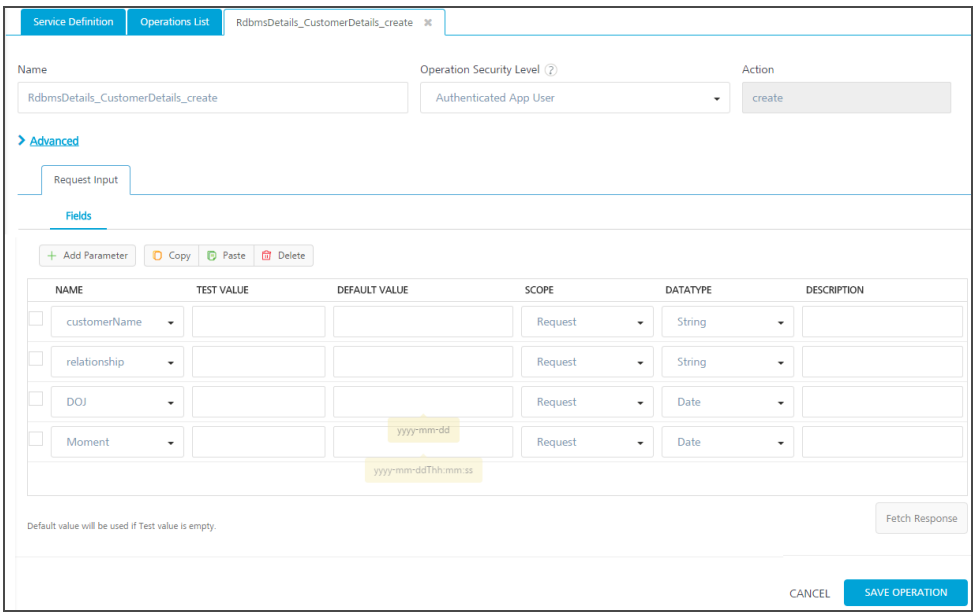
To create a database record with create operation, follow these steps:

1. Under **Configured Operations**, hover the cursor over the create operation, click **Settings** , and then click **Edit**.

   > ***Note:*** To edit an operation, you can also click the operation from the service tree pane.

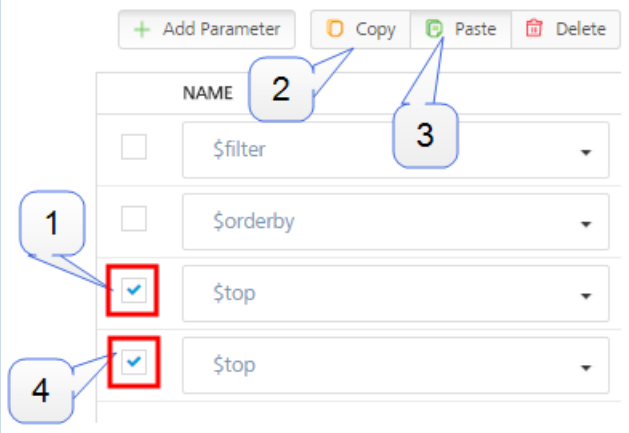
   

   The system displays the selected operation in the edit mode. The create operation has the Request Input tab.



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.

The **Name** field is pre-populated with fields names of the selected database. You can edit this field.

2. Select one of the following security operations in the **Operation Security Level** field. By default, this field is set to **Authenticated App User.**

   Authenticated App User – indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.

   Anonymous App User – indicates that a user must have the app key and app secret to access this operation.

   Public – indicates that this operation requires no special security.

   The **Action** field is pre-populated with operation names of the selected database. You cannot edit this field.

3. Click the **Advanced** tab, and follow these steps:

   i. Under the **Custom Code Invocation**, upload the JAR file containing the preprocessor class name and postprocessor class name. This step allows you to further filter the data received from a service call.

      Specify Custom Jar - Browse and select the JAR containing preprocessor or postprocessor libraries.

      Preprocessor class - Enables a developer to include any business logic on the data before forwarding the request to the external data source. Select the JAR file from the list.

      Postprocessor class - Enables a developer to include any business logic on the data before sending the response to a mobile device. Select the JAR file from the list.

   ii. Based on the operation - for example, post or get - provide custom HTTP headers. To provide customer headers, click **HTTP Headers** . In the **Test values** text box, provide custom HTTP headers required by the external data source, shown below:

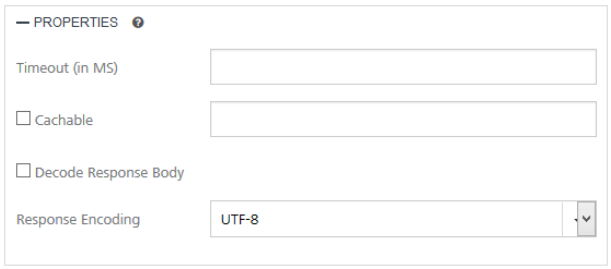
      ID: The rows are created based on the selected operation. Change the value if required.

      Test value: Enter a value. A test value is used for testing the service.

      Default value: change the syntax if required.

Scope: select request or session. By default, this field is set to **Request.**

iii. 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.

Cachable(sec) - the duration in seconds within which the service response is fetched from the cache. Select the **Cachable(in sec)** check box and provide the details in the text box.

Decode Response Body - To ignore the Database response received in the XMLvalue field, select the **Decode Response Body 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.

4. In the **Request Input** tab, do the following:
   a. In the **TEST VALUE** filed, enter the user input for the selected column.
   b. In the **DEFAULT VALUE**, enter the value if required. The default value will be used if the test value is empty.

c. Select request or session in the **Scope** field. By default, the Scope field is set to Request.

Request - Indicates that the value must be retrieved from the HTTP request received from a mobile device.

Session - Indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.

The default data type for the selected column is loaded under the **DATATYPE** field.

d. In the **DESCRIPTION**, provide the description.

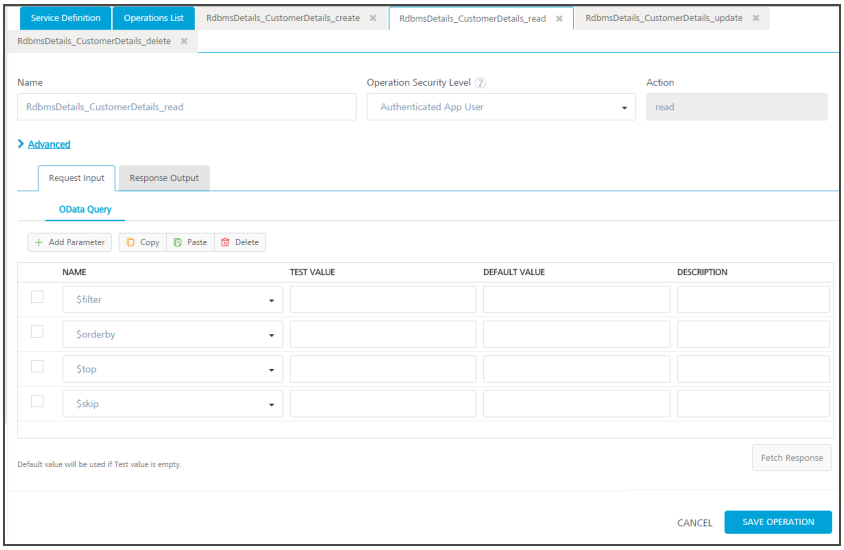
To validate the details, click **Fetch Response**. The result of the operation appears.

5. Click **SAVE OPERATION** to save the changes in the create operation.

Query a Database and Display Information with the Read Operation

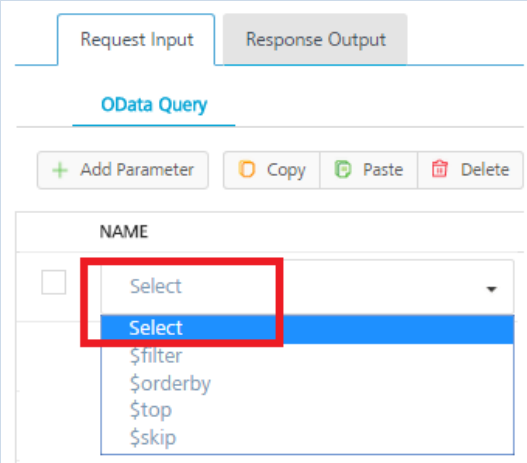
This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

1. Under **Configured Operations**, hover your cursor over the **Read** operation, click the **Settings** button, and then click **Edit**.The system displays the selected operation in the edit mode. The Read operation has the **Request Input** and **Response Output** tabs.

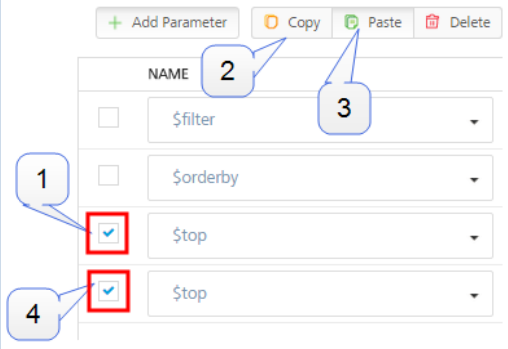


Note: You can add an entry by clicking the **Add Parameter** button if entries for the input and the output tabs do not exist.

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.

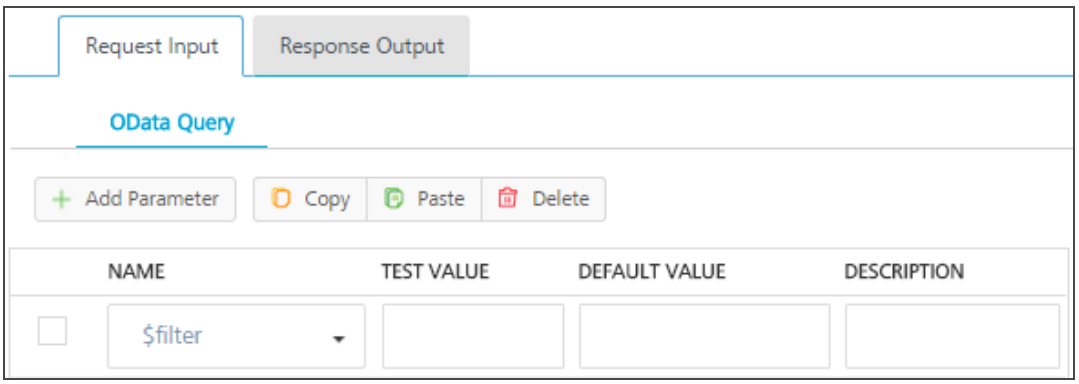


- To make duplicate entries, select the check box for the entry, click **Copy**, and then click **Paste**.



- 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:



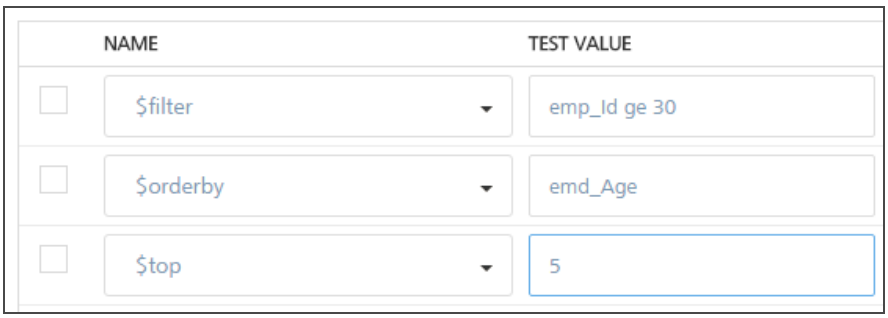
The **NAME** field in the **Request Input** is pre-populated with ODATA commands.

a. 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:



b. In the **DEFAULT VALUE**, enter the value if required.

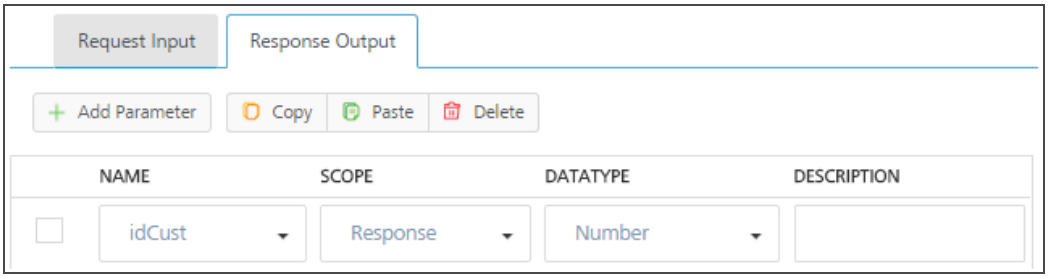
c. In the **DESCRIPTION**, provide the description.

To validate the details, click **Fetch Response**. The result of the operation appears.

Click **SAVE OPERATION** to save the changes in the read operation.

3. In the **Response Output** tab, configure the fields of the table for displaying the data:

The **Name** field in the Response Output tab is pre-populated with database columns.



a. Select request or session in the **SCOPE** field. By default, this field is set to **Request**.

Request - Indicates that the value must be retrieved from the HTTP request received from a mobile device.

Session - Indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.

b. In the **DESCRIPTION**, provide the description.

To validate the details, click **Test**. The result of the operation appears.

4. Click **SAVE OPERATION** to save the changes in the read operation.

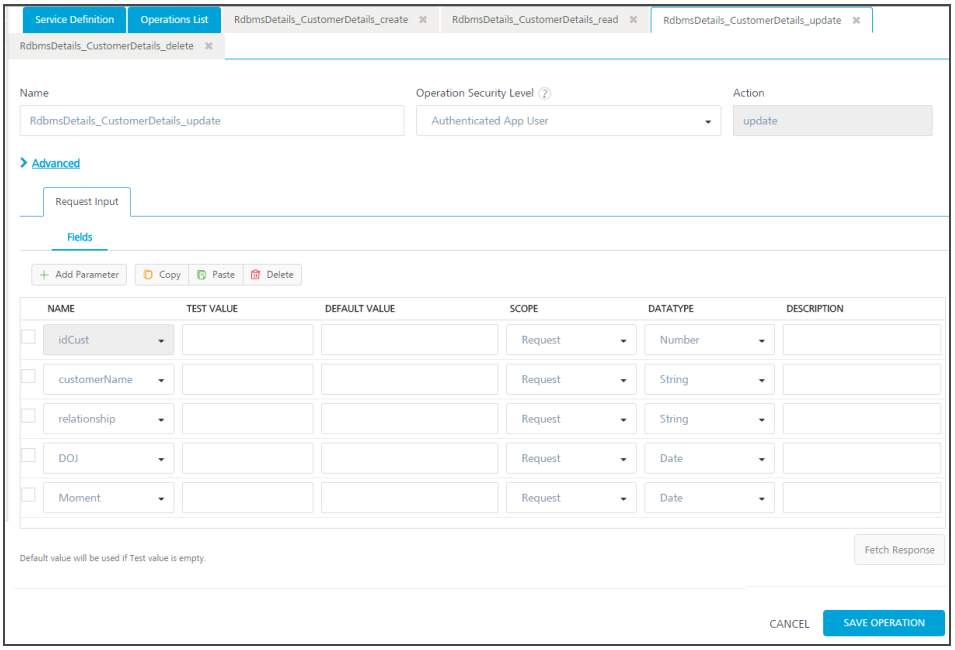
Update a Database Record with Update Operation

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

To update a database record with update operation, follow these steps:

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.

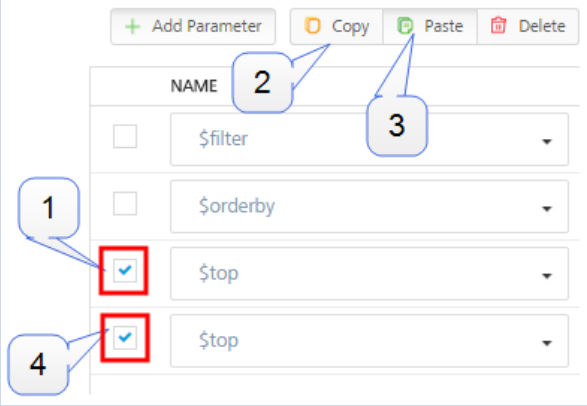


2. The **NAME** field contains primary key of the table. You cannot modify these details.

   The Name column is pre-populated 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.

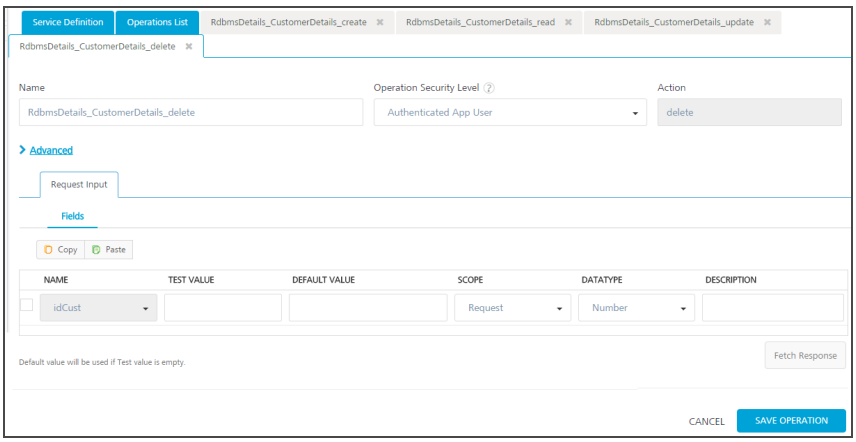
   To validate the details, click **Fetch Response**. The result of the operation appears.

4. Click **SAVE OPERATION** to save the changes in the update operation.

How to Delete a Database Record with Delete Operation

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

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.
6. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Configure an SAP Service

Applies to *Kony Visualizer Classic*.

With Kony Fabric, you can access external Kony SAP services by using the Kony SAP Gateway connector. Based on your Kony SAP Gateway authentication, you can use Kony SAP libraries and objects along with the supported HTTP methods in your app.

Adding a Kony SAP Gateway service involves the following steps:

Configure a New Kony SAP Gateway

Edit or Test an Existing Kony SAP Gateway Integration Service

Publishing the Service

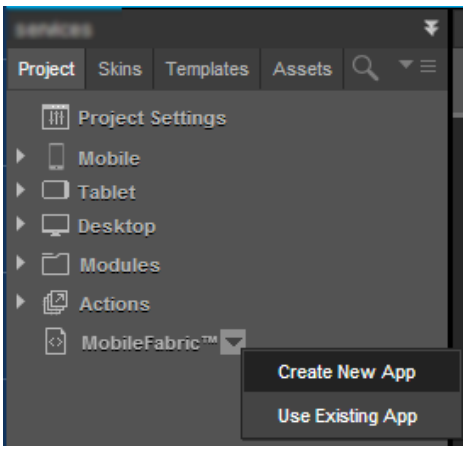
Mapping the Service Output to Widgets on a Form.

Configure a New Kony SAP Gateway

This procedure assumes that you have already configured Kony Fabric in Kony Visualizer. For more information, see Connect to the Kony Fabric Console.

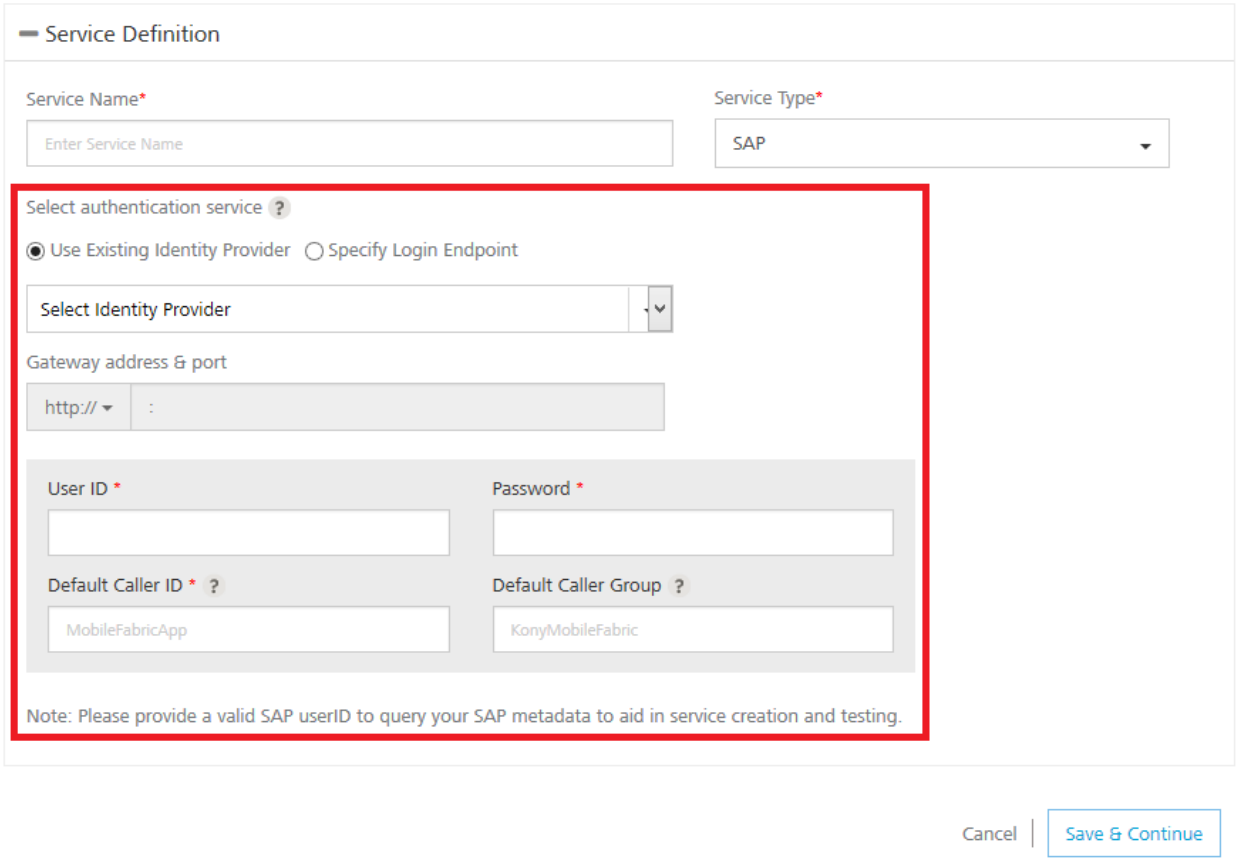
To configure your Kony SAP Gateway, follow these steps:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.

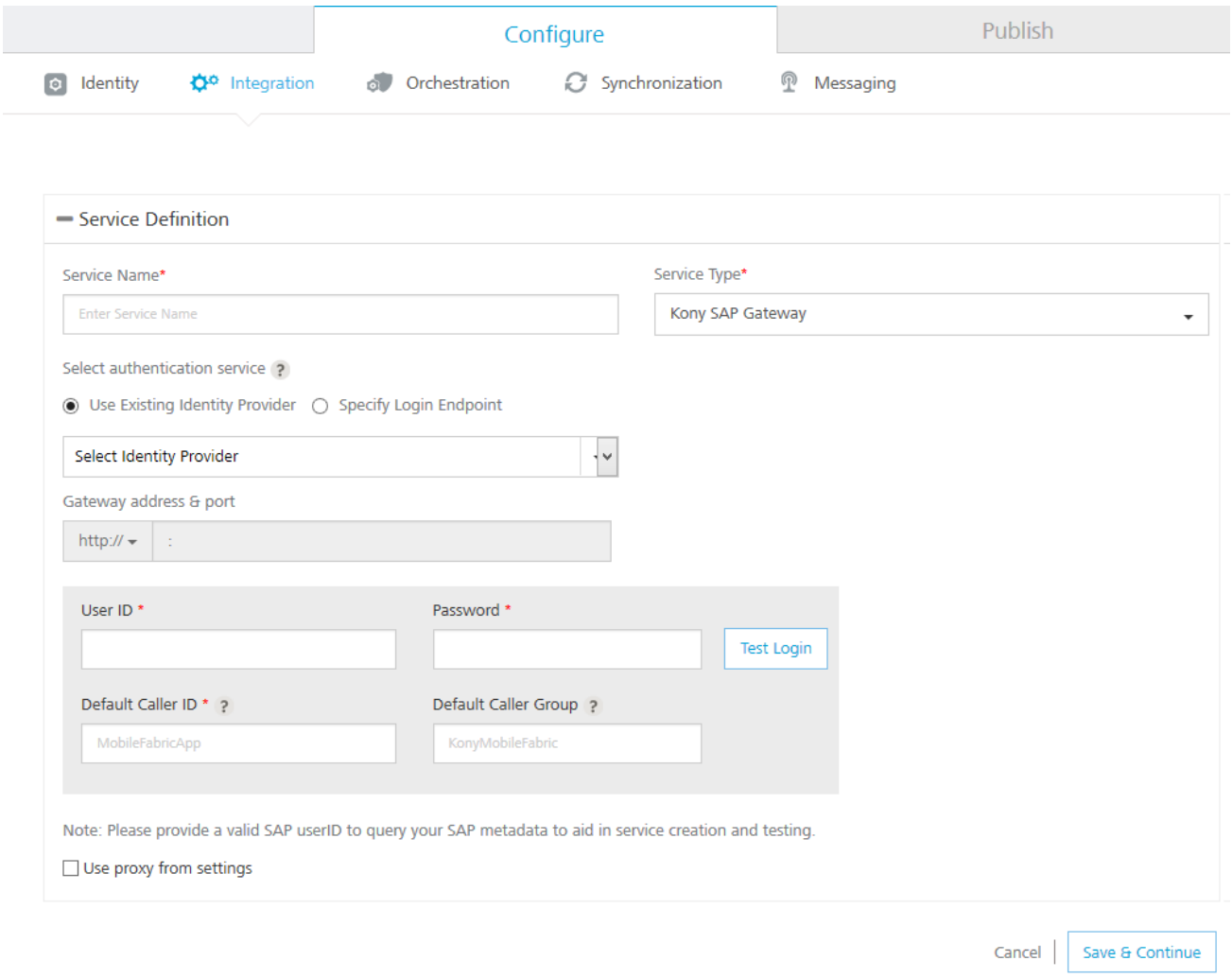


> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

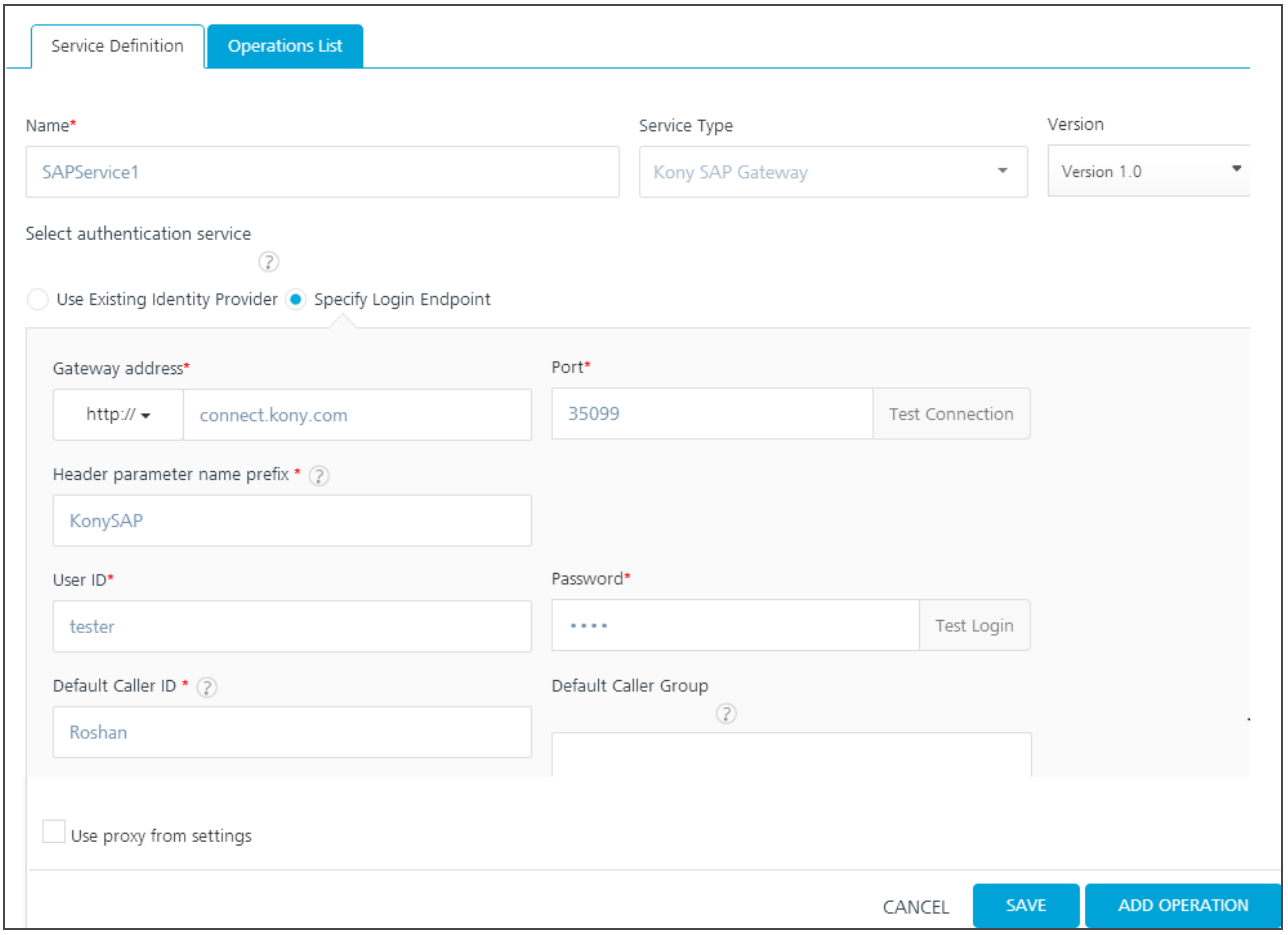
4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.



5. In the **Integration** tab, click **CONFIGURE NEW** to create an integration service.



6. In the **Service Definition** section, follow these steps:
    a. In the **Service Name** text box, enter a unique name for your service.
    b. From the **Service Type** list, select **Kony SAP gateway**.
    By default, XML is selected. If you select **Kony SAP gateway**, the **Select authentication service** section is displayed, shown below.



c. Under **Select authentication service**, click Specify Login Endpoint:

- Use Existing Identity Provider - to select an identity provider. This drop-down lists all identity providers if you have already created identity providers for SAP in the Identity page.
- Specify Login Endpoint- to configure a new endpoint.

To configure **Specify Log-in Endpoint**, fill in the details for the following fields:

i. In the **Gateway address**, enter the domain - for example, connect.kony.com.

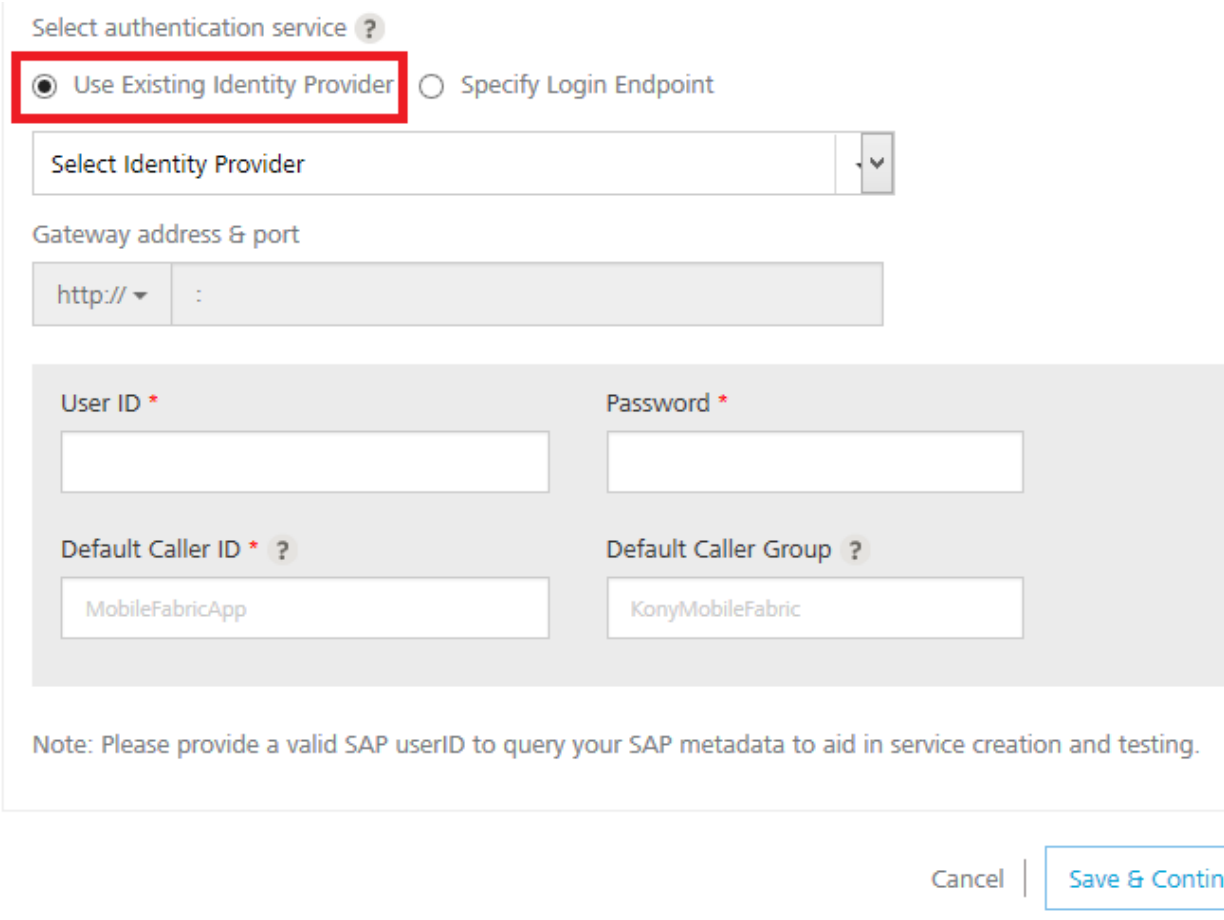
ii. In the **Port** text box, enter a valid port number ranging from 1 to 65535.

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 optional.



7. Click the **Advanced** tab to specify API throttling. All options in the Advanced section are optional.
    - API throttling enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, the API will not return the service response.
      To specify throttling, 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.

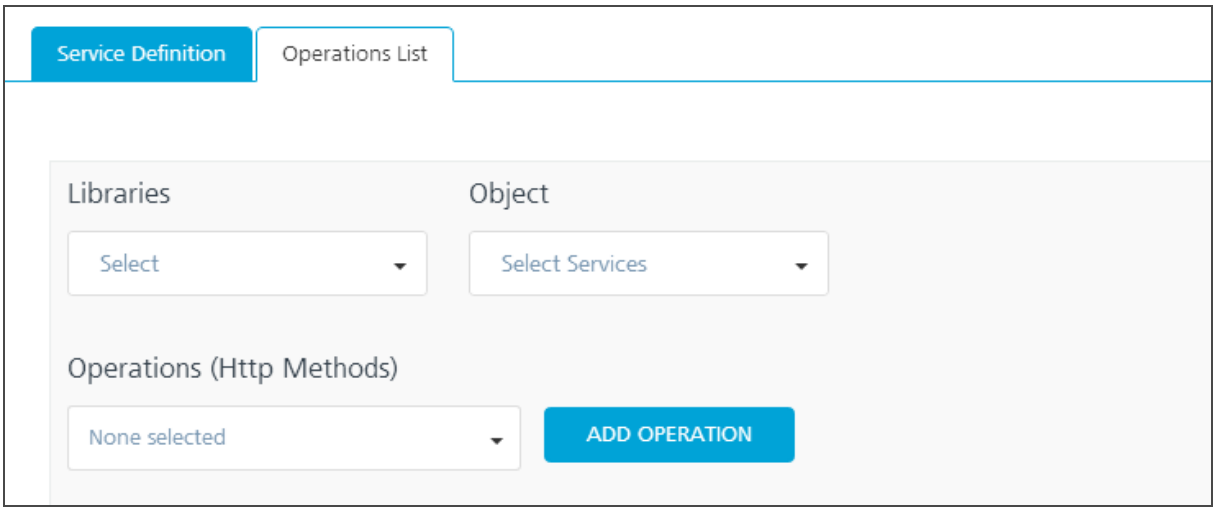
        To override throttling, refer to Override API Throttling Configuration.

        > ***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.

8. 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**.

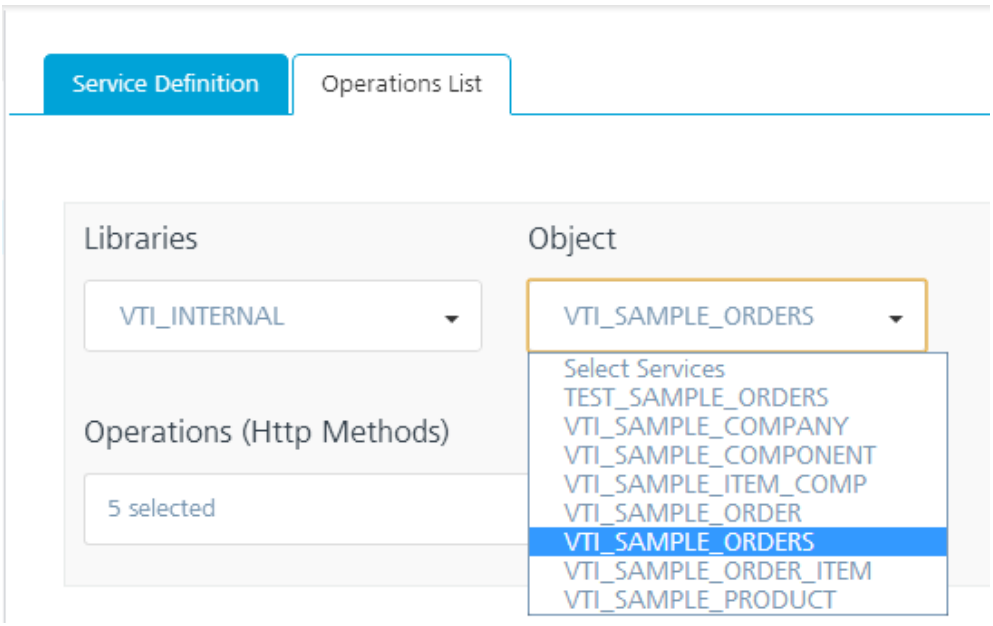
9. After you configure the authentication service, click **Save** .

10. Click **Operations List** tab. Based on your Kony SAP Gateway authentication, the system loads all tables such as libraries and objects along with supported HTTP methods

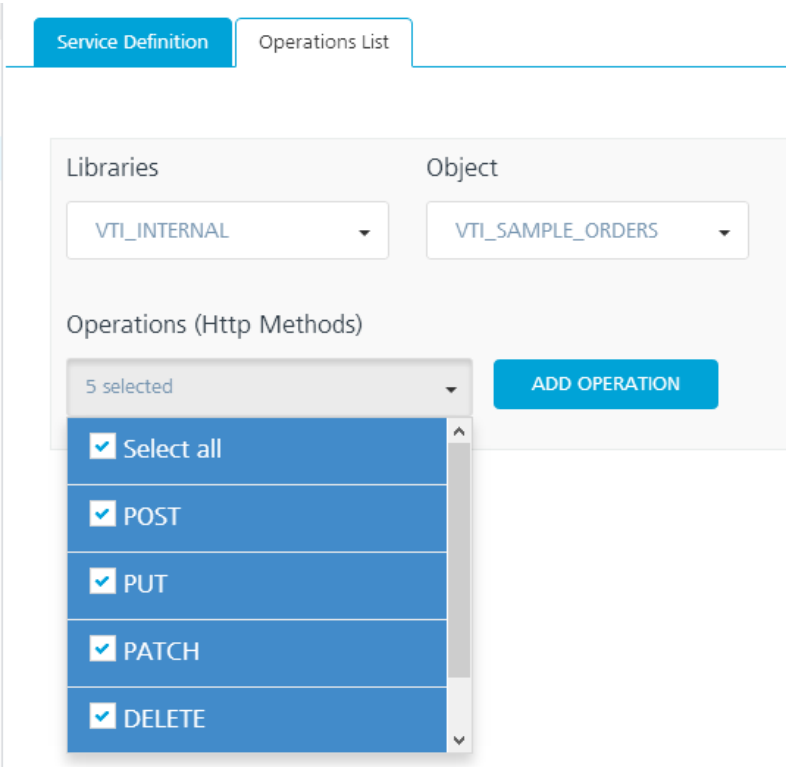


11. Select a library form the **Libraries** list. The objects of the selected library will be loaded in the Objects list.

12. Select an object from the **Objects** list.



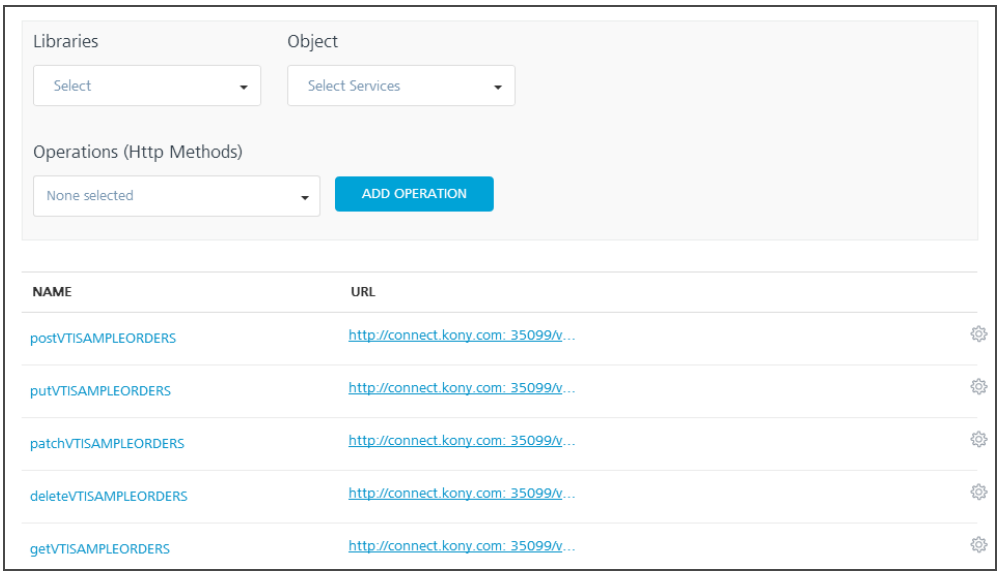
In the **Operation** list, select an operation or select all the operations.



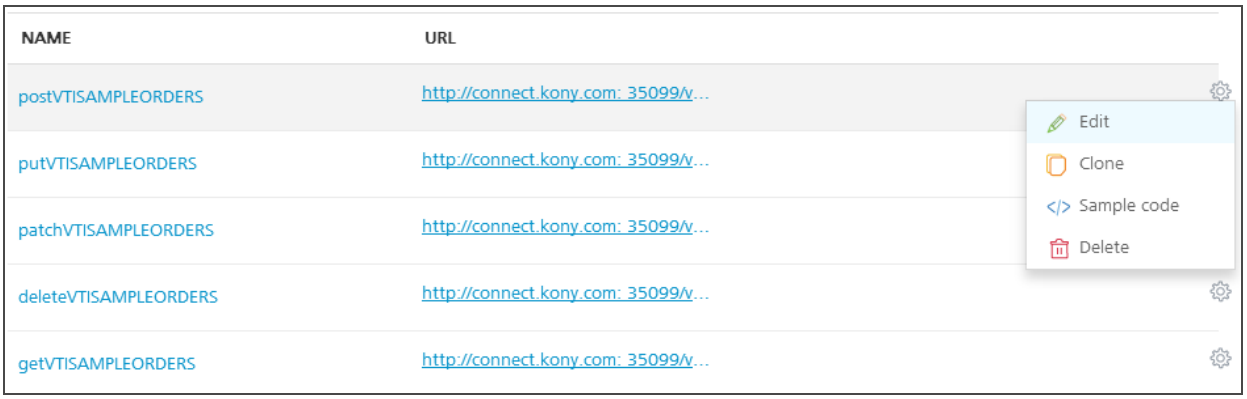
a. To configure more operations for your Kony SAP Gateway integration service, repeat steps a through b. You can select a new library and object, and supported operations.

b. Click **Add Operation**. The system adds your operation under the **Configured Operations**

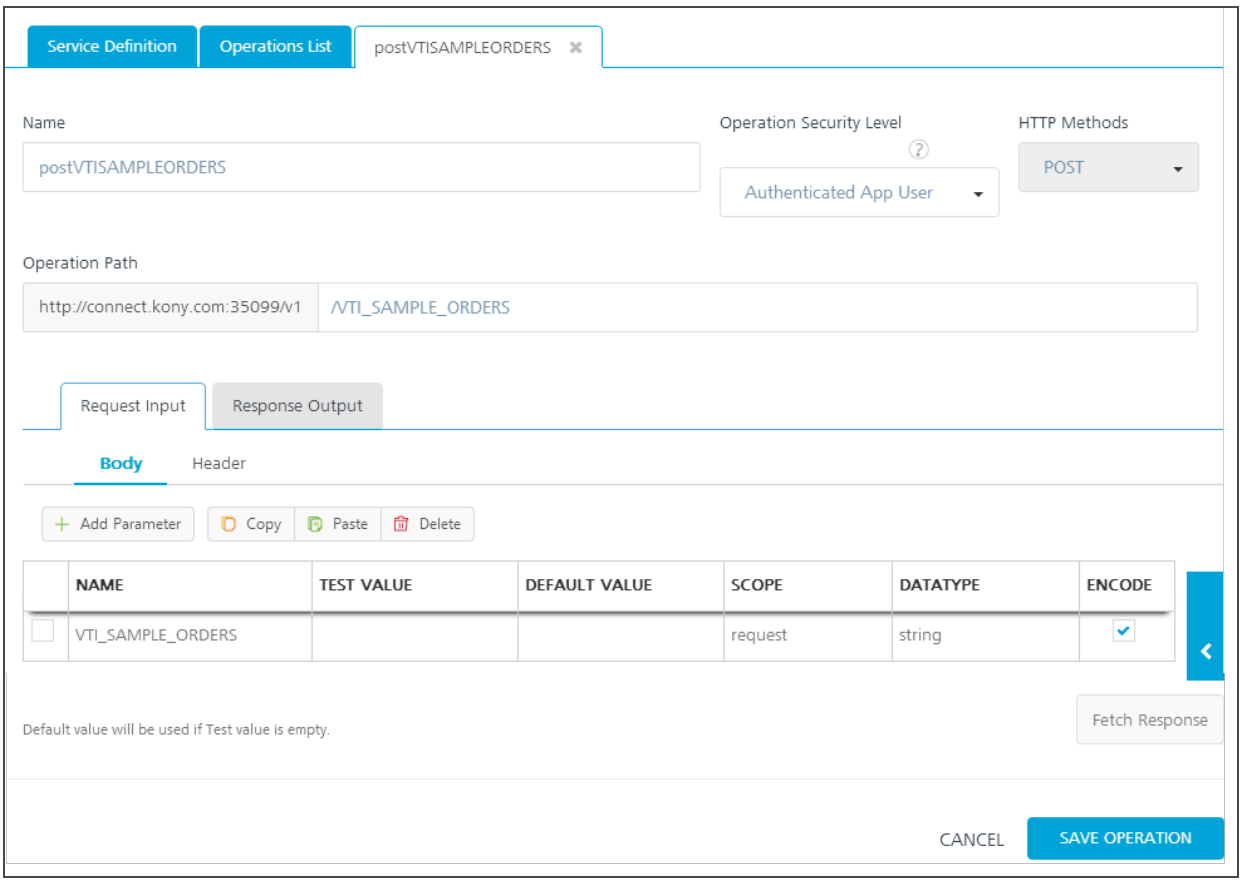
section, and it also adds your new Kony SAP Gateway service into the **Integration** page.



13. To configure operations, under **Operations** > **Configured Operations**, hover your cursor over the required service, click the **Settings** button, and then click **Edit**.



The operation details are displayed.



14. In the **Name** box, modify the name if required.
15. Select one of the following security operations in the **Operation Security Level** field. By default, this field is set to **Authenticated App User.**
    - **Authenticated App User** – indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.
    - **Anonymous App User** – indicates that a user must have the app key and app secret to access this operation.
    - **Public** – indicates that this operation requires no special security.

16. In the **Operation Path** box, modify the path if required.

> ***Note:*** You can add an entry by clicking the **Add** button if entries for the input and the advanced tabs do not exist. You can also delete an existing entry by clicking the **Delete** button.

17. In the **Request Input** tab, provide the following information:
    a. The **ID** field contains a unique identifier for a parameter. Change the identifier if required.
    b. The **Test value** field contains a value to be used to test the service. Change the syntax if required.
    c. In the **Default value** field, change the syntax if required.
    d. Click **Test** to view the results.
    You can add more properties to an input parameter such as scope and data types by following the below step. Otherwise, proceed to Step 14.
    e. To add more properties, click the **Edit** button. The **Input Parameter Modal** dialog appears.
        i. Select request or session in the **Scope** field. By default, this field is set to **Request**.
            - **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.
        ii. Select a data type in the **Datatype** field:
            - **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.

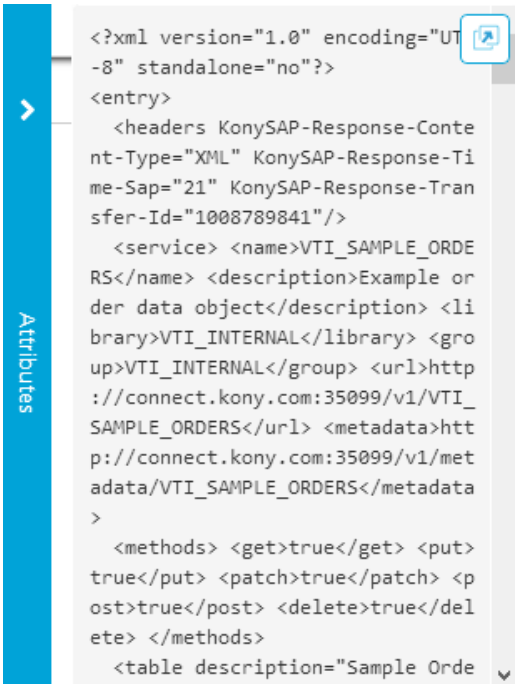
- **Number** - an integer or a floating number.
- **Collection** - a group of data, also referred to as data set.

iii. Select the **Encode** check box 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 to the HTML URL encoding standards.

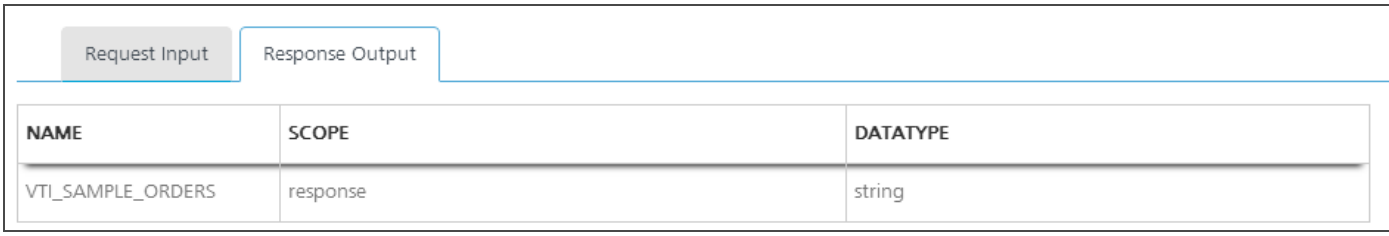
iv. Click **Submit**. The input parameter is saved with additional properties.

18. Click the **Attributes** tab to view schema. This is a meta-data schema for which user has configured at Kony SAP. The schema is in XML format.

    For example, the schema includes elements for the configured tables, such as, table name, description, library name, group name, URL, methods.



19. Click the **Response Output** tab to view the output test values, such as ID, scope, data type. You cannot edit these values.



20. Based on the operation - for example, post or get - provide custom HTTP headers.
    To provide customer headers, click **Advanced**. In the **Test values** text box, provide custom HTTP headers required by the external data source, shown below:
    - **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.
21. Click **Save Operation** to save the operation. They system displays the **Operation** section for your service.
22. Click **Done** to navigate to the **Integration** page.
23. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Edit or Test an Existing Kony SAP Gateway Integration Service

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.

Configure a Identity Provider

To configure a identity provider and use it in creating a service, follow these steps:

1. In Kony Visualizer, launch the Kony Fabric console.
2. In the **Identity** tab, create a new identity service and save it.
3. Navigate to the **Integration** tab, configure a new service.
4. In the **Select Authentication Service** section, select **Use Existing Identity Provider** button. The Select Identity Provider list appears.
5. Select the identity service from list. This list is populated with the identity providers created in the Identity tab.
6. Provide information in all the fields and click **Save**.

Configure a Salesforce Service

Applies to *Kony Visualizer Classic*.

A service that communicates with an external data source using an Salesforce data connector is known as an Salesforce service, and you can construct your application to use multiple Salesforce services from various sources that work together. Configured using the Kony Fabric Console, Salesforce web service manages the interface between the requests of your app to an Salesforce based data source and the data source's responses to those requests.

In this topic, you will learn about:

Configure a Salesforce Service

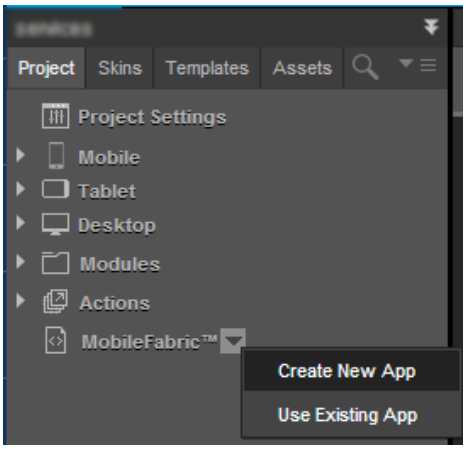
Publishing the Service

Mapping the Service Output to Widgets on a Form.

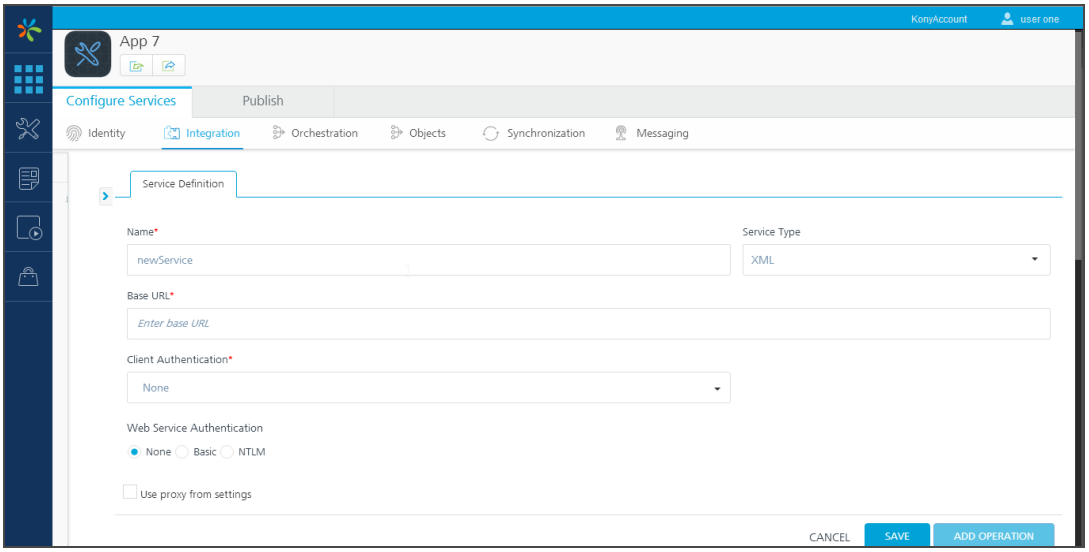
Configure a Salesforce Service

To configure your Salesforce service, follow these steps:

1. In Kony Visualizer, open either an existing application or create a new one.
2. In Kony Visualizer, open either an existing application or create a new one.
3. In Kony Visualizer, open either an existing application or create a new one.



4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.

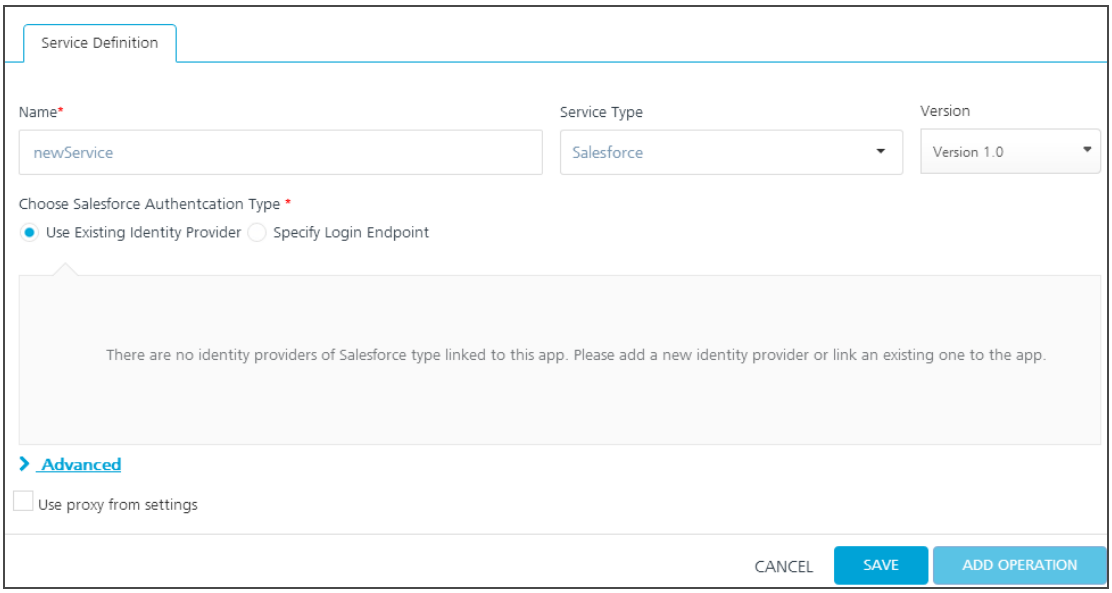


5. Click the **Integration** tab, click **CONFIGURE NEW** to create an integration service.

6. In the **Service Definition** section, follow these steps:

    a. In the **Service Name** text box, enter a unique name for your service.

    b. From the **Service Type** list, select **Salesforce**.
    By default, XML is selected. If you select **Salesforce**, the **Choose Salesforce Authentication Type** section is displayed, shown below.



c. Under **Choose Salesforce Authentication Type**, click one of the following modes:

- 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.
- Specify Login Endpoint- to configure a new endpoint.

To configure Use Existing Identity Provider, 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.

To configure Specify Log-in 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** box, enter a valid client id.

iii. In the **Client Secret** box, enter a valid client secret.

iv. In the **User ID** box, enter a valid user ID.

v. In the **Password** box, enter a valid password.

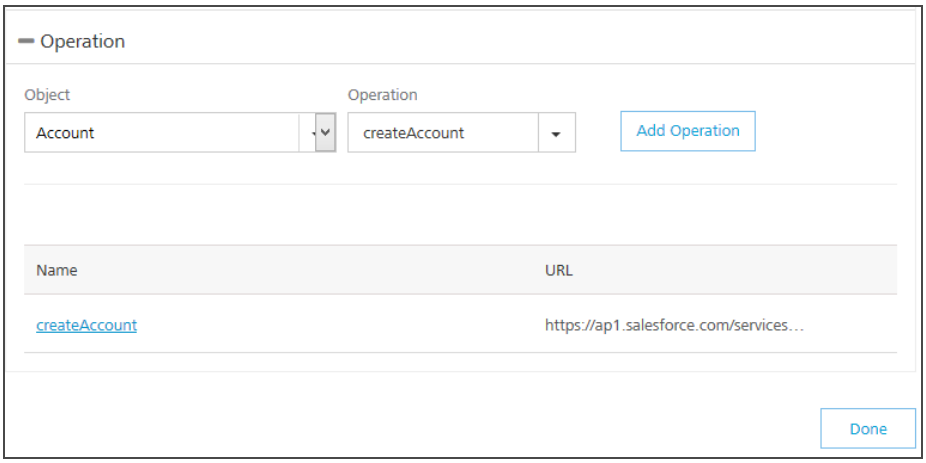
7. After you configure the authentication service, click **Save and Continue** to display the **Operation** section. Based on your Salesforce authentication, the system loads all tables such as objects and operations.

8. 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.

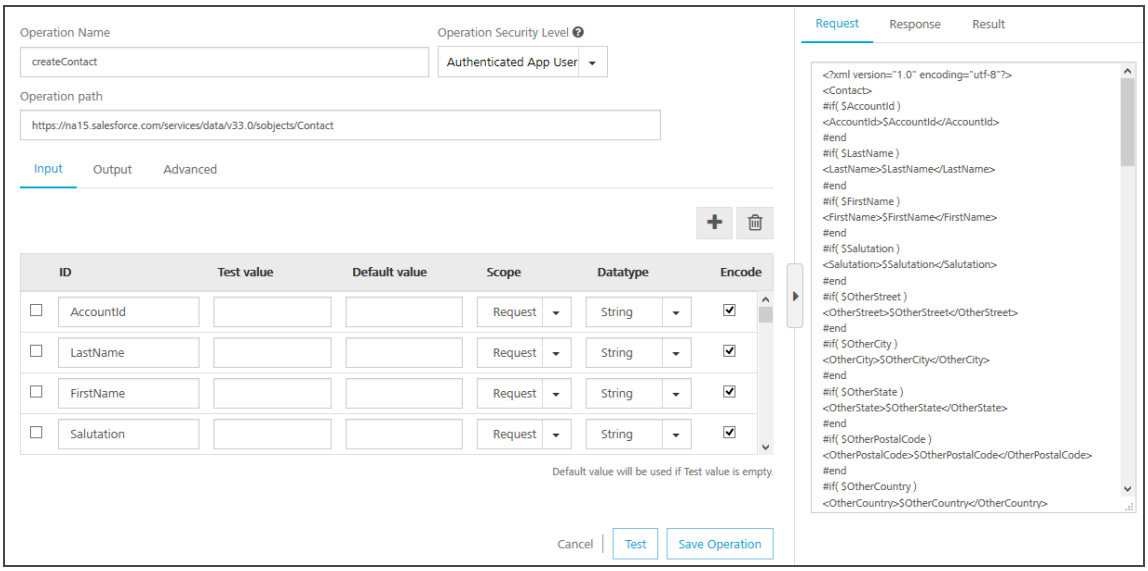
9. Based on the object added, the operation list is populated. Select the check boxes for required operations.

10. Click **Add Operation**. The system adds your operation under the Configured Operations section, and it also adds your new Salesforce service into the Integration page.



11. To configure operations, under **Operations** > **Configured Operations** section, hover your cursor over the required service, click an operation under the **Name** column or click the **Settings** button, and then click **Edit**.

The operation details are displayed.



12. In the **Operation Name** box, modify the name if required.
13. Select one of the following security operations in the **Operation Security Level** field. By default, this field is set to **Authenticated App User.**
    - **Authenticated App User** – indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.
    - **Anonymous App User** – indicates that a user must have the app key and app secret to

access this operation.

- **Public** - indicates that this operation requires no special security.

14. In the **Operation Path** text box, modify the path if required.

> ***Note:*** You can add an entry by clicking the **Add** button if entries for the input and the advanced tabs do not exist. You can also delete an existing entry by clicking the **Delete** button.

15. In the **Input** tab, provide the following details:
    a. The **ID** field contains a unique identifier for a parameter. Change the identifier if required.
    b. The **Test value** field contains a value to be used to test the service. Change the syntax if required.
    c. In the **Default value** field, change the syntax if required.
    d. In the **Scope**, select request or session in the **Scope** field. By default, this field is set to **Request**.
        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.
    e. Select a data type in the **Datatype** field:
        - **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 datatype is string, then the options in the Format Type are Currency, Number and Date.

- If the datatype is number, then the options in the Format Type are Currency and Date.
- If the datatype 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 to as data set.

f. Select the **Encode** check box 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 to the HTML URL encoding standards.

16. 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.

17. Click the **Advanced** tab, and follow these steps:

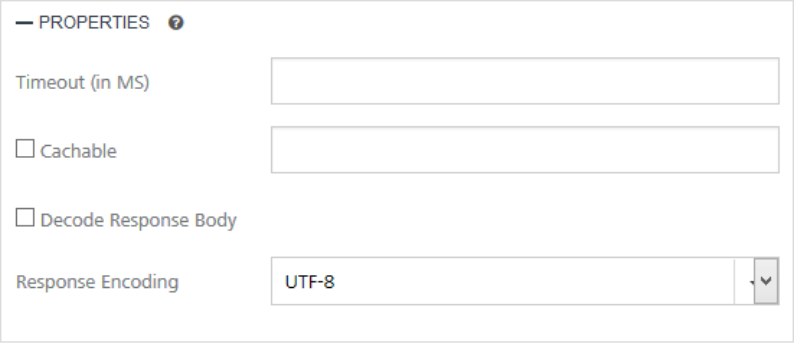
    i. Under the **CUSTOM CODE INVOCATION**, upload the JAR file containing the preprocessor class name and postprocessor class name. This step allows you to further filter the data received from a service call.

    - **Specify Custom Jar** - Browse and select the JAR containing preprocessor or postprocessor libraries.
    - **Preprocessor class** - Enables a developer to include any business logic on the data before forwarding the request to the external data source. Select the JAR file from the list.
    - **Postprocessor class** - Enables a developer to include any business logic on the data before sending the response to a mobile device. Select the JAR file from the list.

ii. Based on the operation - for example, post or get - provide custom HTTP headers. To provide customer headers, click **HTTP Headers** . In the **Test values** text box, provide custom HTTP headers required by the external data source, shown below:

- **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.
- **Default value:** change the syntax if required.
- **Scope:** select request or session. By default, this field is set to **Request.**

iii. 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.
- **Cachable(sec)** - the duration in seconds within which the service response is fetched from the cache. Select the **Cachable(in sec)** check box and provide the details in the text box.
- **Decode Response Body -** To ignore the Salesforce response received in the XMLvalue field, select the Decode Response Body 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.

18. Click **Test** to view the results.
19. Click **Save Operation** to save the operation. They system displays the **Operation** section for your service.
20. Click **Done** to navigate to the **Integration** page.
21. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

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.

Because 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.

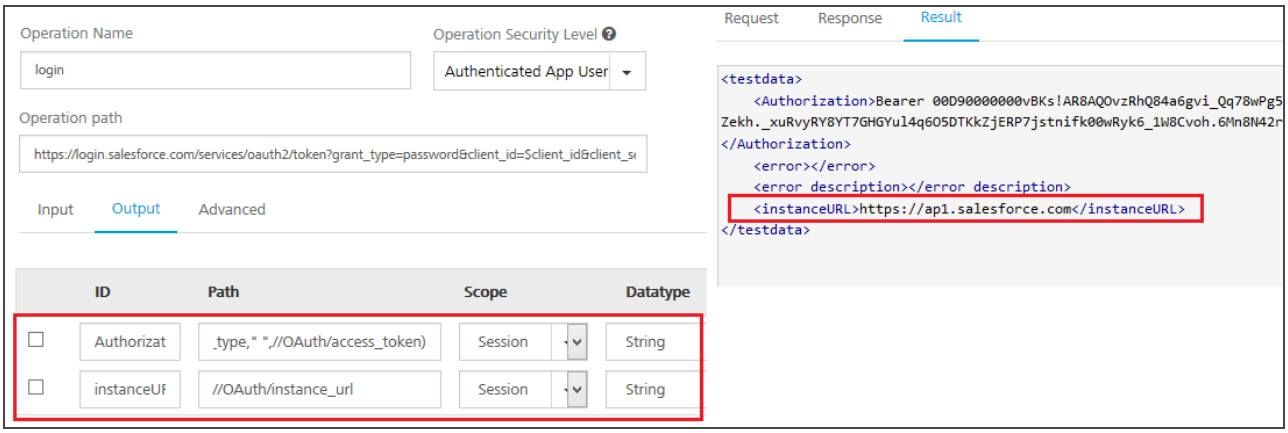
```
Error 401: Request Unsuccessful Server responded with 401.
```

If you want to 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:

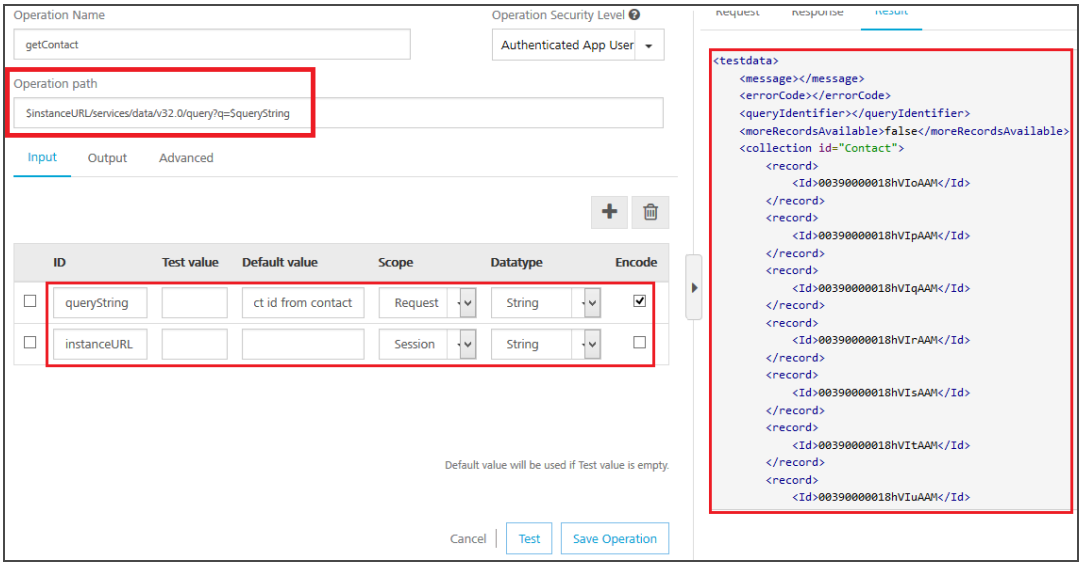
To parametrize an instance URL, follow these steps:

1. In the **login** operation, click **Output** tab, and make the 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 above 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, shown below:

ID	Path	Scope	Encode
queryString	Select ID from contact	session	
instanceURL	//OAuth/instance_url	session	No

5. Click **Test**. The contacts are fetched from the Salesforce.

6. Click **Save Operation**.

Create a MuleSoft Service

Applies to *Kony Visualizer Classic*.

MuleSoft's Anypoint™ Platform helps app developers design custom APIs and deploy them to a Mule Enterprise Service Bus (ESB) runtime. With the MuleSoft integration service in Kony Fabric, developers can interact with over 50 types of connectors.

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 connector and builds a RAML file over the project, all the APIs can be used in a Kony Fabric integration service. When a Kony Fabric user selects a MuleSoft connector from the Kony Fabric Console, based on the cloud hub portal credentials of the MuleSoft connector, the system retrieves metadata from the RAML file and displays all its APIs. Developers can add these APIs as operations in the MuleSoft integration service in the Kony Fabric Console.

Kony Fabric discovers the MuleSoft endpoints through the RAML file, parsing it and exposing all the MuleSoft endpoints through the integration service. Mobile app developers can use the configured MuleSoft integration service to access the backend systems supported by MuleSoft's connectors.

> ***Note:*** In the Kony Fabric Console, you can provide login credentials for MuleSoft or upload a RAML file to configure an integration service.

Advantages of Kony Fabric database connector:

- Developers can design custom APIs
- Developers can use more than 50 types of connectors.

Limitations of Kony Fabric MuleSoft connector:

- 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.

Prerequisites

- Log in to MuleSoft with your credentials at https://anypoint.mulesoft.com/#/signin.
- Download MuleSoft Anypoint Studio from https://www.mulesoft.com/platform/studio.
- Create a project and deploy it.
- Build a RAML file. For more information, refer to the MuleSoft documentation at https://developer.mulesoft.com/anypoint-platform.

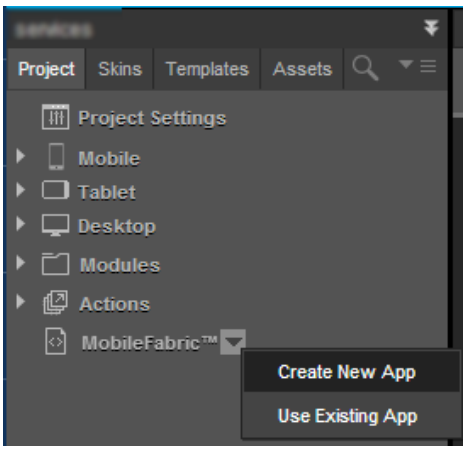
Adding a MuleSoft service involves the following steps:

- Configure a Service Definition for a MuleSoft Service
- Create Operations for a MuleSoft Service
- Configure Operations for a MuleSoft Service

Configure a Service Definition for a MuleSoft Service

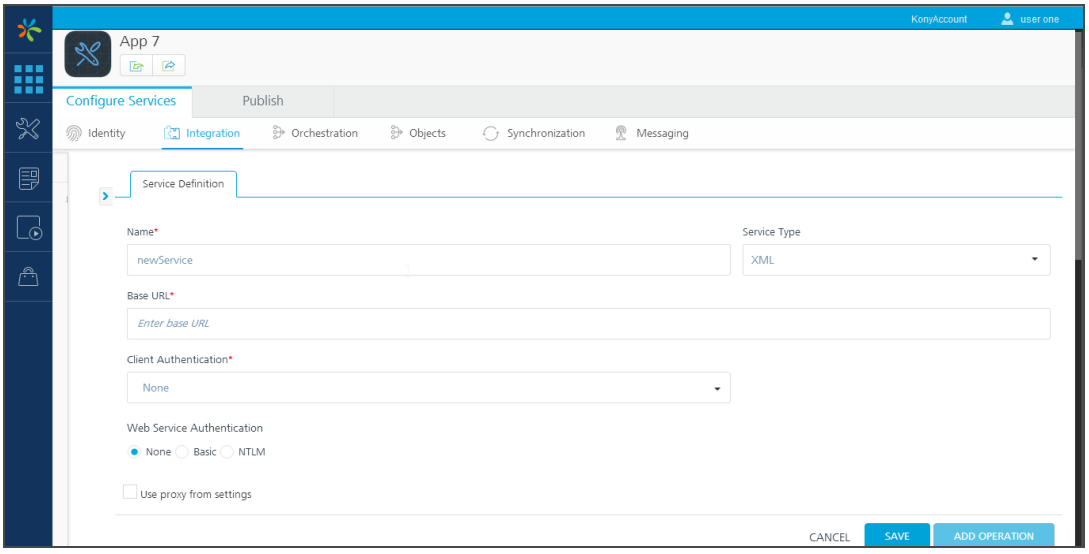
To configure a service definition for a MuleSoft service, do the following:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.



> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.

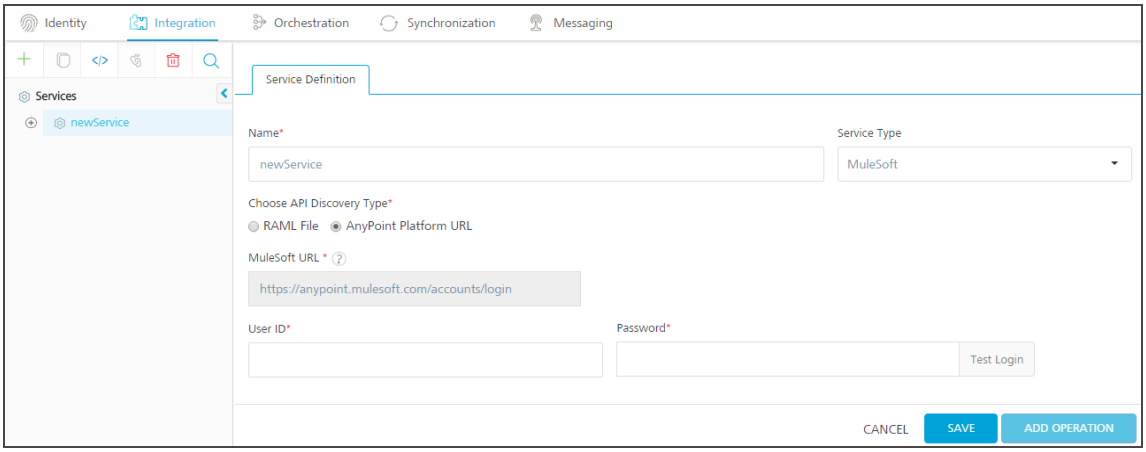


5. On the **Integration** tab, click **CONFIGURE NEW** to create an integration service.

6. In the **Service Definition** section, follow these steps:

    a. In the **Name** text box, enter 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.

    b. From the **Service Type** list, select **MuleSoft**.
    By default, XML is selected. If you select **MuleSoft**, the below details are displayed, shown below.



    c. Under **Choose API Discovery Type**, click one of the following modes:

- To upload a **RAML File**, follow these steps:

    i. Click **RAML File**.

    ii. Click the **Upload RAML File** and select the 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 **Choose API Discovery Type** section.

- To use **AnyPoint Platform URL**, follow these steps:

i. Click **AnyPoint Platform URL**. The system displays the MuleSoft URL in the 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 `MuleSoft AnyPoint` Platform.

To test your database connection details, click **Test Connection**. If the entered details are correct, the system displays the message: Valid MuleSoft connection details.

> ***Note:*** For on-premises, proxy support is not available for MuleSoft service.

d. Click the **Advanced** tab to specify dependent JAR and API throttling. All options in the Advanced section are optional.

- **To specify dependent JAR, follow these steps**:

  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. The 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.

- **API throttling** enables you to limit the number of request calls within a minute. If an API exceeds the throttling limit, the API will not return the service response.
  To specify throttling, 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, 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.

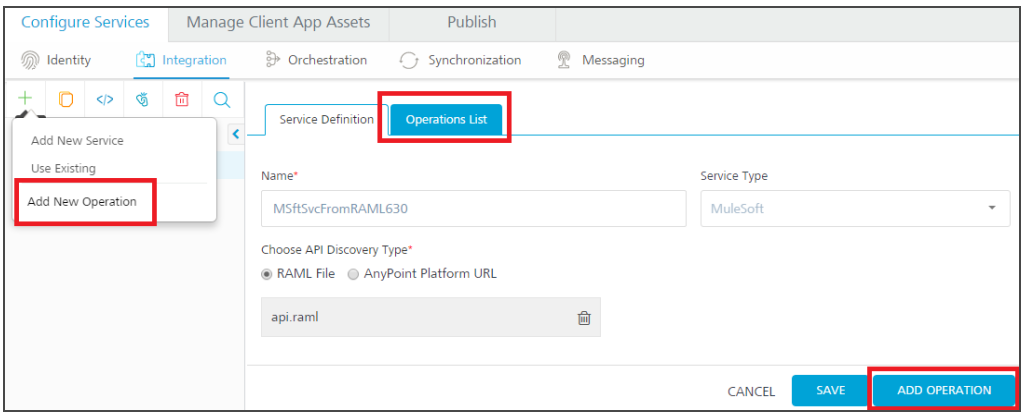
e. To enable the proxy, select the **Use proxy from settings** check box. By default, the check box is cleared.
The **Use proxy from settings** check box dims when no proxy is configured under the **Settings > Proxy**.

f. Click **SAVE** to save your service definition. The system displays the success message: Service Saved Successfully.
The **Operations List** tab appears only after the **Service Definition** is saved. For creating operations for a MuleSoft service, refer to How To Create Operations for MuleSoft Service.

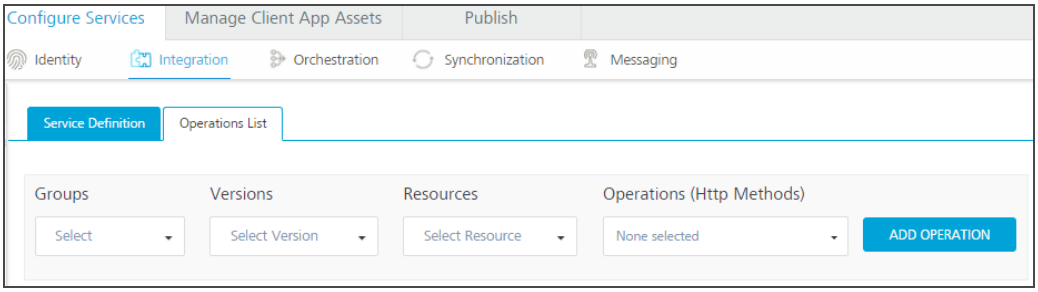
Create Operations for a MuleSoft Service

1. After you configure the service definition for a MuleSoft service, click **ADD OPERATION** to display the **Operations List** tab.

   > ***Note:*** You can also display the **Operations List** tab, by following steps:
   > - Click the **Operations List** tab.
   > - From the tree in the left pane, click **Add > Add New Operation**, shown below:



2. In the **Operations List** tab, follow these steps to configure operations:



3. From the **Groups** drop-down list, 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.

4. From the **Versions** drop-down list, select the required version. You can select only one version of the RAML file.
5. From the **Resources** drop-down list, select the required resource.
6. From the **Operations (Http Methods)** drop-down list, select the required check boxes for methods that you defined in the RAML file. You can click **Select all** check box to select all the operations.
7. Click the **ADD OPERATION** button. The new operations are created under the **Configured Operations** section.

   The system creates a JSON service for the MuleSoft service.

   The default name format of a MuleSoft operation is `<operation_name><resource_name>`. You can change the operation name if required.
   For example, `postfolder`.

Configure Operations for a MuleSoft Service

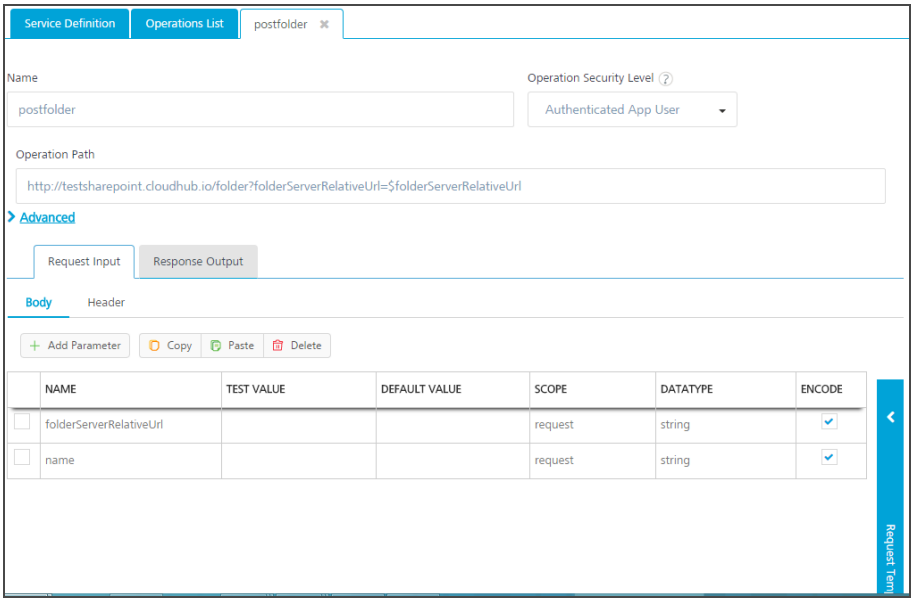
Once you create operations for MuleSoft service, you can configure the operations such as adding parameters, adding test values, and fetching the response.

To configure operations for a MuleSoft service, do the following:

1. Under **Configured Operations**, hover your cursor over the create operation, click the **Settings** button, and then click **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.



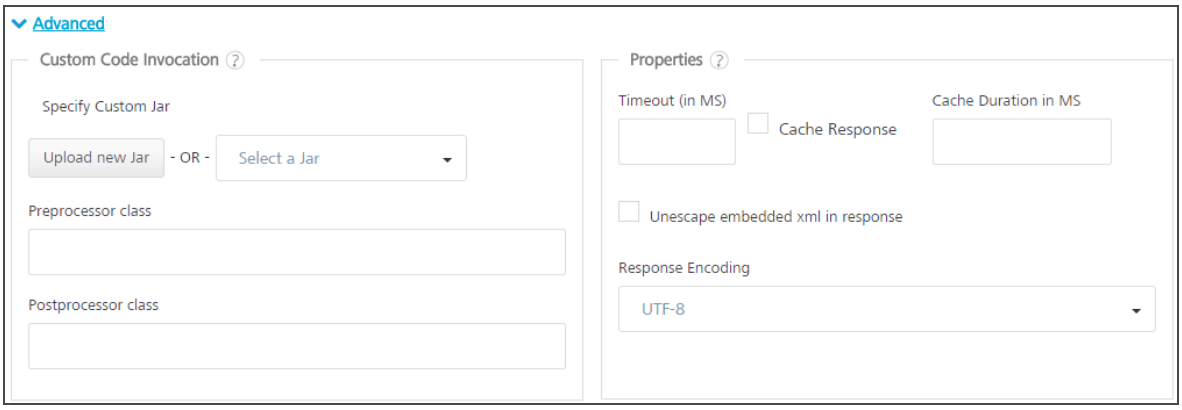
   The **Name** field is pre-populated with operation names. You can edit this field if required.

2. Select one of the following security operations in the **Operation Security Level** field. By default, this field is set to **Authenticated App User.**

   - **Authenticated App User** – indicates that this operation is secured. To use this operation, an app user must be authenticated by an associated identity service.
   - **Anonymous App User** – indicates that a user must have the app key and app secret to access this operation.
   - **Public** – indicates that this operation requires no special security.

   The **Operation Path** field is pre-populated with MuleSoft URL. You can edit this field if required.

3. Click the **Advanced** tab, and follow these steps:



   i. Under the **Custom Code Invocation**, upload the JAR file containing the preprocessor class name and postprocessor class name. This step allows you to further filter the data received from a service call.

      - **Specify Custom Jar** - Browse and select the JAR containing preprocessor or postprocessor libraries.
      - **Preprocessor class** - Enables a developer to include any business logic on the data before forwarding the request to the external data source. Select the JAR file from the list.
      - **Postprocessor class** - Enables a developer to include any business logic on the data before sending the response to a mobile device. Select the JAR file from the list.

   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.
      - **Cachable(sec)** - the duration in seconds within which the service response is fetched from the cache. Select the **Cachable(in sec)** check box and provide the details in the text box.

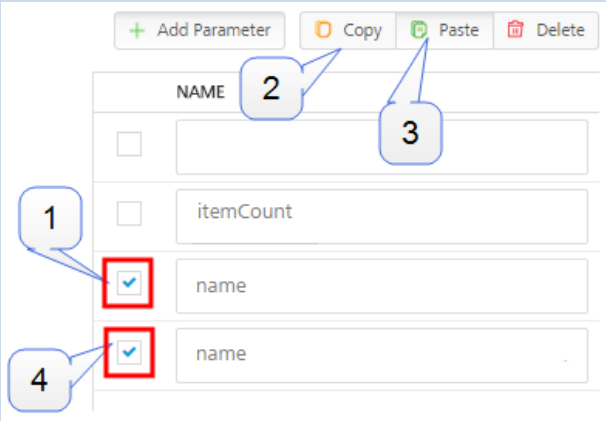
- **Unescape embedded xml in response -** To ignore the MuleSoft response received in the XML value field, select the Decode Response Body response check box.
- **Response Encoding -** select the appropriate response encoding. The default value is UTF-8. For more information about different encoding schemes, refer to Response Encoding Schemes.

4. In the **Request Input** tab, do the following:

   Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

   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.Integration services accept only `form-url-encoded` inputs for all input parameters provided in service input parameters (request input).

a. Under the **Body** tab, do the following:

   i. In the **NAME** field, enter the name for the request input parameter.

   > ***Note:*** In the **Body** tab > **NAME** filed, the input parameters are pre-populated based on the properties of the input schema of a RAML file.

   ii. In the **TEST VALUE** filed, enter the user input for the selected column.

   iii. In the **DEFAULT VALUE**, enter the value if required. The default value will be used if the test value is empty.

   iv. Select request or session in the **Scope** field. By default, the Scope field is set to Request.

   - **Request** - Indicates that the value must be retrieved from the HTTP request received from a mobile device.
   - **Session** - Indicates that the value must be retrieved from the HTTP session stored on Kony Fabric.

   The default data type for the selected column is loaded under the **DATATYPE** field.

   v. Select the **Encode** check box 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 to the HTML URL encoding standards.

b. Under the **Header**, do the following:
   Based on the operation - for example, post or get -, provide custom HTTP headers.
   To provide customer headers, click **Header**.

   i. In the **NAME** text box, provide custom HTTP headers required by the external source.

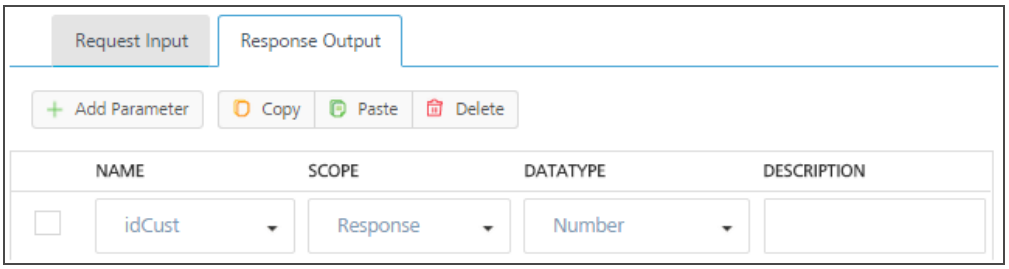
ii. **TEST VALUE**: Enter a value. A test value is used for testing the service.

iii. **DEFAULT VALUE:** change the syntax if required.

iv. **SCOPE:** select request or session. By default, this field is set to **Request.**

To validate the details, click **Fetch Response**. The result of the operation appears.

5. In the **Response Output** tab, configure the fields of the table for displaying the data:



The **Name** field in the Response Output tab is pre-populated with 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.

To validate the details, click **Test**. The result of the operation appears.

6. Click **SAVE OPERATION** to save the changes in the create operation.

7. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Orchestration

Applies to *Kony Visualizer Classic*.

Service orchestration is the coordination or integration of several services and exposing them as a single service. The mix of services supports the automation of business processes.

The following types of Orchestration services are available in Kony Fabric:

- Composite Services: Allows you to run two or more services either concurrently or sequentially.
- Looping Services: Allows you to run a single service in a loop till the loop ends or an exit criteria is met.

Composite Services

Kony Fabric supports the following type of Orchestration Services:

Composite Services is a combination of two or more services. You can specify the order of execution for the services. The order can be sequential or concurrent.

- **Sequential** indicates that the services will be invoked in the order you specify. And also, the output of one service becomes an input to the next service. This service is useful in the following case:
    - **Movie booking portal**: After you purchase a movie ticket, you are provided with an option to order food items that would be delivered at a specified time during the movie.
- **Concurrent** indicates that the services are invoked independent of each other. Invoking and execution of one service does not affect the execution of other services.
    - News Portal: Portal such as Google News (http://google.com/news) run multiple services simultaneously to get news on various topics such as business, technology, entertainment, sports, and Science.

Looping Services

Looping Services is a type of service that runs in a loop for the specified number of times. The specified service gets invoked and executed multiple times till the loop ends or an exit criteria is met.

The looping service is useful in the following cases:

- If you want to fetch the stock price and display it.
- If you want to retrieve the weather details.
- If you want to display the current news.

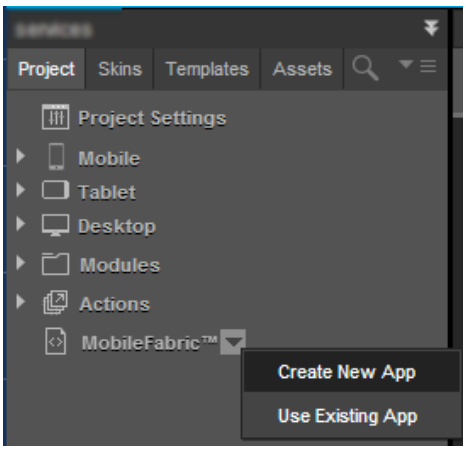
Features of Orchestration Service

- You can select multiple services or operations.
- All the defined services need not be of the same type. For example, your Orchestration services can contain a combination of Looping service, Concurrent service, and Sequential service.
- You can add an existing Orchestrations services to a new service.
- For creating a new composite service, you select the services from the existing **Integration Services** and **Orchestration Services**.

Creating a New Composite Service

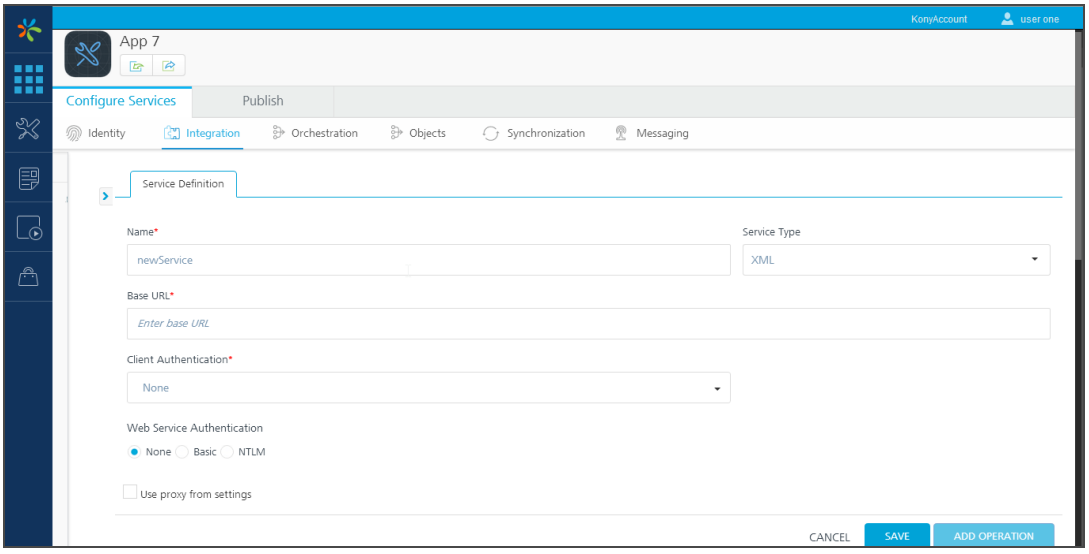
To create a composite service, follow these steps:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony user account, and then click **Sign in**.
3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.

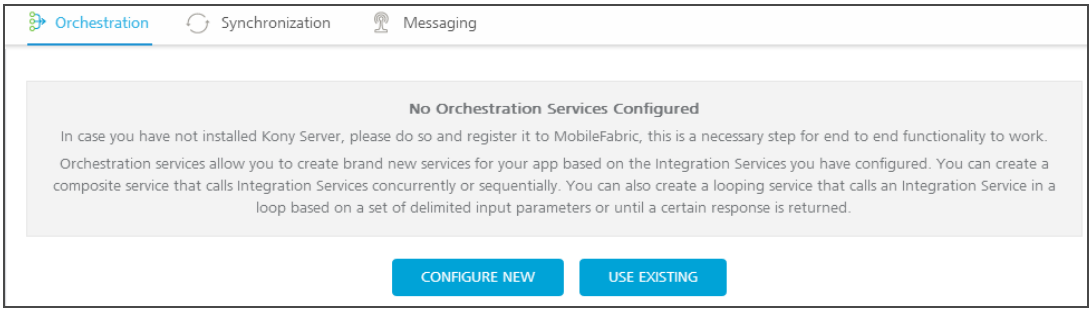


> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

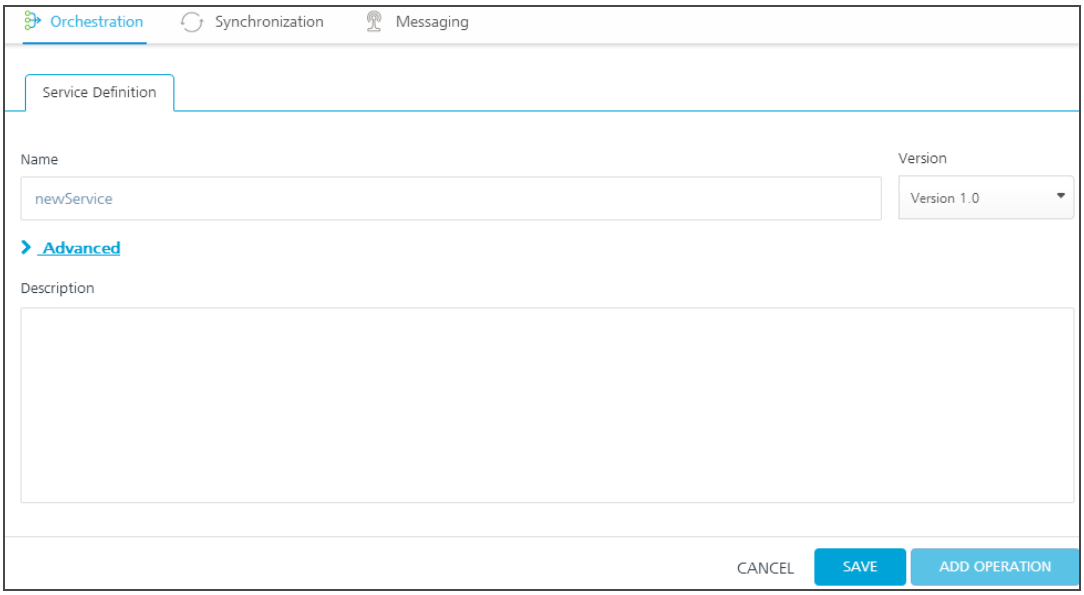
4. To create a new integration service, on the **Integration** tab, click **CONFIGURE NEW**. The **Service Definition** section appears.



5. Go to the **Orchestration** tab. The page lists the existing services (if any).



6. Click **CONFIGURE NEW**.



7. The **Composite Services Configuration** page is displayed. Update the following details :
    a. **Name** : Type a new name for this service. Ensure that name is unique and does not conflict with an existing service.

> ***Note:*** Service name should not contain special characters, must begin with a letter, and should be between 4 and 30 characters long.

    b. Select a JAR File from the list or you can click **Upload New**to add a new JAR file.

> ***Note:*** JAR files are available globally within the same Kony account. That is, a JAR file created for one app is available for another app provided that both the apps are created using the same Kony account.

c. Add a description for the service in **Description** text area.

d. Click **Save**.

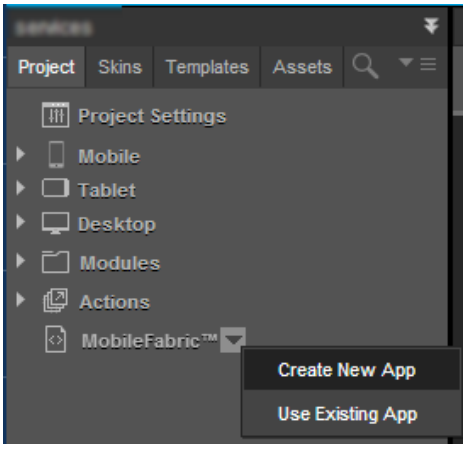
8. Click **Add Operation**. The New Operation tab appears.
9. Provide a name and security level for the operation.
10. In the Operation Type list, select **Composite**.
11. Select a service execution mode from the list.
12. Click **Advanced** link and provide a custom code if required.
13. Click **Save Operation**.
14. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Creating a New Looping Service

To create a Looping service, follow these steps:

1. In Kony Visualizer, open either an existing application or create a new one.
2. If you have not done so already, log in to your Kony account. To do so, in the top right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Enter your email and password credentials for your Kony Cloud account, and then click **Sign in**.

3. Create a new Kony Fabric application or use an existing one. To do so, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**, and then select from the Kony Fabric Application dialog box the services application that you want to publish. The Kony Fabric Console opens.



> ***Note:*** If you want to associate your Kony Visualizer project with a different Kony Fabric app, on the **Project** tab of the Project Explorer, click the context menu arrow for **Kony Fabric**, and then click **Unlink App**. To link to a different Kony Fabric app, click the context menu arrow for **Kony Fabric**, and then click either **Create New App**, or **Use Existing App**.

4. Go to the **Orchestration** tab. The page lists the existing services (if any).

5. In the **Add Orchestration**, click **Create Looping**.

6. The **Looping Services Configuration** page is displayed. Update the following details :

    a. **Name of the service**: Type a new name for this service. Ensure that name is unique and does not conflict with an existing service name.

    b. **Left Pane**: Lists all the existing Integration Service and Orchestration Services available for the app. From here, select an orchestration Service or an operation from an integration service by clicking **Add**.

    c. **Central Pane**: Displays the operation or the orchestration service added from the left pane. You can delete an operation or a service by clicking the **Delete** button.

d. **Right Pane**: Contains the following options:

   i. **Select Jar File**: Allows you to select an existing JAR file.

   > ***Note:*** JAR files are available globally within the same Kony account. That is, a JAR file created for one app is available for another app provided that both the apps are created using the same Kony account.

   ii. **Choose File**: Allows you to select a new JAR file from your machine. Click **Choose File**, and then select a file from your machine.

   iii. **Preprocessor class**: It is a component that is invoked before passing the data to the external data source. This enables the developer to include any business logic on the data before forwarding the request to the external data source.

   iv. **PostProcessor class**: It is a component invoked after the data is received from the external data source but before that data is returned to the mobile device. This enables the developer to include any business logic on the data before sending the response to the mobile device.

   v. **Configuration Params**: Here, you specify the looping conditions that include:

      - **Max Loop Count**: Specifies the number of times the service included in the looping service must get executed.
      - **Input Parameter Separator**: Specifies the character that separates the input parameters of a service.
      - **Break Loop Parameter Name**: Specifies if there are any breakout parameters for the execution of the looping service. The breakout parameter will be a part of the result returned from the service call within a loop.
      - **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, it breaks out of the loop.

e. Click **Save** to create a new looping service.

f. To close the Kony Fabric Console and return to the panes, views, and tabs of the Kony Visualizer integrated development environment (IDE), from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Workspace icon  . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric services.

Use an existing Service

Kony Fabric allows you to use an existing Orchestration Service.

To use an existing Orchestration Service, follow these steps:

1. Go to the **Orchestration** tab. The page lists the existing services (if any).
2. In the **Add Orchestration**, click **Use Existing**.
3. In the **Existing services** page, a list of existing services are displayed. Hover over a service, click the **Settings** button to view **Share** and **Clone** options.
   - Click **Share**, to reuse an existing service. Changes made to this service will affect 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. Changes made to this service will have no impact on the original service.
4. Based on the type of service (Composite or looping), one of the following page is displayed: the Composite Services Configuration page or the Looping Services Configuration page.

Existing Services - Actions

You can perform following actions on an existing service:

- **Edit**: Allows you to edit a service. 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.

- **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.
- **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.

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

  > ***Note:*** If you wanted to add an unlinked service, see Use an existing Service.

Connect to Back-End Services by using Data & Services Panel

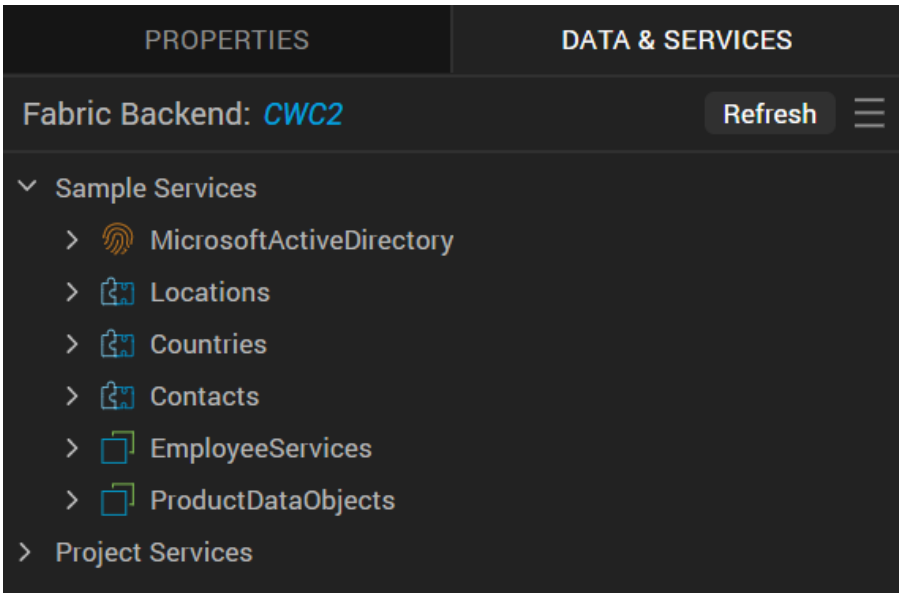
The **Data & Services** panel feature in Kony Visualizer enables you to link back-end data services to your application's user interface (UI) elements seamlessly, with low-code to no-code. You can use the sample services in the Data & Services panel to bind back-end services to your apps and test the UI. If you are an advanced user of Kony Visualizer and have previously created back-end services in your Kony Fabric instance, you can view those services in the Data & Services panel. Furthermore, you can create new back-end services from the Data & Services panel and associate the services with your apps.

The Data & Services panel leverages the **Kony Fabric Stub Back-End Response** feature. For more information on Kony Fabric Stub Back-End Response, click here.

Click here to watch a video on how to integrate data by using the Data & Services panel in Kony Visualizer.

The Data & Services panel is available in both Kony Visualizer and Kony Visualizer Classic, and it contains two lists: Sample Services and Project Services.

The Sample Services drop-down list contains various sample services that you can directly start using in your app. These configured sample services come pre-built with Kony Visualizer.



You can use Project Services to create a new service. You can now create, edit, and delete various services from within Visualizer. These services are as follows:

- Identity
- Integration
- Object

The Data & Services panel consists of the following features:

- Use Sample Services
    - Use a Sample Identity Service
    - Use a Sample Integration Service
    - Use a Sample Object Service
- Use Existing Services
- Create and Use New Services
    - Create an Identity Service
    - Create an Integration Service
    - Create an Object Service
    - Create a Backend Workflow
- Use Project Services
    - Use an Identity Project Service
    - Use an Integration Project Service
    - Use an Object Project Service
    - Use a Backend Workflow
    - Disable/Enable the Categorization of Project Services
    - Unlink a Service
    - Edit a Service
- Object Service-related Features
    - Generate Object Services UI for Responsive Web
    - Generate CRUD Forms for an Object Service

- Create a Data Table for an Object Service
- Configure an Object Data Model by Importing an Excel File
- Customize the Generation of Data Model Objects

- Set Data for the Segment in a Component
- Send Data between Two Forms
- Associated Data & Services Panel Features

Use Data & Services Panel Features

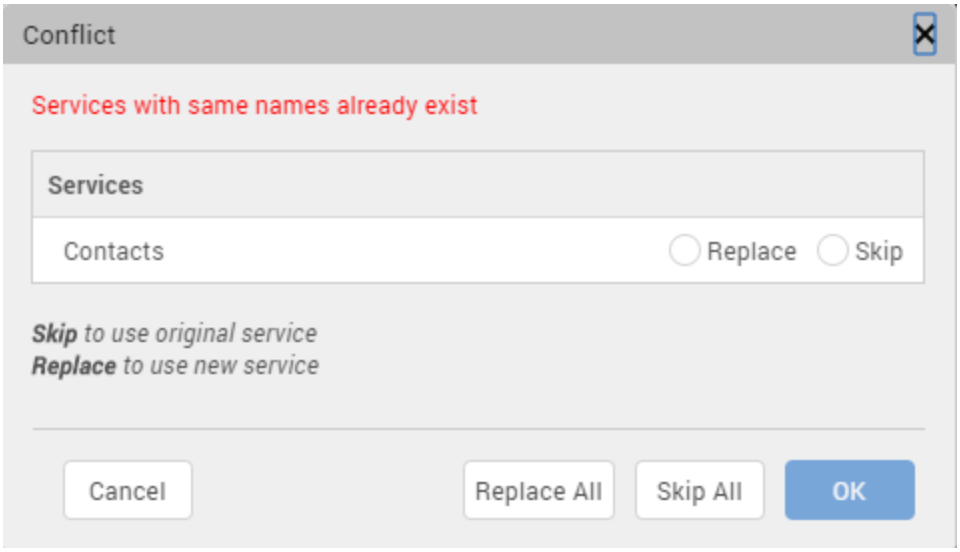
Use Sample Services

This section provides information on how to discover and use existing sample services in an app.

In Visualizer, a set of sample Integration and Object services are bundled. These services are uploaded to Kony Fabric when the sample data is dragged onto a form. When you drag and drop a sample service operation on a form, a project service is added. This is the operation with which the view mapping action is associated.

If the sample service that you are trying to create already exists, a **Conflict** window appears. You must perform any one of the following actions:

- Select the Skip radio button, and then click Ok to use the original service.
    - If multiple instances of the same service exist and you want to use the original service, click Skip All.
- Select the Replace radio button, and then click Ok to use a new service.
    - If multiple instances of the same service exist and you want to use a new service, click Replace All.



If any services are already associated to your Kony Fabric application, you can view them in the Data & Services panel. You can drag and drop these services on to various widgets in a form. When you perform the drag and drop action, the code is auto-generated to bind the UI element to the back-end service.

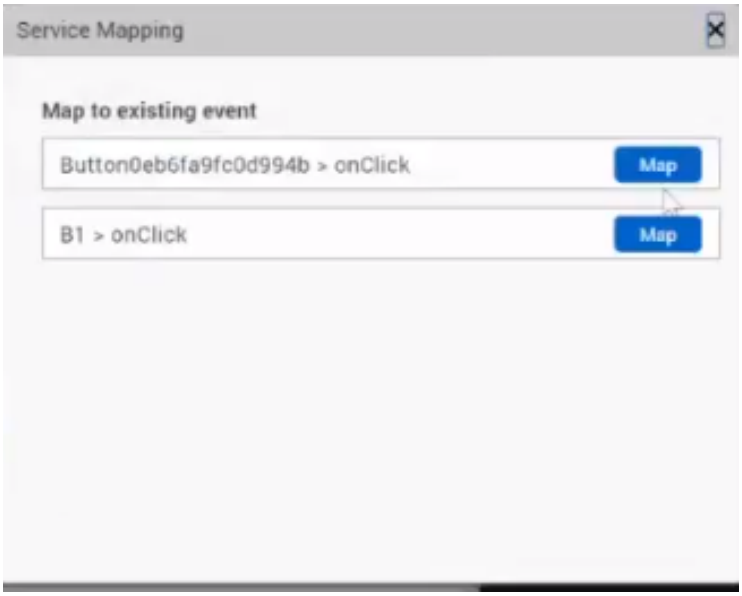
The type of code generated depends on:

- The type of service that is being created (GET, POST, DELETE, UPDATE)
- The type of widget it is being mapped to
- Once a data widget is linked to the Post Operation, the data is sent based on test parameters and not based on the label's text value.

You can use the following types of sample services and bind them to various forms in your application:

- Use a Sample Identity Service
- Use a Sample Integration Service
- Use a Sample Object Service

> ***Note:*** If you invoke the same service and operation on any action in a form or its child widgets, the mapping gets added to that service call. If multiple such service calls exist, you must select the service call to which you want to add the mapping.



Use a Sample Identity Service

You can view Identity services in the Data & Services panel. You can also set up new Identity services. Once you configure the Identity service, you can view the request and response parameters in the Data & Services panel.

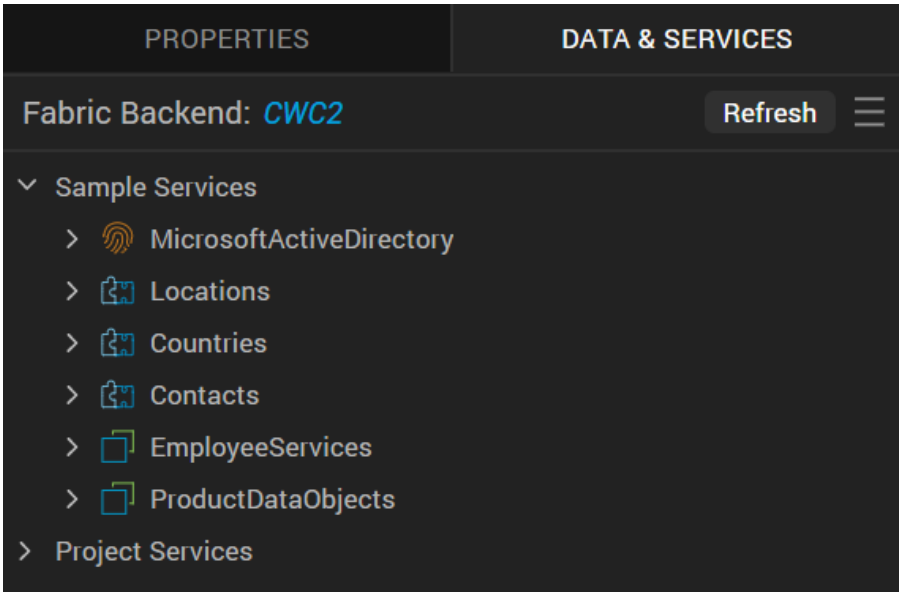
The login credentials for the bundled sample Identity service are as follows:

- **User ID**: testuser@samples.kony.com
- **Password**: test@123

To use a sample Identity service, do the following:

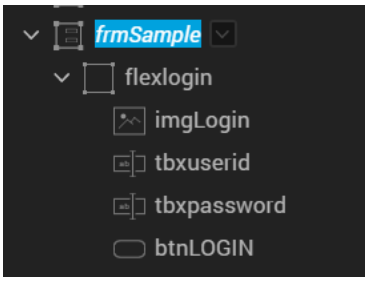
1. In Kony Visualizer, create a new project or open an existing one.
2. Create a form; for example, frmSample.

3. From the **Data & Services** panel, click Sample Services. The list of pre-configured services appears.

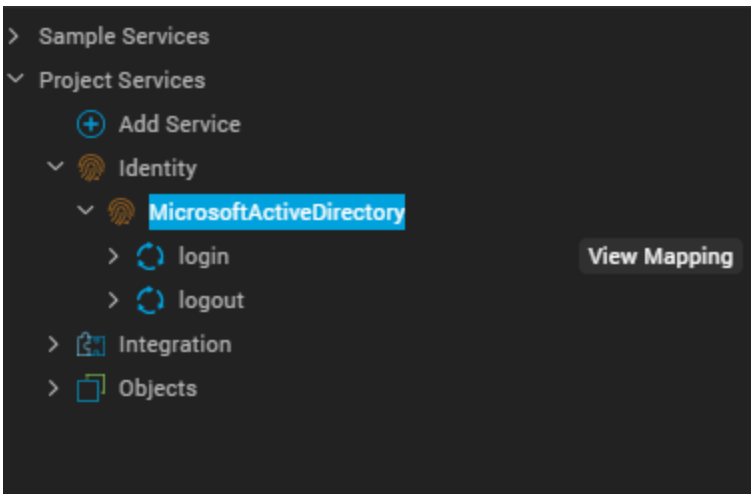


4. Expand **MicrosoftActiveDirectory**; the login and logout services appear.

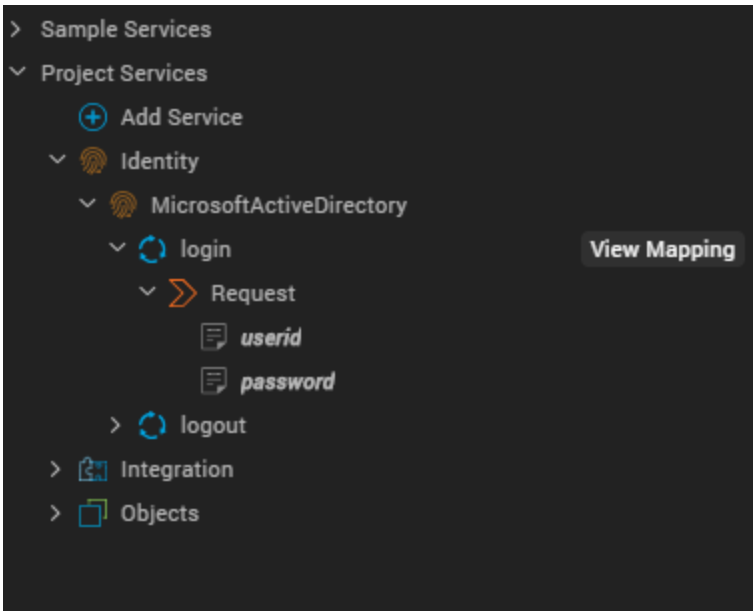
5. Drag and drop the login service on to the form.



6. Post this action, widgets are added to the form and the Identity service is associated with the form. The service also appears under Project Services in an auto-highlighted state. Here, the **MicrosoftActiveDirectory** service is auto-highlighted.

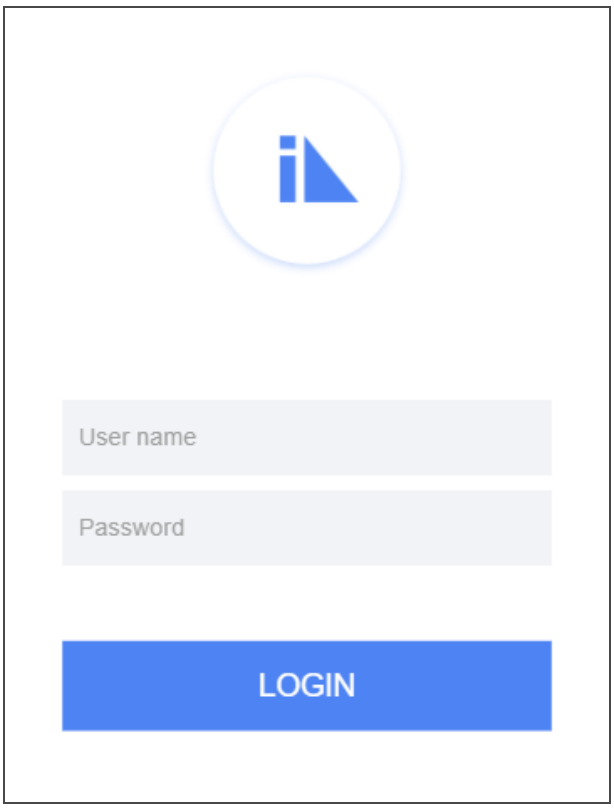


You can expand the **login** service to view its Request parameters. The mapped parameters of any service appear in a bold and italics style to differentiate from unmapped parameters. Here, **userid** and **password** are displayed in a bolded and italicized manner.



You can also click **View Mapping** to view the mapping details of the **login** service in the Action Editor. The specific operation of the service is auto-highlighted in the Action Editor. Click **Generate Code** to view all the mappings of the **login** operation.

Dragging and dropping operations onto the form leads to the auto-generation of UI elements.

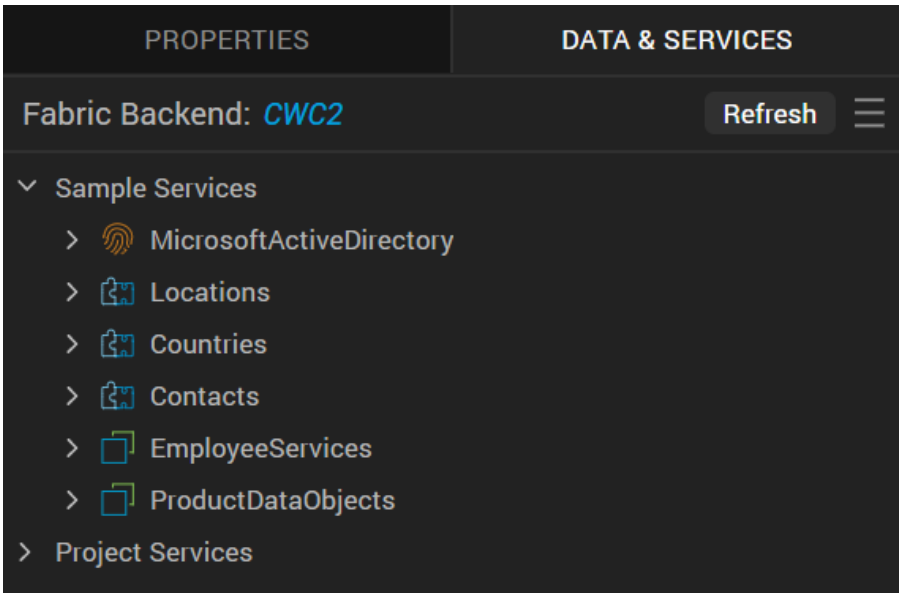


Use a Sample Integration Service

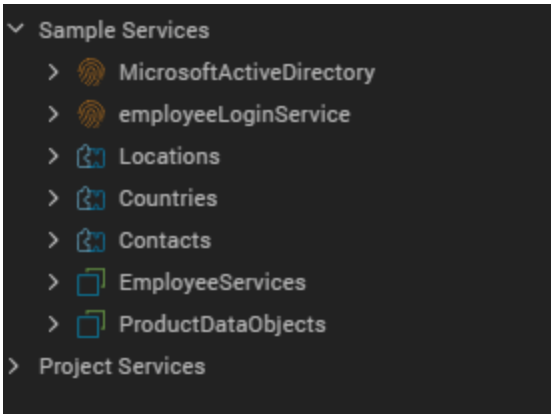
You can view Integration services in the Data Panel. You can also set up new data sources. Once you configure the Integration service, you can view the request and response parameters in the Data & Services panel.

To use a sample Integration service, do the following:

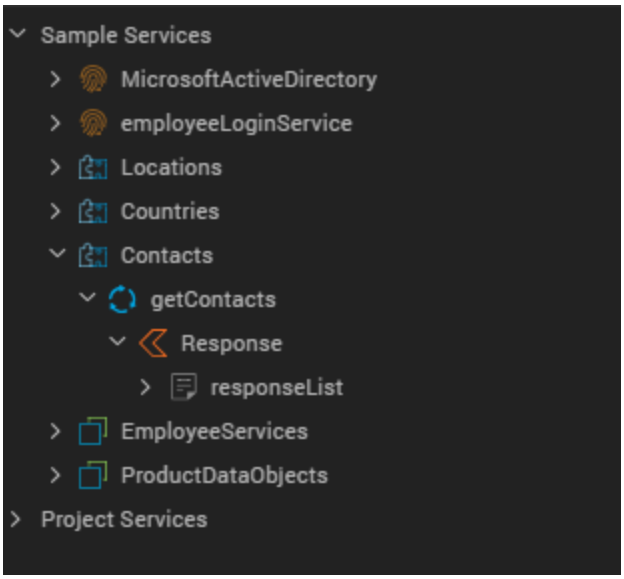
1. In Kony Visualizer, create a new project or open an existing one.
2. Create a form; for example, frmSample.
3. From the **Data & Services** panel, select **Sample Services**. The list of pre-configured services appears.



4. Expand any Integration service. Here, **Locations**, **Countries**, and **Contacts** are the sample Integration services.



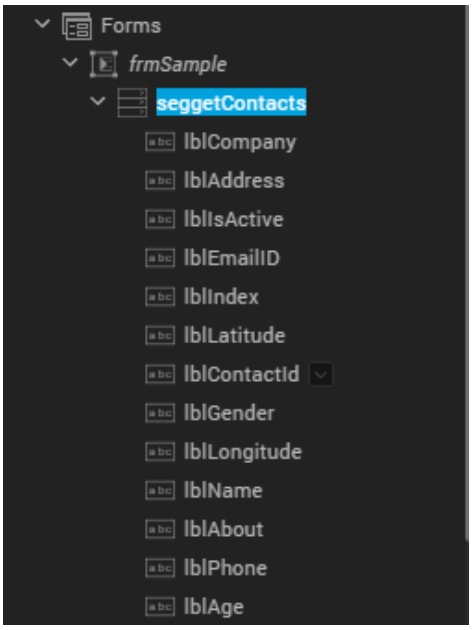
5. Expand any service to view its available Request or Response parameter. The respective responseList or requestList is auto-displayed. Here, **Contacts > getContacts > Response > responseList**.



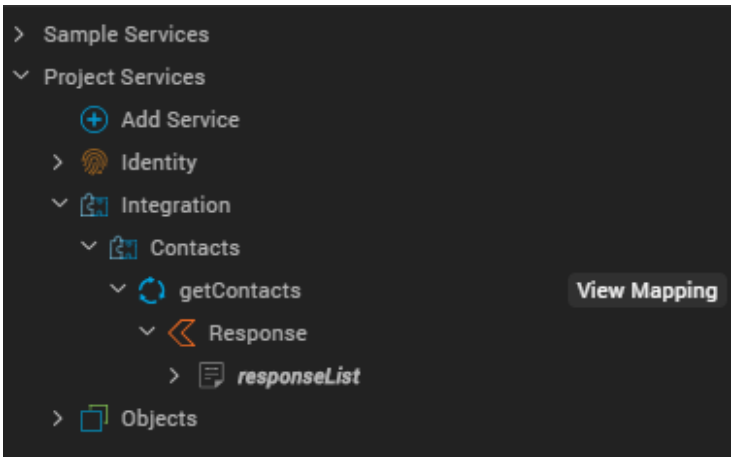
6. You can perform any of the following actions:
    - Drag and drop the service onto your form.
      A dialog box appears asking you to generate either a List, or a Details, or an Entry form. You can select any of the three options as follows:
        - List Using Response: If the response contains a collection, a Segment is added to the parent form. This Segment is mapped to the Response parameter of the service.
        - Details Using Response: A FlexContainer with Labels is added. These Labels are mapped to the Response parameter of the service.
        - Entry form for Request: A FlexScrollContainer containing TextBoxes and Labels is added. The TextBoxes are mapped to the Request parameter of the operation.
    - Alternatively, you can directly drag and drop either the Response or Request parameter of the service onto the form:
        - Drag and drop Response parameter: If a service invocation already exists in the form, the Response parameter is added to the mapping of that service invocation. Otherwise, a new service invocation is added to the form's onMapping Event.

- Drag and drop Request parameter: If a service invocation already exists in the form, this mapping is appended to it. Otherwise, a new service invocation is added on a generated Button widget.

7. Post this action, widgets are added to the form and the Integration service is associated with the form.

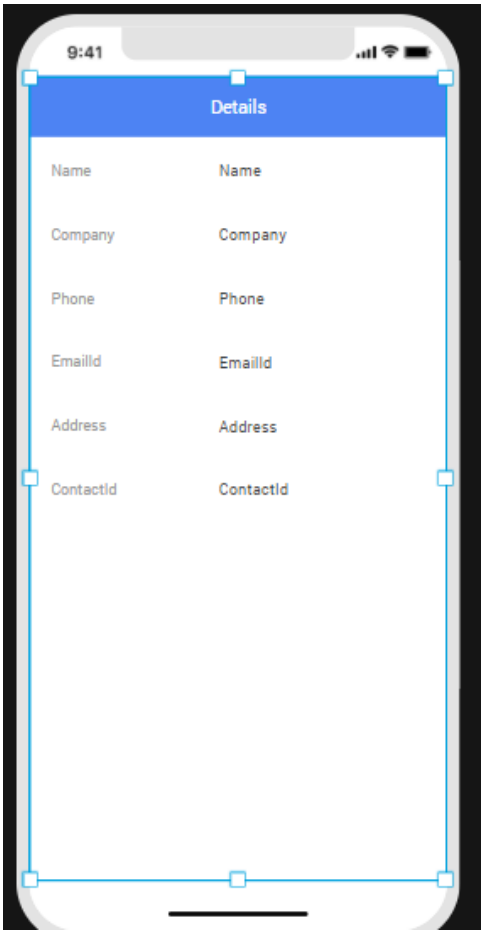


The selected Integration service appears under Project Services. You can expand any service to view its Request or Response parameters. The mapped parameters of any service appear in a bold and italics style to differentiate from unmapped parameters. Here, **responseList** is displayed in a bolded and italicized manner.



You can also click **View Mapping** to view the mapping details of the **getContacts** service in the Action Editor. The specific operation of the service is auto-highlighted in the Action Editor. Click **Generate Code** to view all the mappings of the **getContacts** operation.

Dragging and dropping operations onto the form leads to the auto-generation of UI elements.

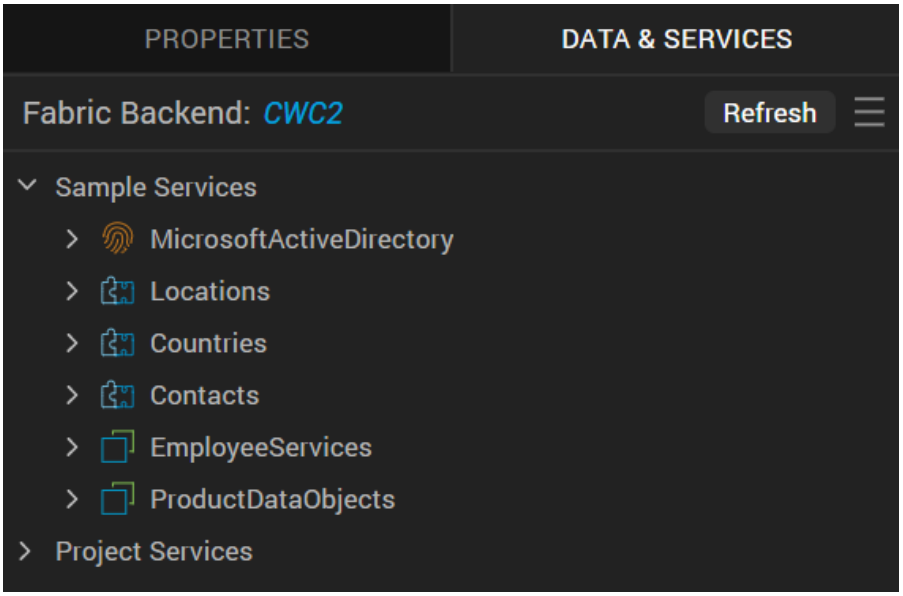


Use a Sample Object Service

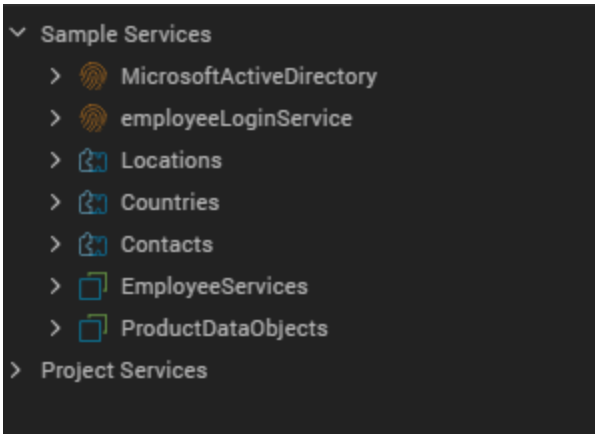
To use a sample Object service, do the following:

1. In Kony Visualizer, create a new project or open an existing one.
2. Create a form; for example, frmSample.

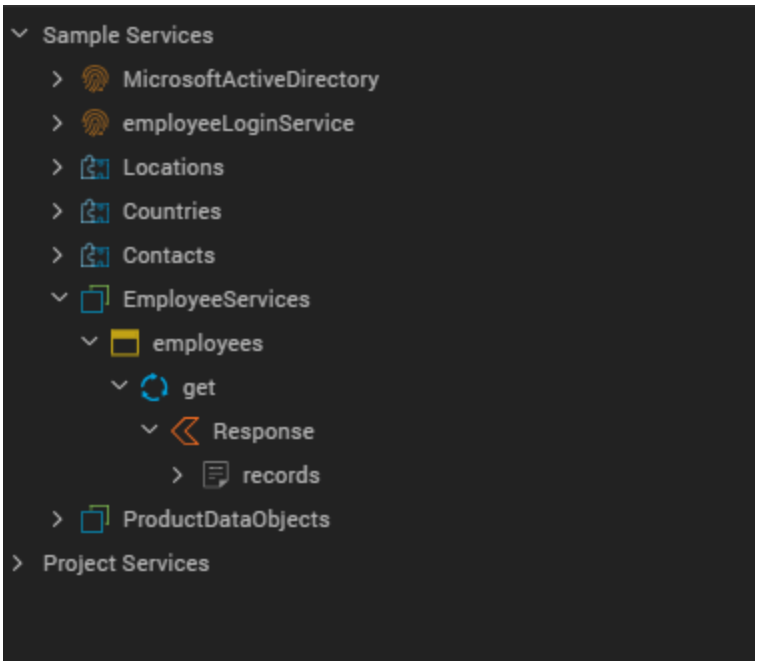
3. From the **Data & Services** panel, select **Sample Services**. The list of pre-configured services appears.



4. Expand any Object service. Here, **EmployeeDataObjects** and **ProductDataObjects** are the sample Object services.



5. Expand any Object service to view its available Request parameter. The respective responseList or requestList is auto-displayed. Here, **Employee Services > employees > get > Response > records**.



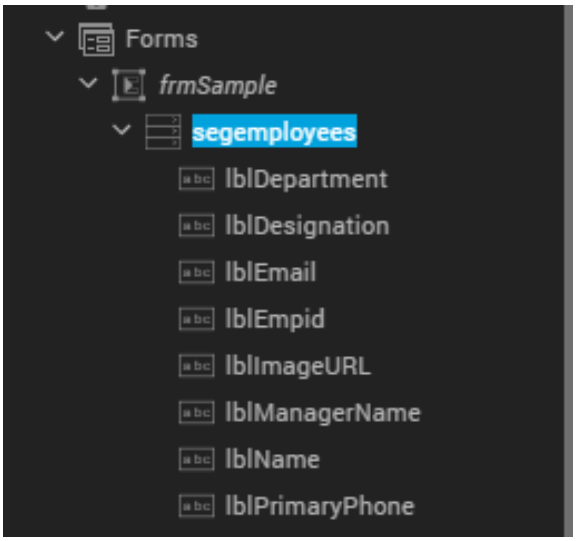
6. You can perform any of the following actions:
    - Drag and drop the service on to your form.
      A dialog box appears asking you to generate either a List, or a Details, or an Entry form. You can select any of the three options as follows:
        - List Using Response: If the response contains a collection, a Segment is added to the parent form. This Segment is mapped to the Response parameter of the service.
        - Entry form for Request: A FlexScrollContainer containing TextBoxes and Labels is added. The TextBoxes are mapped to the Request parameters of the service.

          > ***Note:*** For the GET method in Object Services, the Entry form for Request option is disabled.

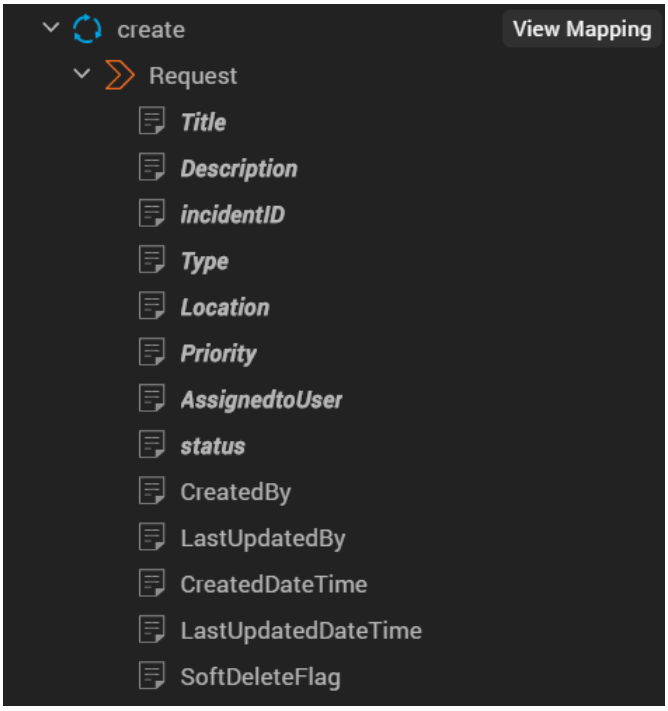
        - Details Using Response: A FlexContainer containing Labels is added. These Labels are mapped to the Response parameters of the service.

- Alternatively, you can directly drag and drop either the Response or Request parameter of the service onto the form:
    - Drag and drop Response parameter: If a service invocation already exists in the form, this is added to the mapping of that service invocation. Otherwise, a new service invocation is added to the form's onMapping Event.
    - Drag and drop Request parameter: If a service invocation already exists in the form, this mapping is appended to it. Otherwise, a new service invocation is added on a generated Button widget.

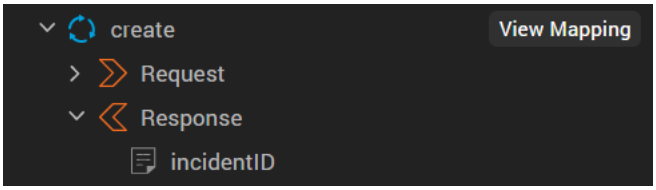
7. Post this action, widgets are added to the form and the Object service is associated with the form.



The selected Object service appears under Project Services. You can expand any service to view its Request or Response parameters. The mapped parameters of any service appear in a bold and italics style to differentiate from unmapped parameters.

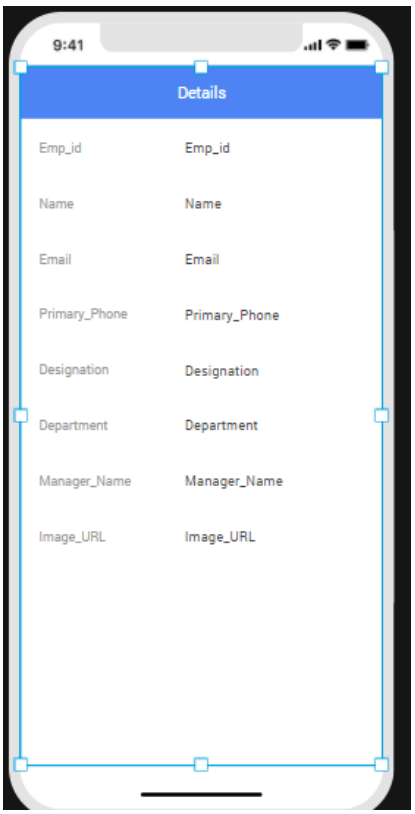


When you use the **create** method with primary key as **incidentID**, a Response parameter is automatically generated based on the run-time response of the app. under **create** > **Response** on the Data & Services panel.



You can also click **View Mapping** to view the mapping details of the **get** service in the Action Editor. The specific operation of the service is auto-highlighted in the Action Editor. Click **Generate Code** to view all the mappings of the **get** operation.

Dragging and dropping operations onto the form leads to the auto-generation of UI elements.



Use Existing Services

This section explains how to use and configure existing Identity, Integration, and Object services in your application by using the Data & Services panel.

From the Data & Services panel, configure existing sample services through a simplified view of Kony Fabric console. You can view the associated services in your corresponding Kony Fabric app.

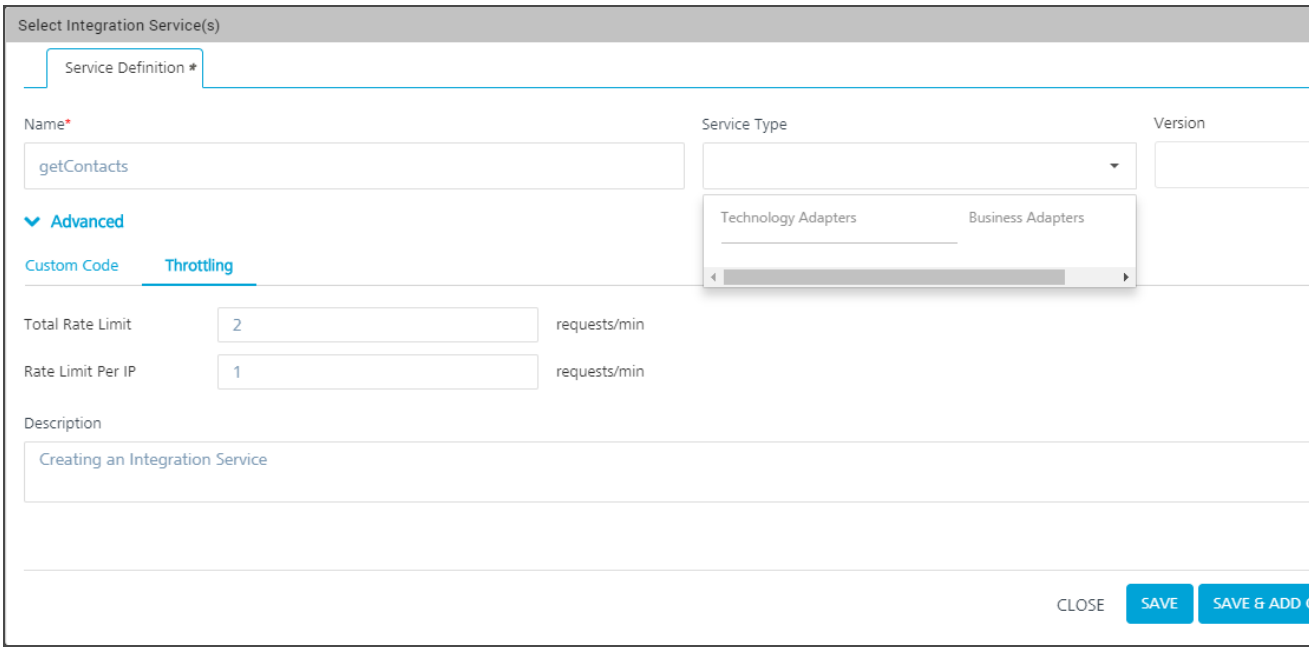
To select and use an existing Identity/Integration/Object service, do the following:

1. In Kony Visualizer, create or open your project.
2. From the **Data & Services** panel, select **Project Services** from the list. The list of available services appears.
3. Click **Use Existing**. A list of options appears.

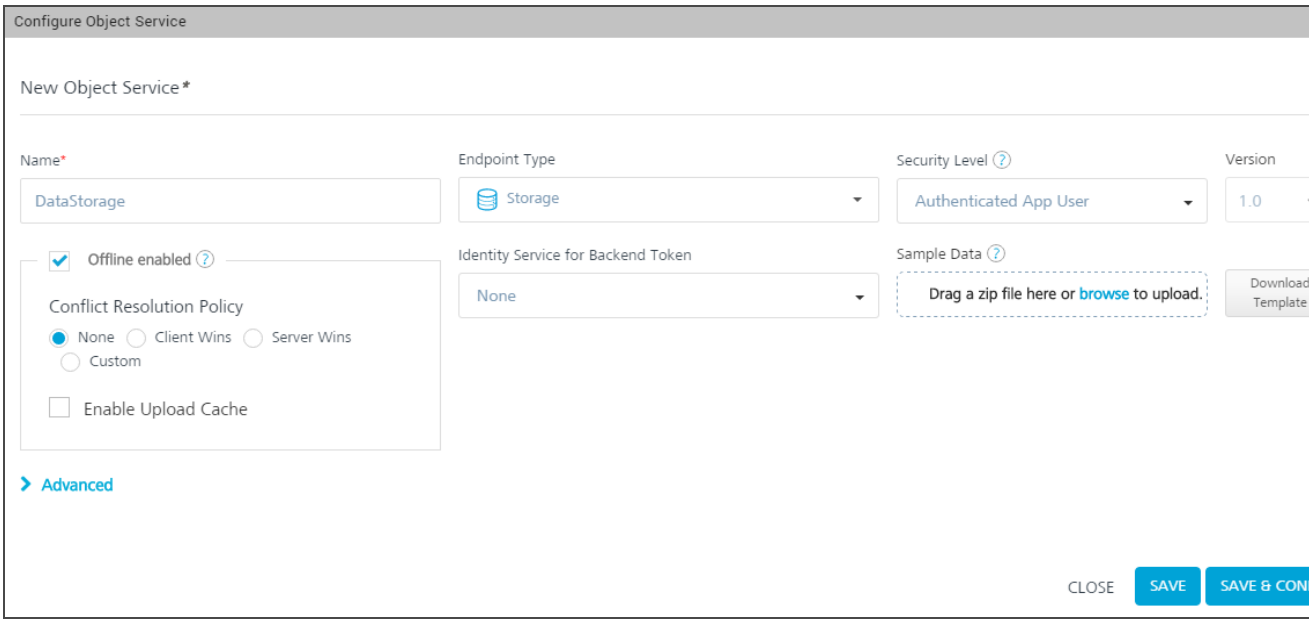
4. Select either **Identity**, **Integration**, or **Objects**, and then sign in to your Kony account to use the required service available in your Kony Fabric instance:

   - For Identity service: The **Select Identity Service(s)** window appears.
     Choose the required Identity service; if no service is available in your Kony Fabric Console, you need to configure one. Specify the details in the required fields, and then click **Save**. For more information on how to create an Identity service, click here.

   - For Integration service: The **Select Integration Service(s)** window appears.
     Choose the required Integration service; if no service is available in your Kony Fabric Console, you need to configure one. Specify the details in the required fields, and then click **Save & Add Operation**. For more information on how to create an Integration service, click here. In this case, the Salesforce Integration service.



   - For Object service: The **Select Object Service(s)** window appears.
     Choose the required Object service; if no service is available in your Kony Fabric Console, you need to configure one. Specify the details in the required fields, and then click **Save & Generate**. For more information on how to create an Object service, click here.



5. Once the service is created, it is displayed in the Project Services list. You can then directly drag and drop the service onto a form and start using it.

Create and Use New Services

This section explains how to create new services by using the Data & Services panel and how to use those services in your application.

From the Data Panel, you can set up new data sources (Kony Fabric services) through a simplified view of Kony Fabric console. You can view the associated services in your corresponding Kony Fabric app.

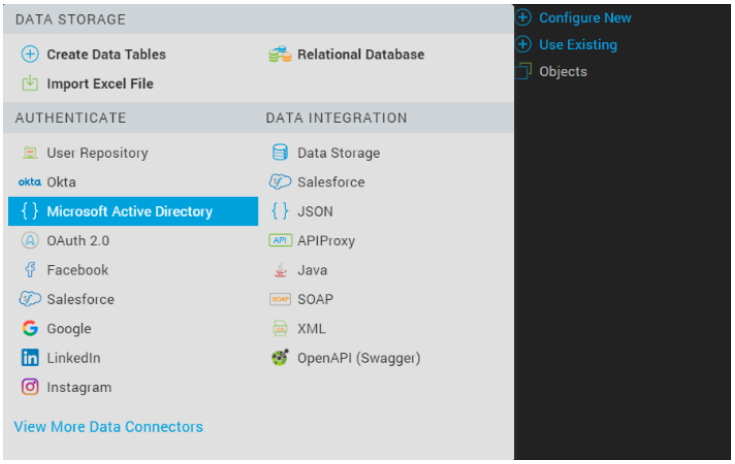
Create an Identity Service

If your Identity service type is a basic one, the input parameters are username and password. Even for the OAuth password grant type of service, the parameters are username and password. For all other types of OAuth services, there are no input parameters. The SAML Identity service follows a similar pattern as the OAuth service.

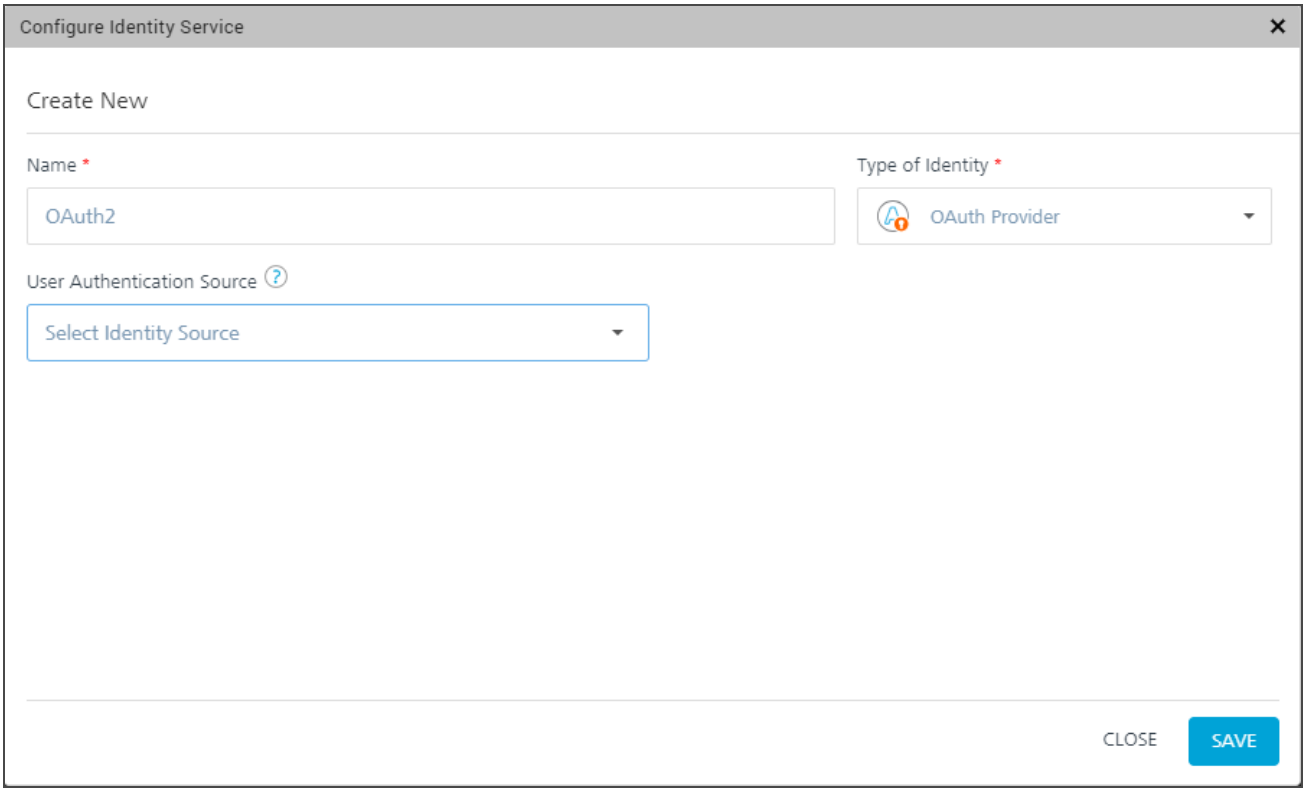
You can view two types of operations for Identity services: login and logout. The SDK code is auto-generated for each operation. You can edit an Identity service by right-clicking on the service and then selecting the Edit option.

To create a new Identity service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, select **Project Services** from the list. The list of available services appears.
3. Click **Configure New**. A list of services that you can configure appears.
4. Under Authenticate, select an Identity service; in this case, **Microsoft Active Directory**.



5. The **Configure Identity Service** window appears, by using which you need to configure the **Microsoft Active Directory** service. Specify the details in the required fields, and then click **Save**. For more information on how to create an Identity service, click here.



6. Once the service is created, it is displayed in the Project Services list. You can then directly drag and drop the service onto a form and start using it.

When you drag and drop an Identity service onto a form, various widgets are created automatically on the form based on the type of service: OAuth or Non-OAuth. For a Non-OAuth service, two text fields (username and password) and a button (Login) are generated. For an OAuth service, only a Login button is generated.

When you click the View button, the newly created form appears. The form's pre-show invokes the Identity service and passes the reference of the Browser widget. You can modify the Browser widget, which you linked to the button, with a different Browser widget.

> ***Note:*** The Data & Services panel supports all types of Identity providers such as Kony User Store, Google, Salesforce, and Facebook.

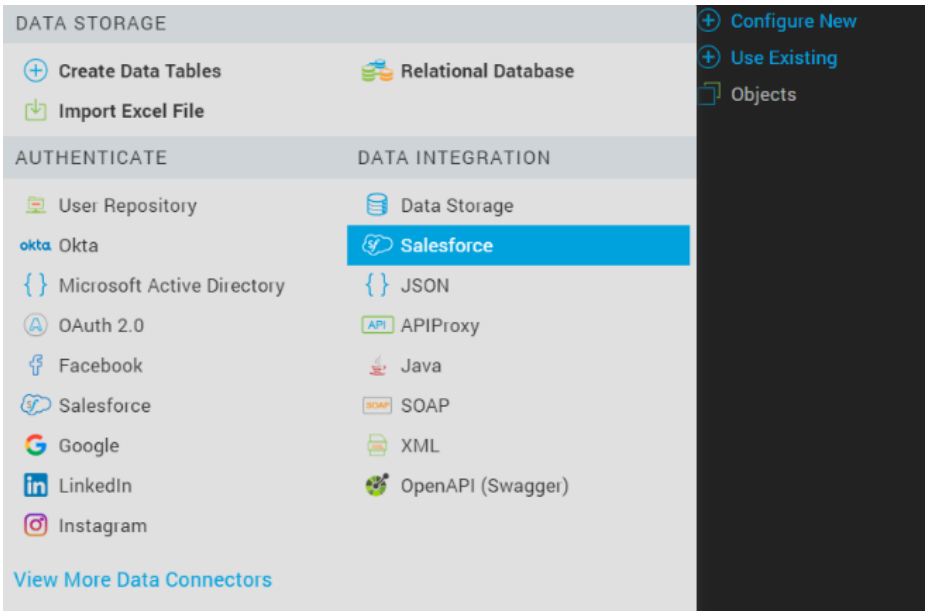
> ***Note:*** The reference of the device Browser widget is mandatory for Google OAuth 2.0 services. For this service, the deeplink URL is auto-generated by Kony Visualizer.

Create an Integration Service

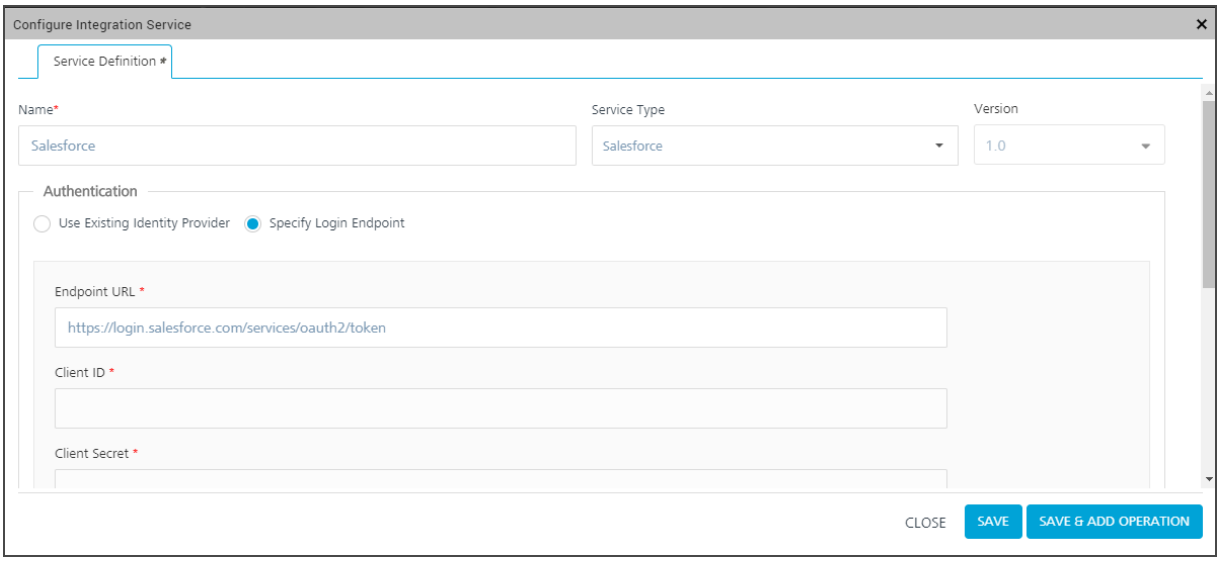
By default, you can view Integration services in the Data & Services panel. You can also set up new data sources. Once you configure the Integration service, you can view the request and response parameters.

To create a new Integration service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, expand **Project Services**.
3. Click **Configure New** to configure a new Integration service. A list of services that you can configure appears.
4. Under Get Data, select an Integration service; in this case, **Salesforce**.



5. The **Configure Integration Service** window appears, by using which you need to configure the **Salesforce** service. Specify the details in the required fields, and then click **Save & Add Operation**. For more information on how to create an Integration service, click here. In this case, the Salesforce Integration service.



6. Once the service is created, it is displayed in the Project Services list. You can then directly drag and drop the service onto a form and start using it.

Create an Object Service

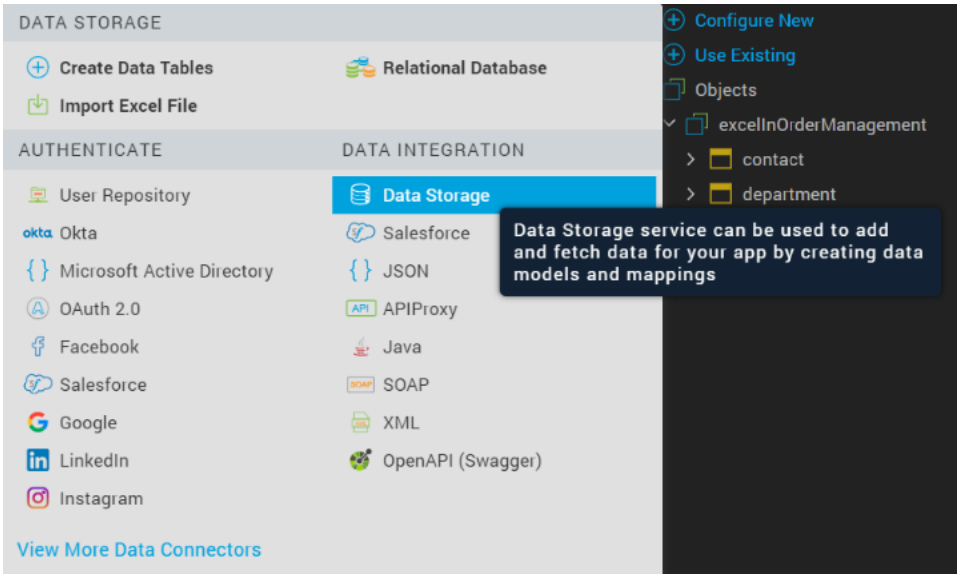
You can view Object services in the Data & Services panel. You can also set up new data sources. Once you configure an Object service, you can view the request and response parameters.

To create a new Object service, follow these steps:

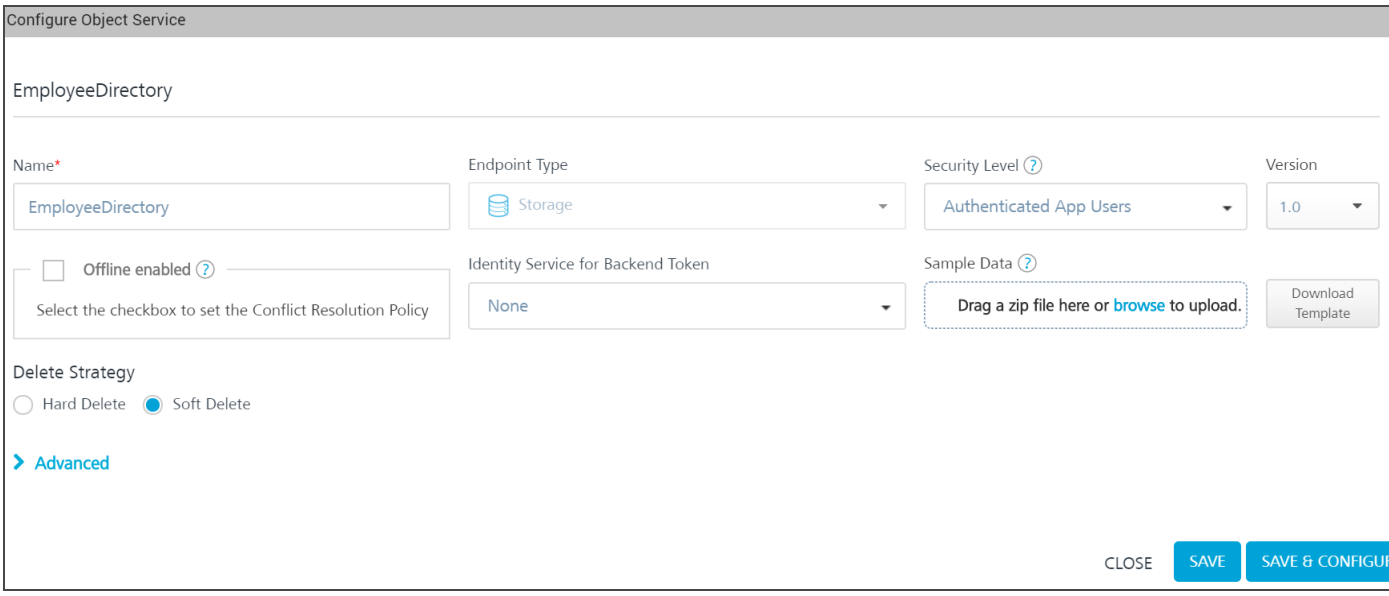
1. In Kony Visualizer, open your project.
2. From the **Data & Services** panel, expand **Project Services**.
3. Click **Configure New**. A list of services that you can add appears.

4. Under the Data Integration section, select an Object service. Here, **Data Storage** is the Object service.

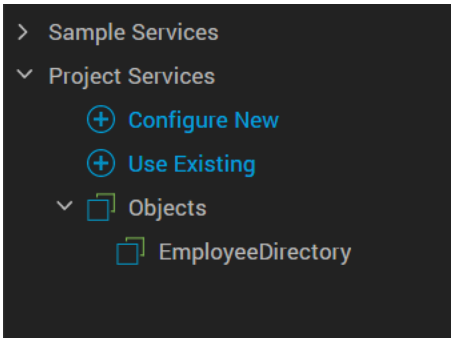
> ***Note:*** From Kony Visualizer V9, the Data Storage Object service is available under the **Data Integration** section in the list of services.



5. The **Configure Object Service** window appears, with **Endpoint Type** selected as **Storage** by default. You must type the **Name** of the Data Storage Object service and also specify other details in the necessary fields. Then either click **Save & Configure** to configure the object data model or click **Save** to just save the new service. For more information on how to create an Object service, click here.



6. After the service is created, it is displayed under **Project Services** > **Objects**. You can then directly drag and drop the service onto a form and start using it.

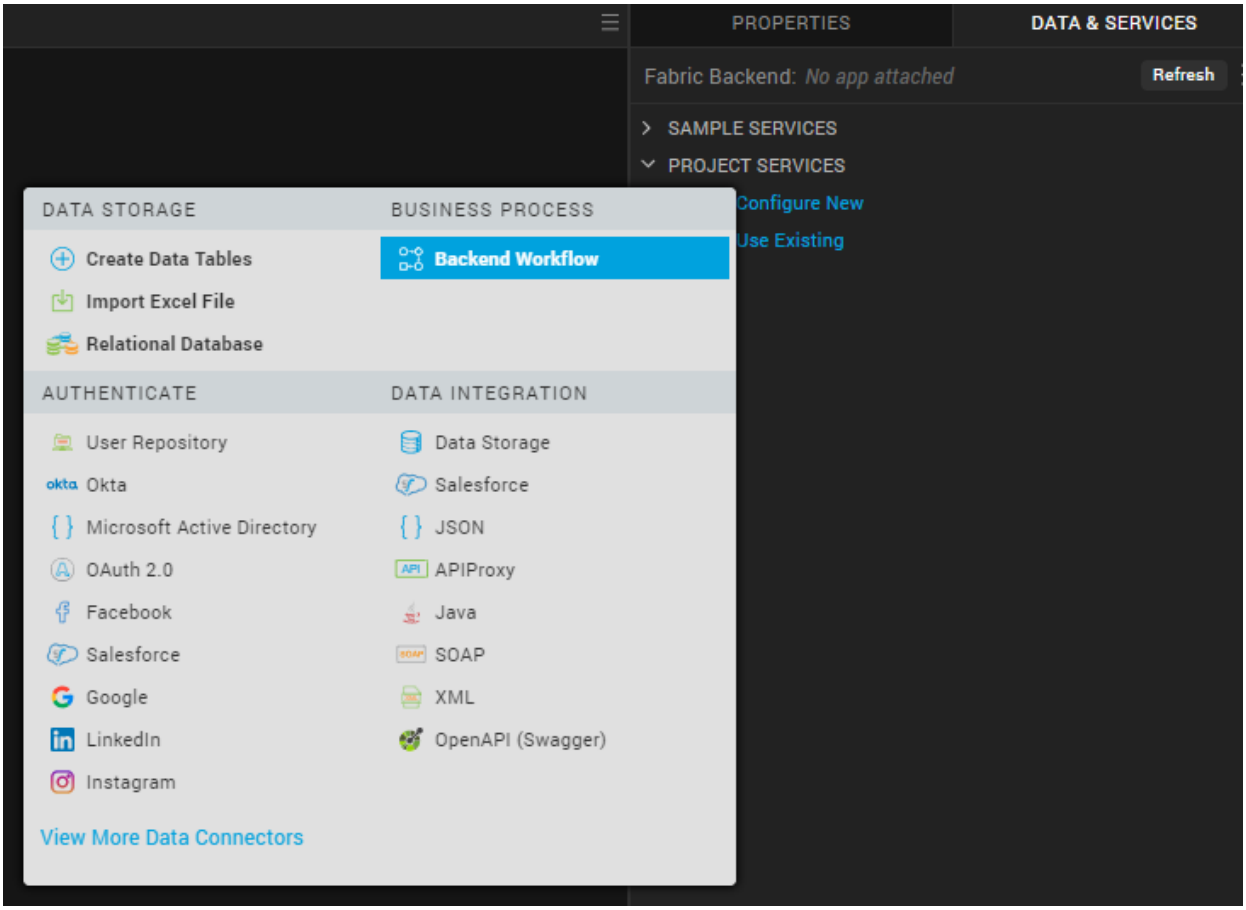


Create a Backend Workflow Service

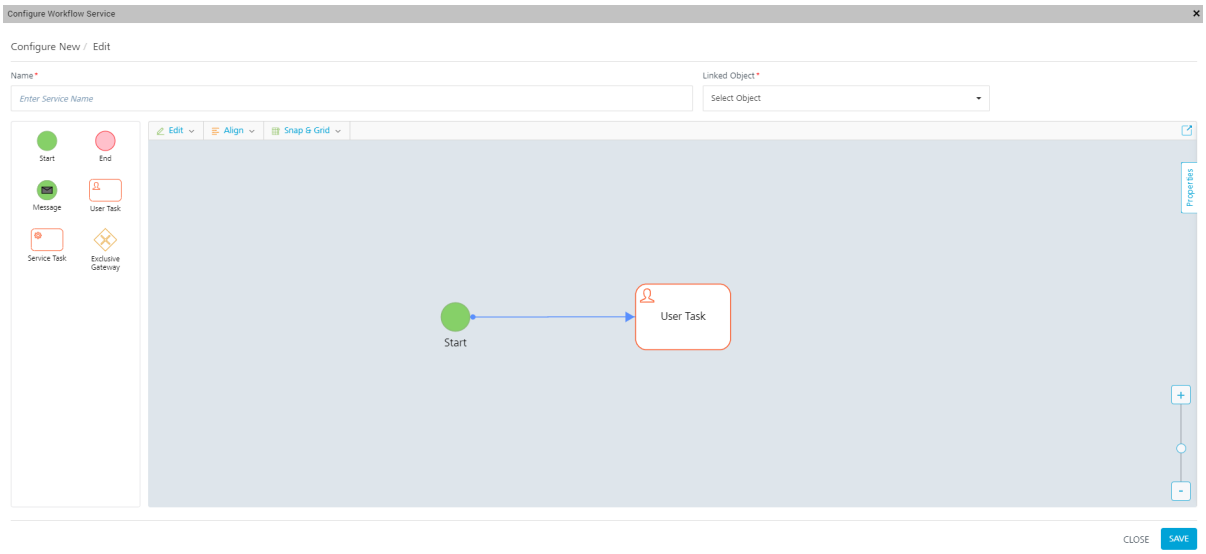
You can view Backend Workflow services in the Data & Services panel. Using Workflow, you can visualize and design a backend process by simply dragging and dropping different types of nodes and connecting them as per the logic you need.

To create a new Backend Workflow service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, expand **Project Services**.
3. Click **Configure New** to configure a new Backend Workflow.
   A list of services that you can configure appears.
4. Under **Business Process**, select **Backend Workflow**.



The **Configure Workflow Service** window appears.



5. In the **Name** field, type a unique name for the Backend Workflow service.
6. From the **Linked Object** list, select the object to which you want to link the new Workflow service. You can either select an existing Object service or Create a new Object service.
7. From the **Nodes** pane, drag and drop the required nodes onto the canvas to create a process. For more information on how to create a Backend Workflow service, click here.
8. Click **Save**. The service is created and appears in the Project Services list. The Workflow service invokes when you drag and drop the associated Object service onto a form and start using it.

Use Project Services

This section provides information on how to discover and use a Project service from the Data & Services panel.

Use an Identity Project Service

To use an Identity Project service, do the following:

1. In Kony Visualizer, open your project.
2. Create a form; for example, frmSample.
3. On frmSample, add button widgets to which you want to add the service.
4. From the **Data & Services panel**, expand **Project Services**.
5. Click **Use Existing**.
   A list of available services appears.
6. Click **Identity Service**.
   The **Select Identity Service** dialog box appears.
7. Select the required service from the list of available services, and then click **Add**.
   The service appears in the Data & Services panel.
8. Expand the service that you want to use and view the requests/responses of the service.
9. Drag and drop appropriate services on to the buttons. The Identity service is associated with the form.

Use an Integration Project Service

To use an Integration Project service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, expand **Project Services**.
3. Click **Use Existing**.
   A list of available services appears.
4. Click **Integration Service**.
   The **Select Integration Service** dialog box appears.
5. Select the required service from the list of available services, and then click **Add**.
   The service appears in the Data & Services panel.

6. Expand the service that you want to use and view the requests/responses of the service.
7. Drag and drop the service onto your form. The service is associated with the form.

Use an Object Project Service

To use an Object Project service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, expand **Project Services**.
3. Click **Use Existing**.
   A list of available services appears.
4. Click **Object Service**.
   The **Select Object Service** dialog box appears.
5. Select the required service from the list of available services, and then click **Add**.
   The service appears in the Data & Services panel.
6. Expand the service that you want to use and view the requests/responses of the service method.
7. Drag and drop the service onto your form. The service is associated with the form.

Use a Backend Workflow

To use a Backend Workflow service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, expand **Project Services**.
3. Click **Use Existing**.
   A list of available services appears.

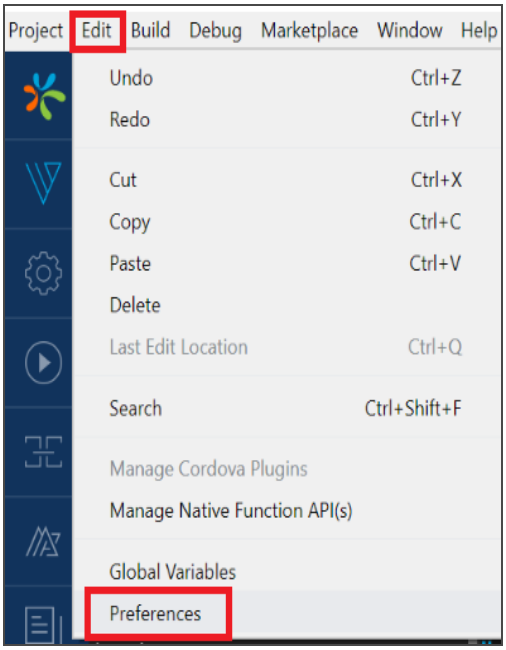
4. Click **Backend Workflow**.
   The **Select Workflow Service** dialog box appears.
5. Select the required service from the list of available services, and then click **Add**.
   The service appears in the Data & Services panel.
6. Expand the Object service that you associated with the Backend Workflow, and view the requests/responses of the service method.
7. Drag and drop the Object service onto your form. The service and the Workflow associated with the service are associated with the form.

Disable/Enable the Categorization of Project Services

From Kony Visualizer V9, the service categories are displayed under Project Services only when a service is linked from the Data & Services panel. In addition, an option to disable/enable the categorization of linked services under Project Services has been provided in the **Visualizer Preferences** window.

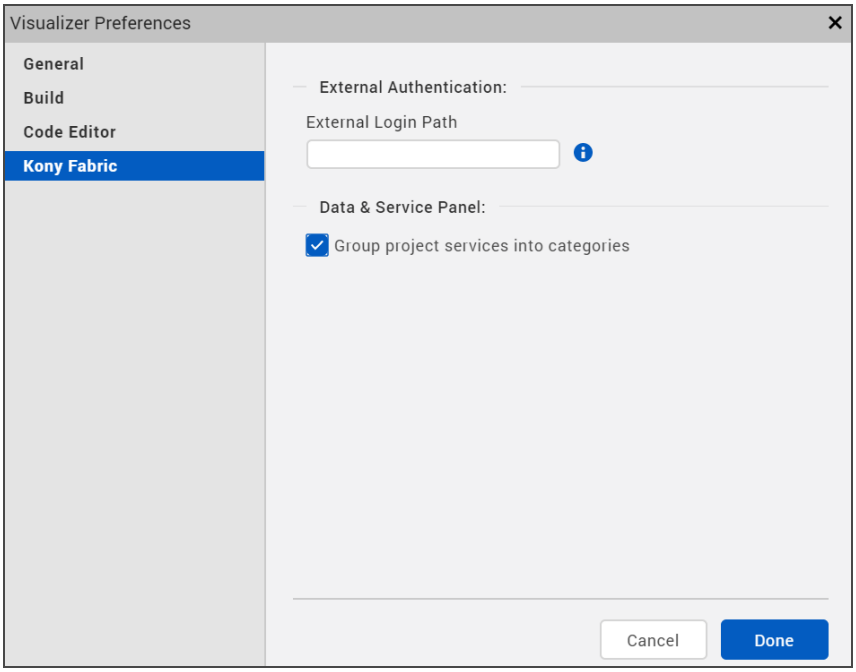
To disable/enable the categorization of linked services under Project Services, follow these steps:

1. In Kony Visualizer, click **Edit** > **Preferences**. The **Visualizer Preferences** window appears.

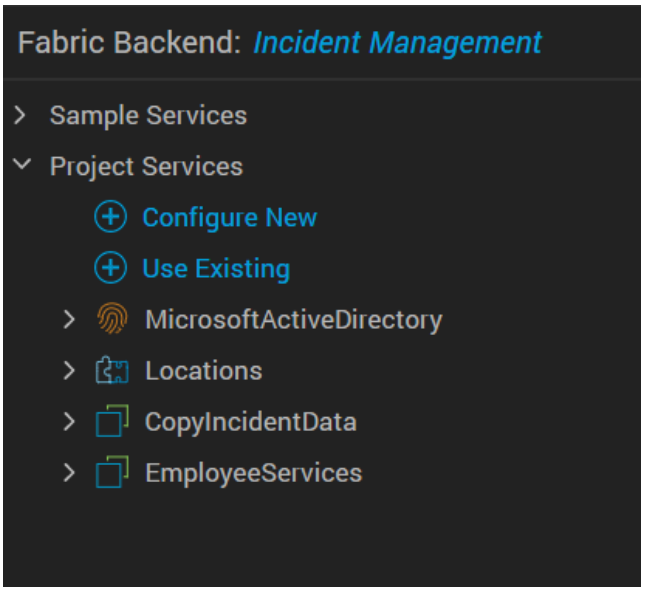


2. Click the **Kony Fabric** tab.

3. Under the **Data & Services Panel** section, the **Group project services into categories** check box is selected by default.

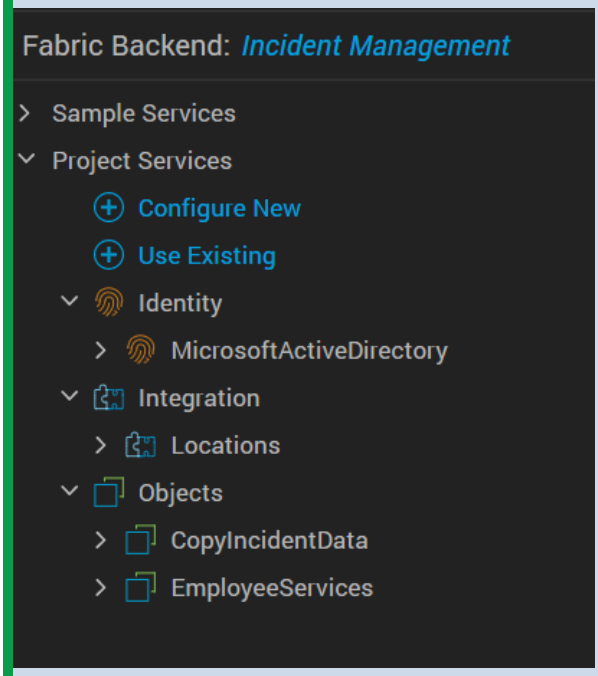


4. To disable the categorization of linked services under Project Services, clear the **Group project services into categories** check box.
5. Click **Done**. The linked services under Data & Services panel > Project Services do not appear in the categories of either Identity, Integration, or Object services.



Note: To enable the categorization of linked services under Project Services again, go to Edit > Preferences > Kony Fabric tab and select the **Group project services into categories** check box.

The linked services under Data & Services panel > Project Services appear in the categories of either Identity, Integration, or Object services respectively.



Unlink a Service

You can unlink a service from your project by right-clicking the service and selecting Unlink.

To unlink a service, follow these steps:

1. In Kony Visualizer, open your project.
2. From the **Data & Services** panel, expand **Project Services**. The list of available services appears.
3. Right-click the service that you want to unlink. A list of options appears.
4. Select **Unlink**. The **Confirm Service Unlinking** window appears. This window provides details on the operations and their respective action sequences that will be unlinked for that service,

and asks for your confirmation whether you want to unlink the service.

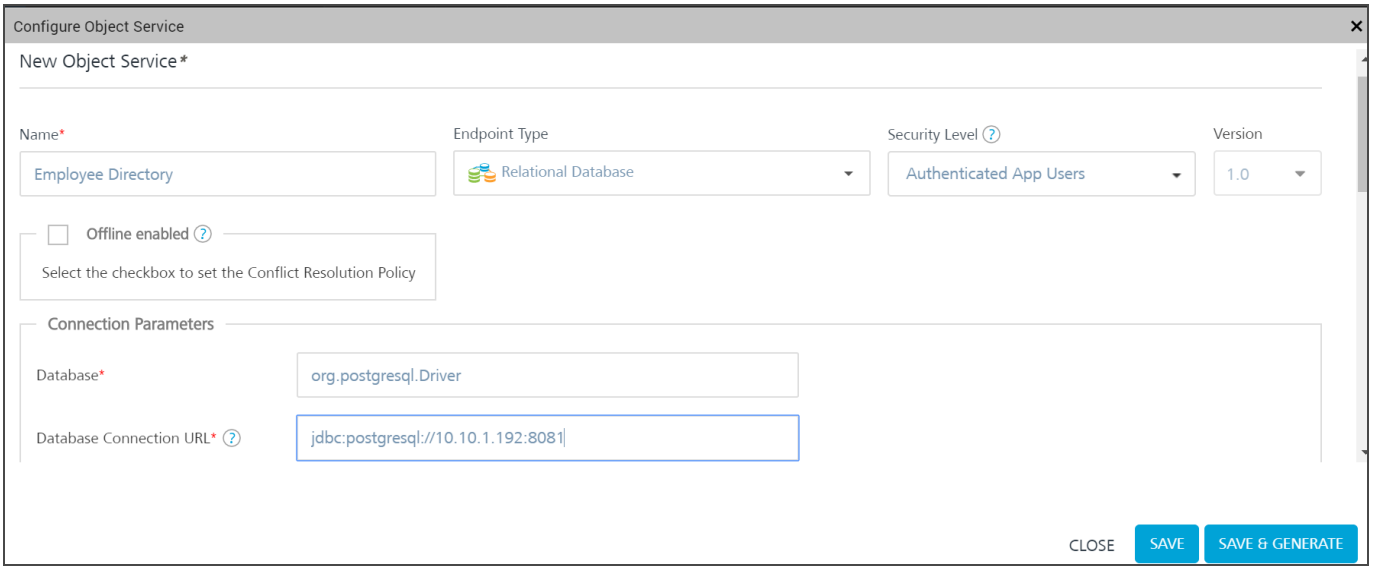
5. Click **Ok**. The service is removed from the Project Services list.

Edit a Service

You can edit a service that is added to your project by right-clicking the service and selecting Edit.

To edit a service, do the following:

1. In Kony Visualizer, open your project.
2. From the **Data & Services panel**, expand **Project Services**. The list of available services appears.
3. Right-click the service that you want to edit. A list of options appears.
4. Select **Edit**. The service opens in Kony Fabric. Here, while editing an Object service, the **Configure Object Service** window appears.



5. Make the required changes, and then click **Save & Add Operation**. The selected service is updated with your changes.

Object Service-related Features

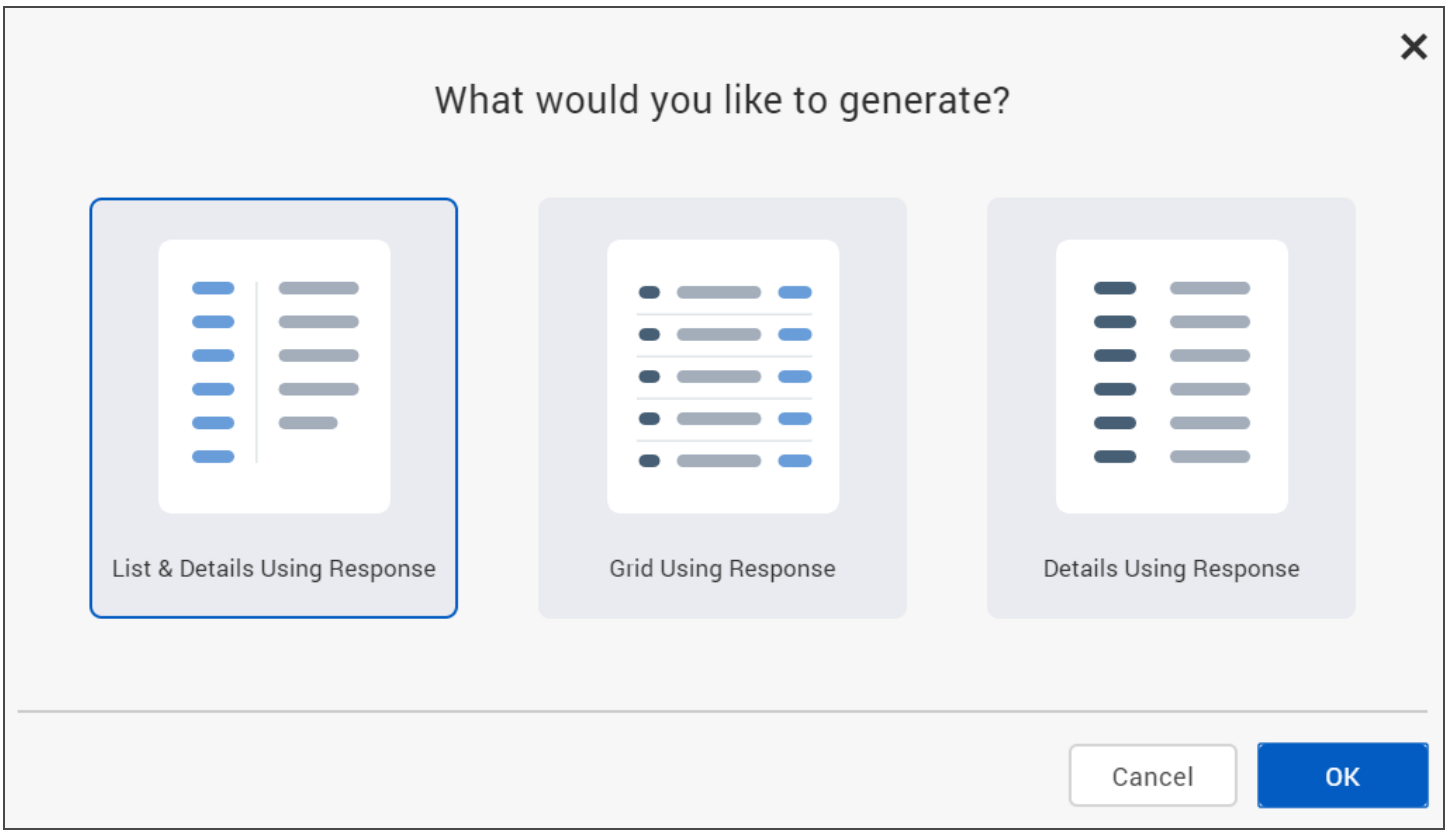
You can use the Data & Services panel to perform the following actions related to Object Services:

- Generate Object Services UI for Responsive Web
- Generate CRUD Forms for an Object Service
- Create a Data Table for an Object Service
- Configure an Object Data Model by Importing an Excel File
- Customize the Generation of Data Model Objects

Generate Object Services UI for Responsive Web

From Kony Visualizer V9, when you drag and drop service operations from the Data & Services panel onto Responsive Web forms, the following types of forms can be generated:

- List and Details Using Response (for Get operations)
- Grid Using Response (for Get operations)
- Details Using Response (for Get operations)
- Entry (for Create and Update operations)



To understand this feature in detail, let us consider a scenario where Steve (a low-code developer) wants to create an app called **Incident Management**. He has already created the associated Kony Fabric app, and now wants to configure the app's UI.

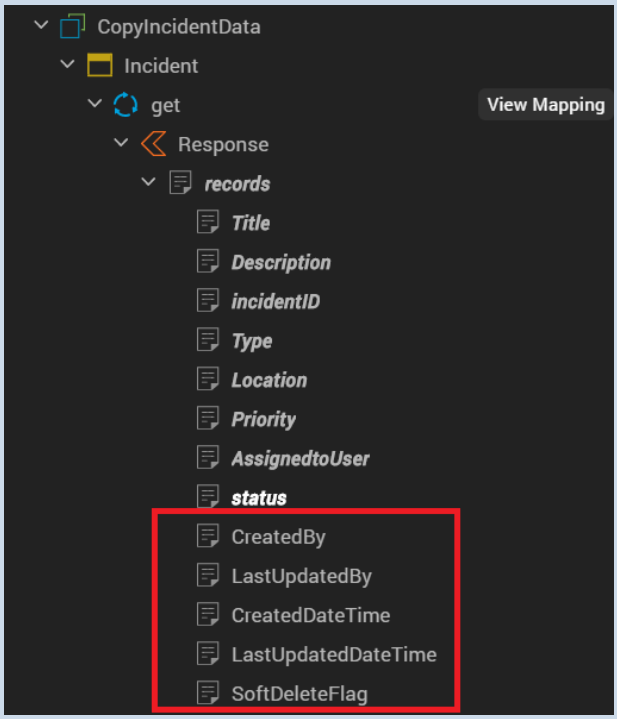
The Admin of an airport can use the Incident Management Responsive Web app to view, create, update, and assign incidents/tasks to relevant staff members. In addition, staff members can use the Incident Management Mobile app to view and act upon their assigned tasks as well as to create new tasks.

The Responsive Web and Mobile apps comprise of four forms: list, detail, update, and create.

> ***Note:*** When you drag service operations from the Data & Services panel and drop onto Responsive Web forms, auto-generated fields such as *CreatedBy*, *LastUpdatedBy*, *CreatedDateTime*, *LastUpdatedDateTime*, and *SoftDeleteFlag* are not added to the forms by default. To add any of these fields to a Responsive Web form, drag the required operation parameter (for example, *CreatedDateTime*) from the Data & Services panel and drop onto the

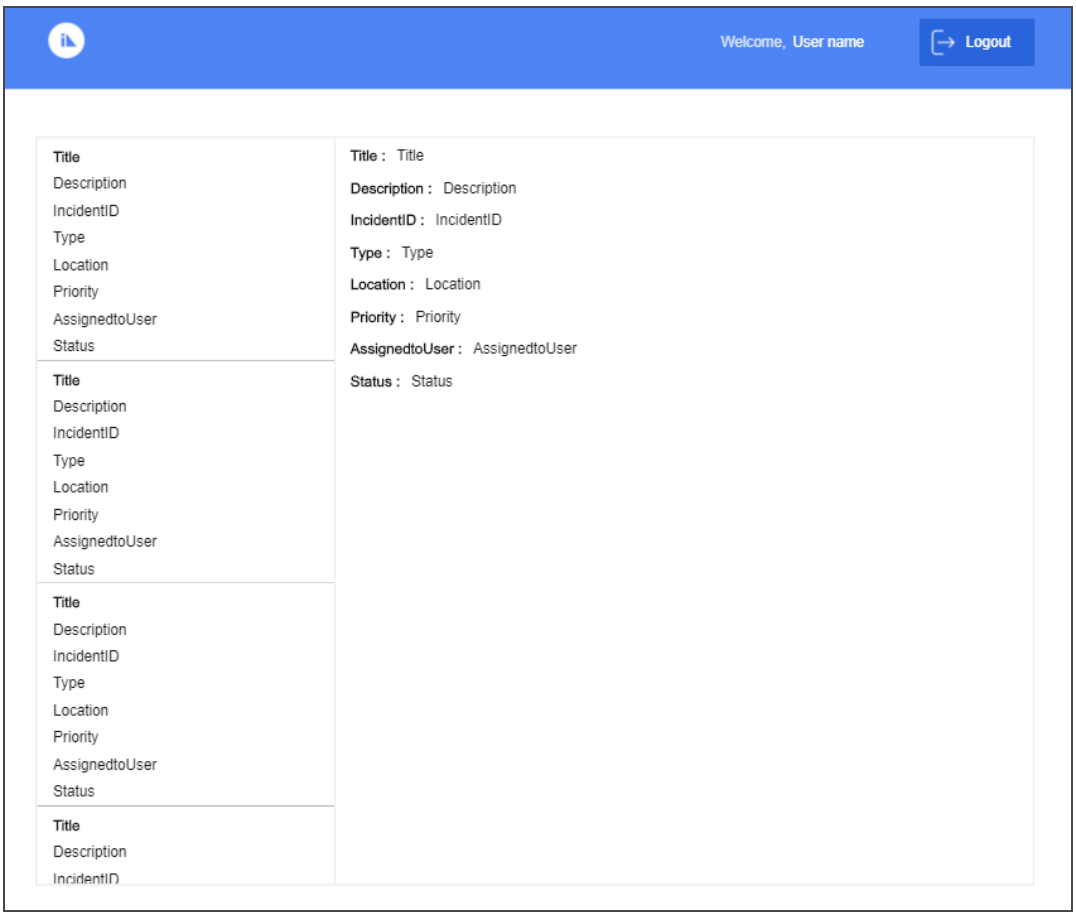
form. This feature is only applicable for Storage services.

In case of other types of services, the Data & Services panel does not auto-generate the *SoftDeleteFlag* and other object tracking fields. You must create a custom field on Kony Fabric, and use the field for tracking purposes.



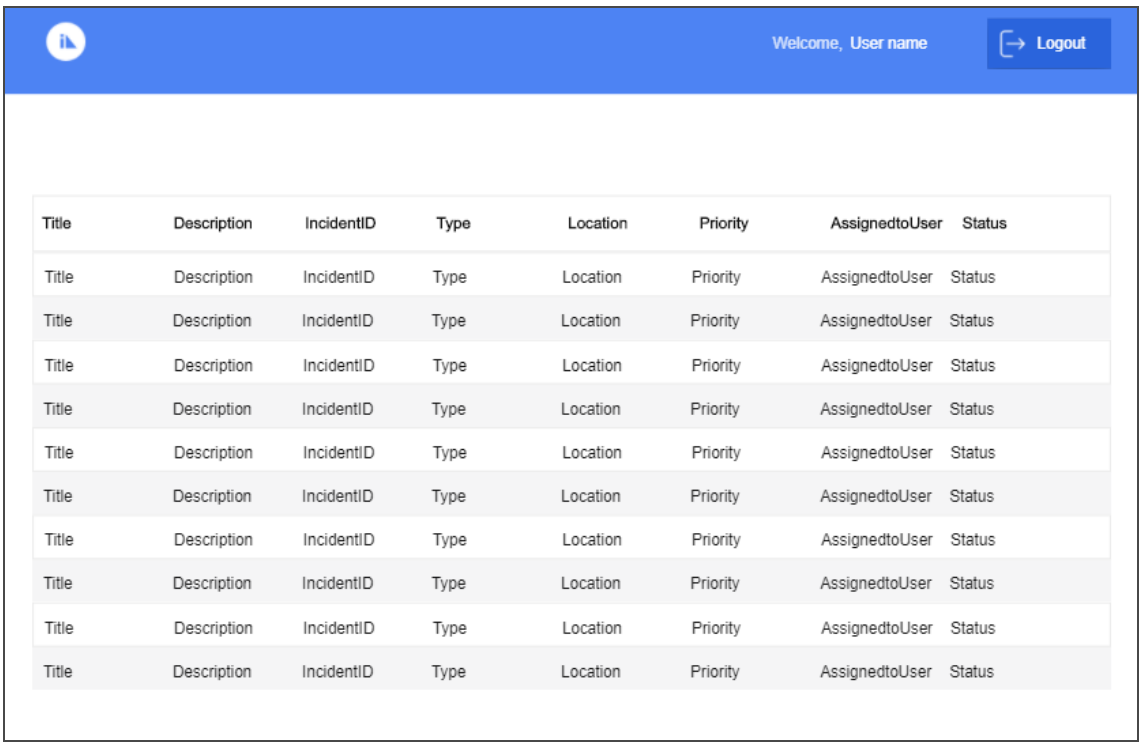
List and Details Using Response Form

When you drag and drop the Get operation in an Object service onto a Responsive Web form and select the **List and Details Using Response** option, this type of form is generated on the Project Canvas. A list and details form consists of a list of items and if a user selects a specific item, the details of that item are displayed on the right of the screen.



Grid Using Response Form

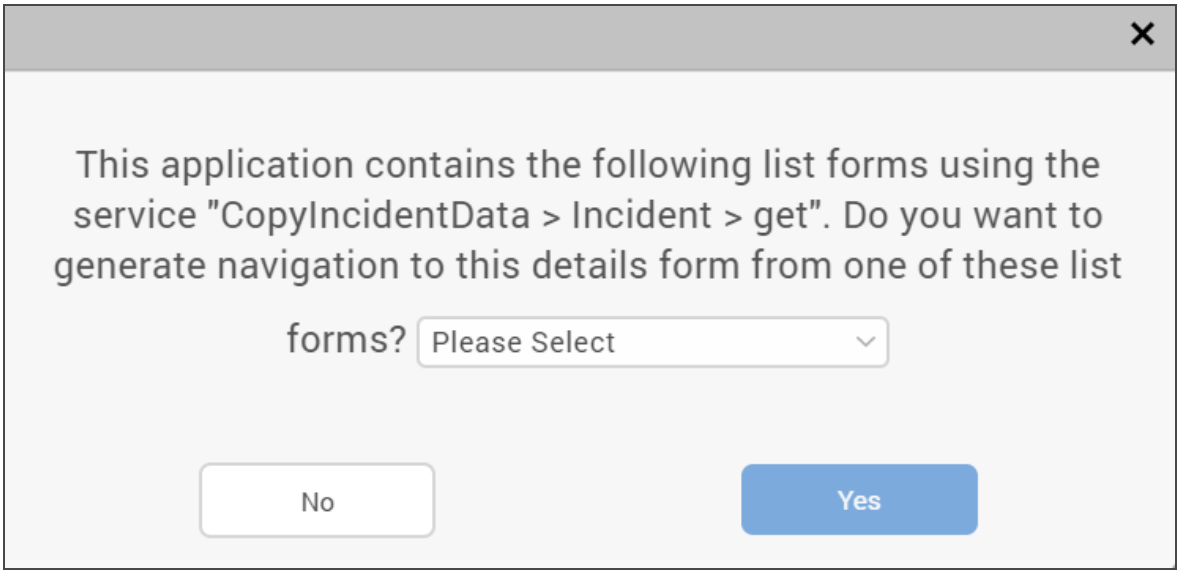
When you drag and drop the Get operation in an Object service onto a Responsive Web form and select the **Grid Using Response** option, this type of form is generated on the Project Canvas. A grid form is a tabular representation of list items. Users can select any row grid item, and they are navigated to the details screen of that item.

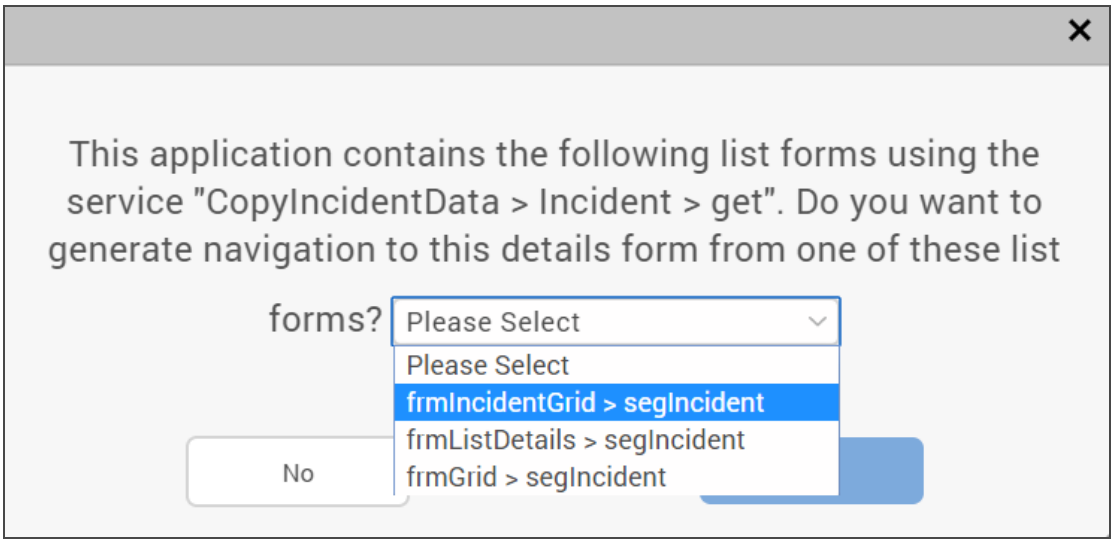


Details Using Response Form

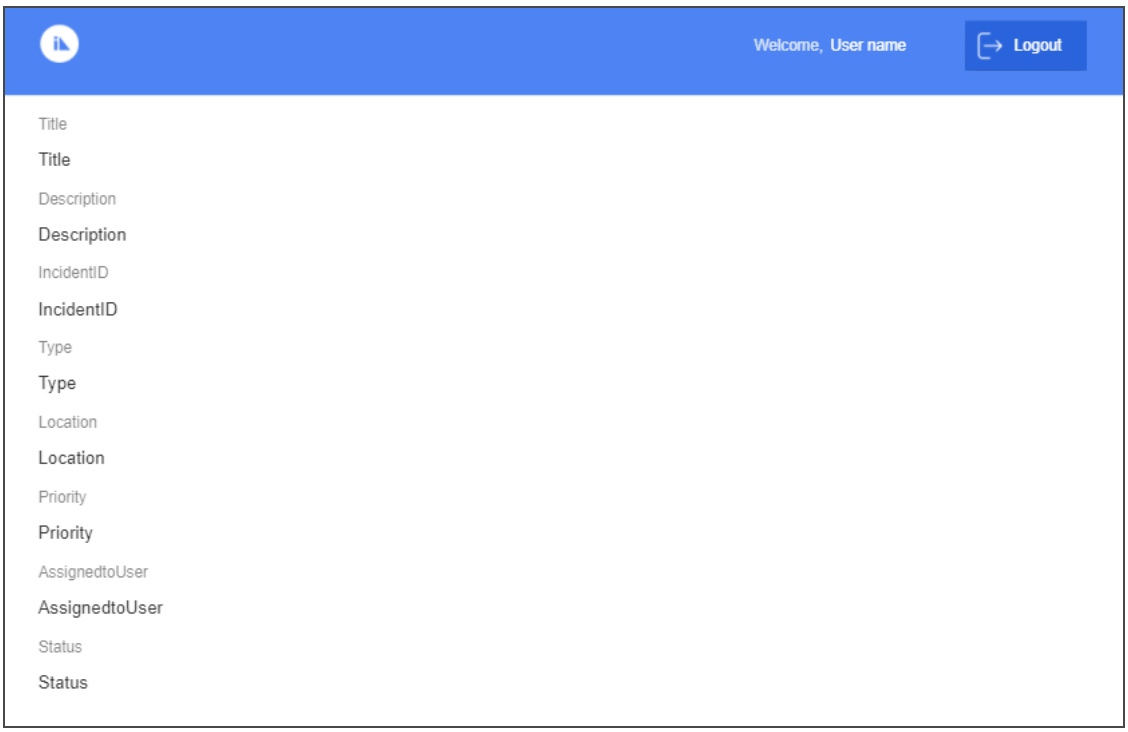
When you drag and drop the Get operation in an Object service onto a Responsive Web form and select the **Details Using Response** option, a dialog box appears. If a Responsive Web list form already exists, the dialog box asks if you want to generate navigation from the link to the details form. Click **Yes** to generate a navigation link between the list and detail forms; otherwise, click **No**.

> ***Note:*** If multiple Responsive Web list forms are present, then you will need to choose one list form to create the navigation link.





A details form is then generated on the Project Canvas. The details form contains additional information of a list item. If you create the navigation link between the list and detail forms, users are taken to the detail form when they select a specific list item.

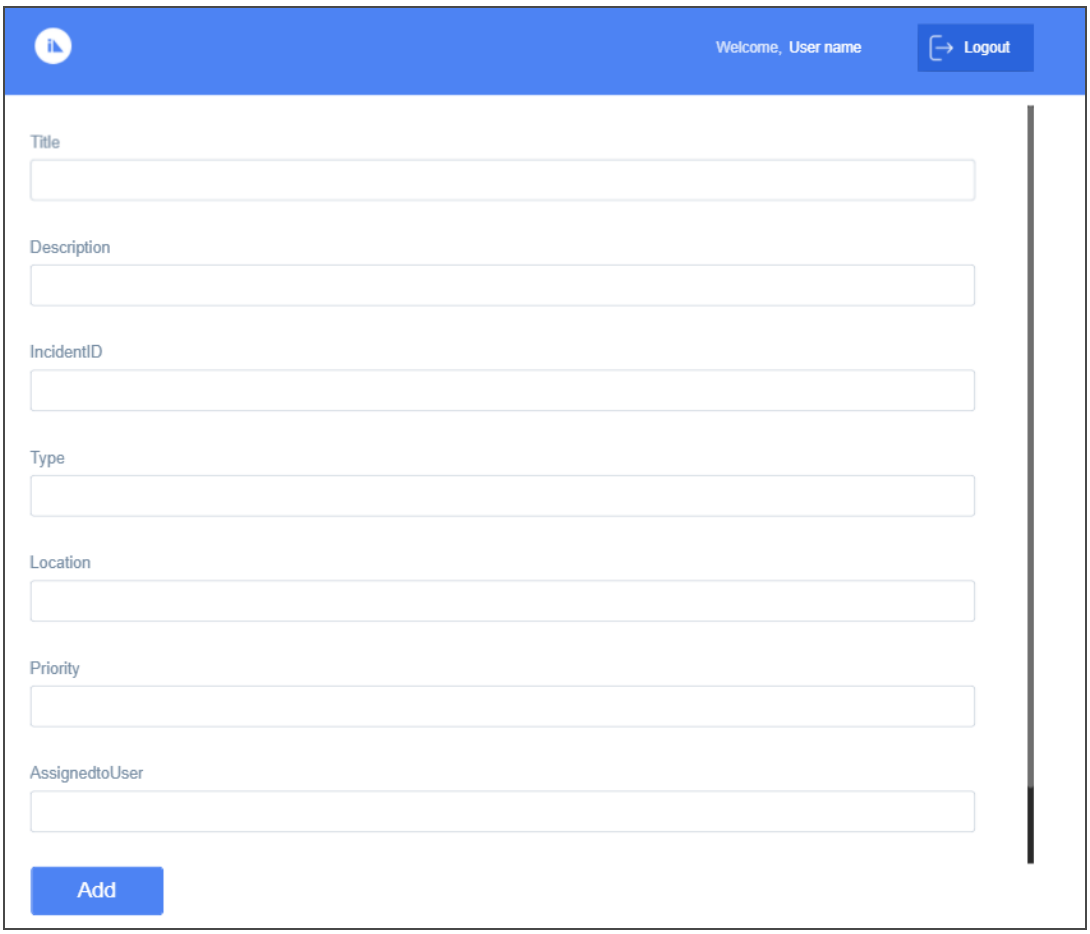


Entry Form

When you drag and drop either the Create or Update operation in an Object service onto a Responsive Web form, an Entry form is automatically generated on the Project Canvas.

Create Operation

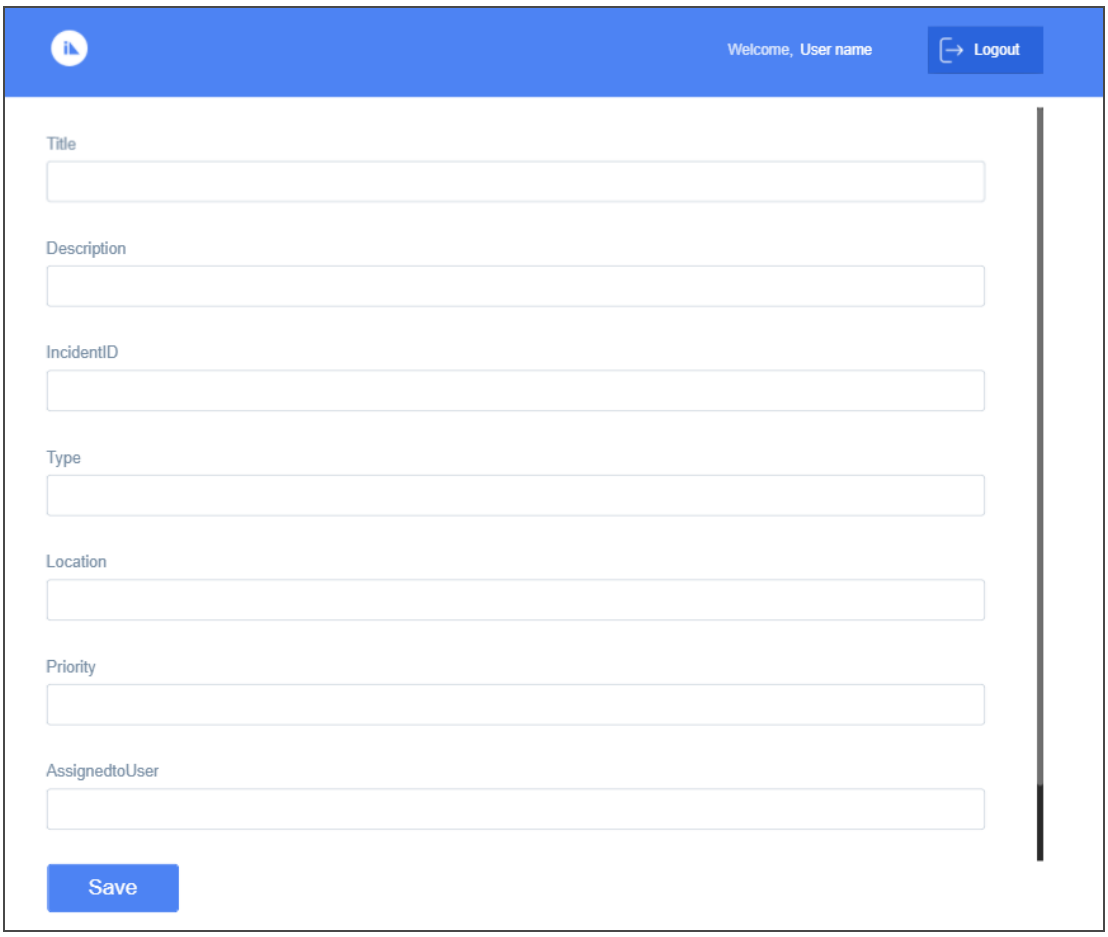
When an entry form is added for a Create operation, users can specify the required details to create new list items. The following image illustrates the Create Incident screen of the Incident Management app. App users can leverage this screen to create new incident items and assign them to relevant people.



Update Operation

When an entry form is added for an Update operation, users can make the required modifications to existing list items. The **update** method automatically fetches the values of the selected record and pre-fills the details in the respective fields.

The following image illustrates the Update Incident screen of the Incident Management app.



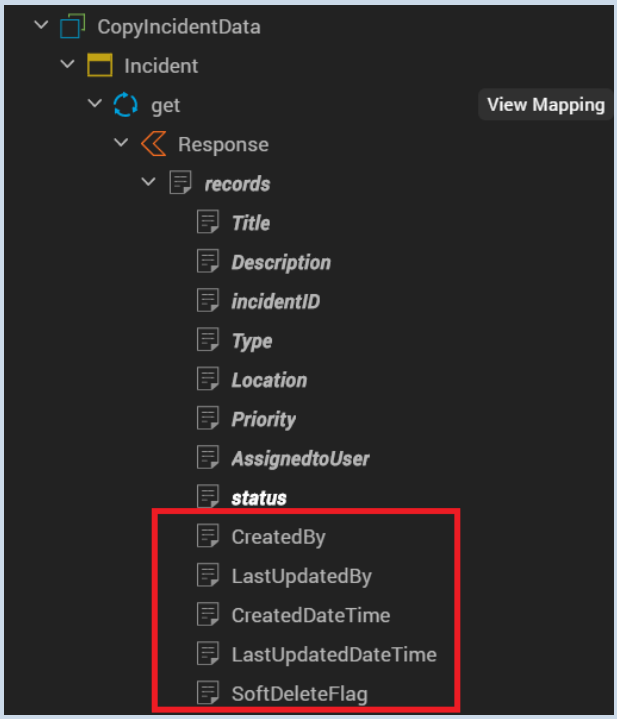
Generate CRUD Forms for an Object Service

From Kony Visualizer V9, you can generate CRUD forms and associated form-navigation links for an Object service. If you have configured the data model for an Object service or linked your Kony Visualizer project to a Kony Fabric app with a defined Object service data model, you can leverage this feature. This enhancement is applicable for Object services on Mobile, Tablet, and Responsive Web (Desktop) channels.

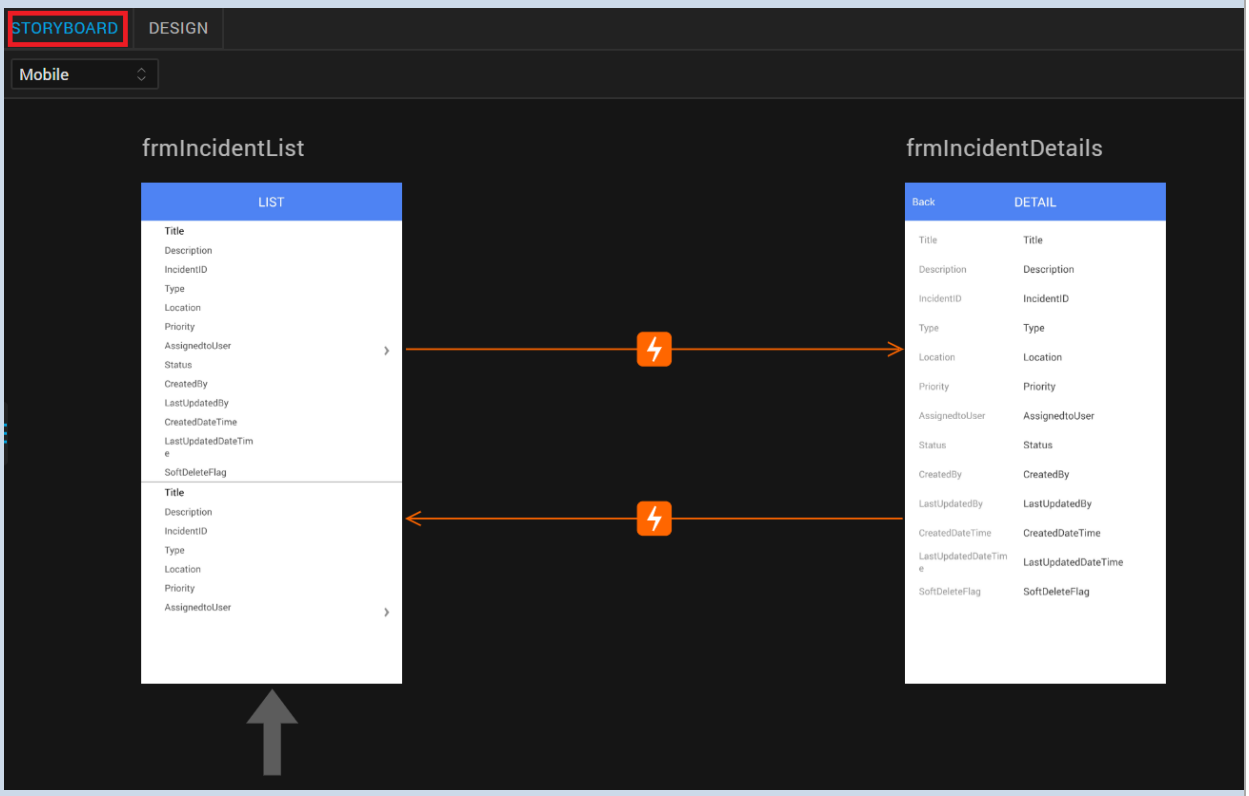
You can generate CRUD forms at method-level, object-level, and at service-level.

Note: When you use the generate forms feature, auto-generated fields such as *CreatedBy*, *LastUpdatedBy*, *CreatedDateTime*, *LastUpdatedDateTime*, and *SoftDeleteFlag* are not added to the CRUD forms by default. To add any of these fields to a form, drag the required operation parameter from the Data & Services Panel and drop onto the form. This feature is only applicable for Storage services.

In case of other types of services, the Data & Services panel does not auto-generate the *SoftDeleteFlag* and other object tracking fields. You must create a custom field on Kony Fabric, and use the field for tracking purposes.



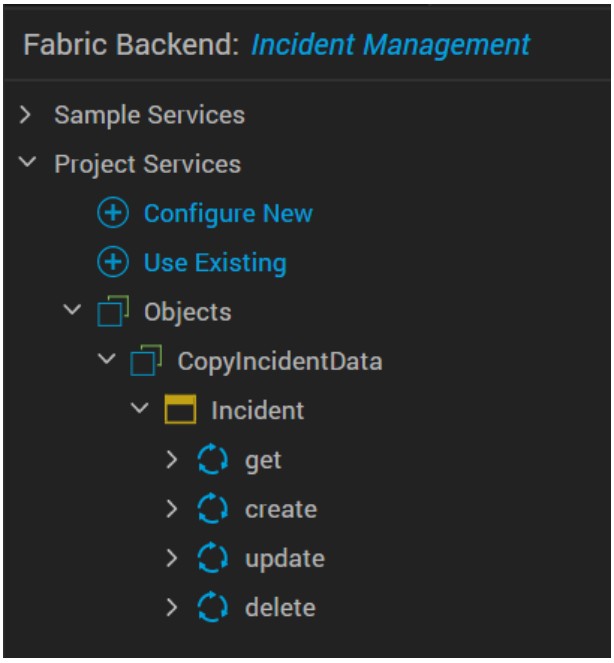
Note: When you drag and drop a method, object, or service from the Data & Services panel to anywhere on the Storyboard view canvas, the relevant forms and associated hard navigation links between the forms are automatically generated.



To generate CRUD forms for an Object service for Responsive Web (Desktop) channel, follow these steps:

1. In Kony Visualizer, configure the data model for a new or existing Object service. Alternatively, link your Kony Visualizer project to a Kony Fabric app with a defined Object service data model. In this procedure, we will discuss the example of the Object service of the Incident Management app.
2. Go to **Data & Services** > **Project Services**.

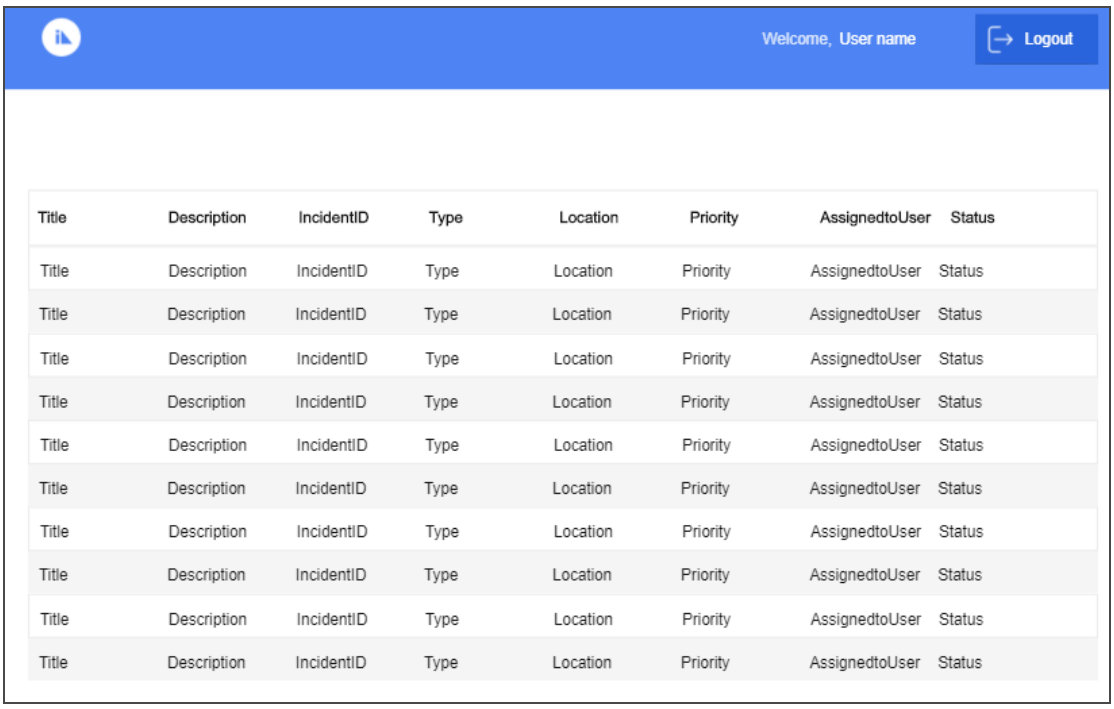
3. Expand **Objects** > **CopyIncidentData** > **Incident**.



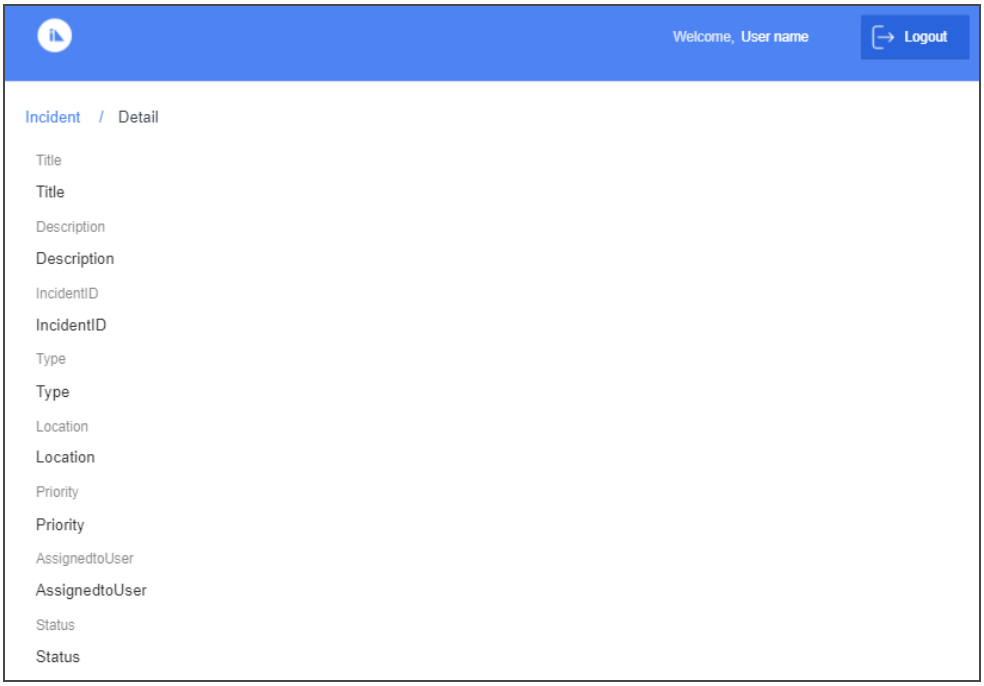
4. You can generate CRUD forms for any of the following items:

   - **Method-level**: Right-click any method; for example, **get**. Select **Generate Forms** > **Desktop**. Two types of forms (Grid and Details) are generated with the respective navigation links between each other. An appropriate breadcrumb link is also displayed at the top of the Details form. Users can click the breadcrumb link to go back to the Grid screen of the app.

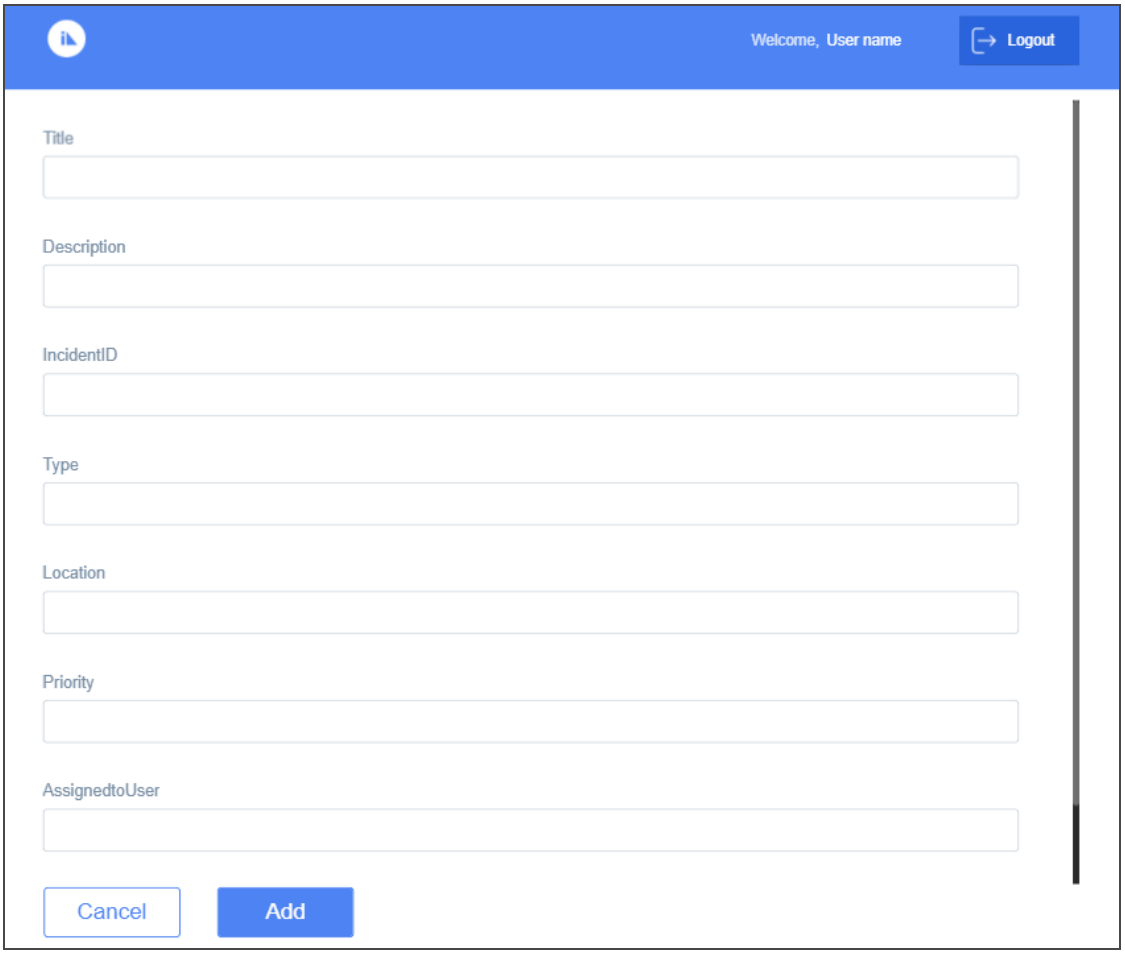
     Grid Form

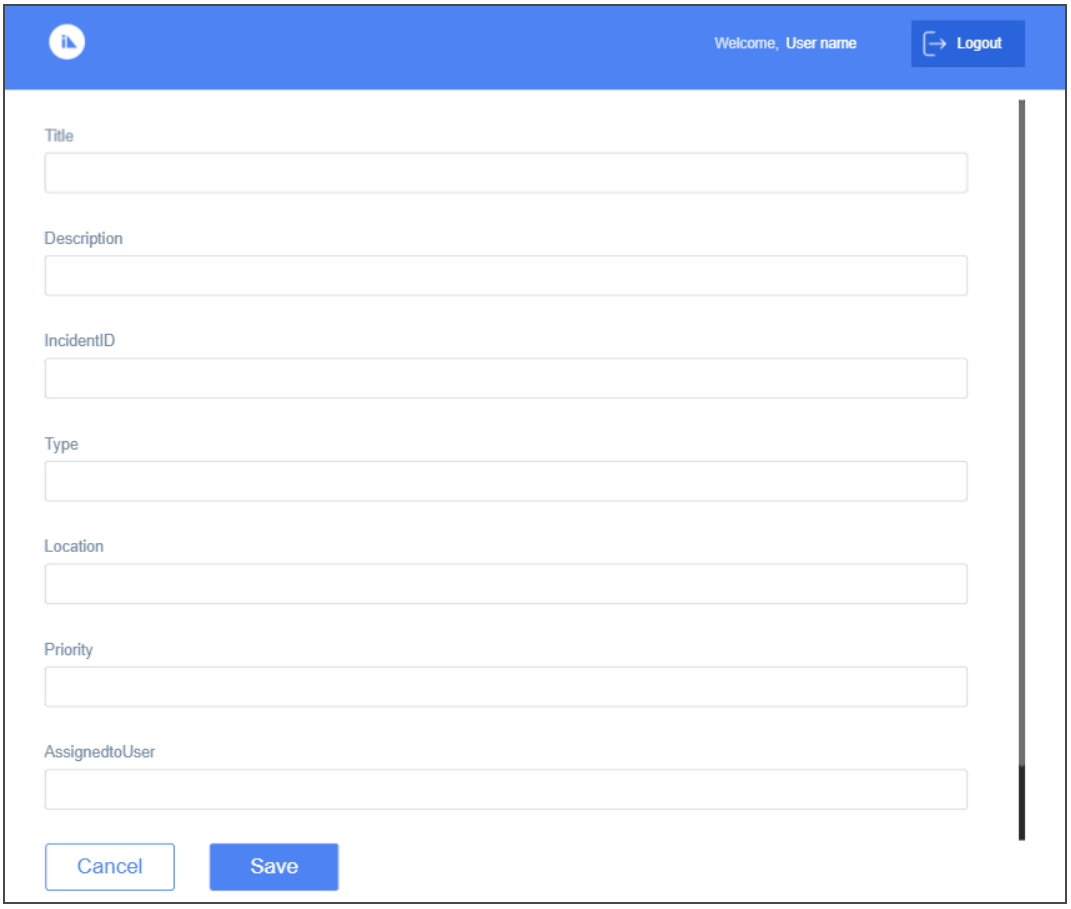


Details Form



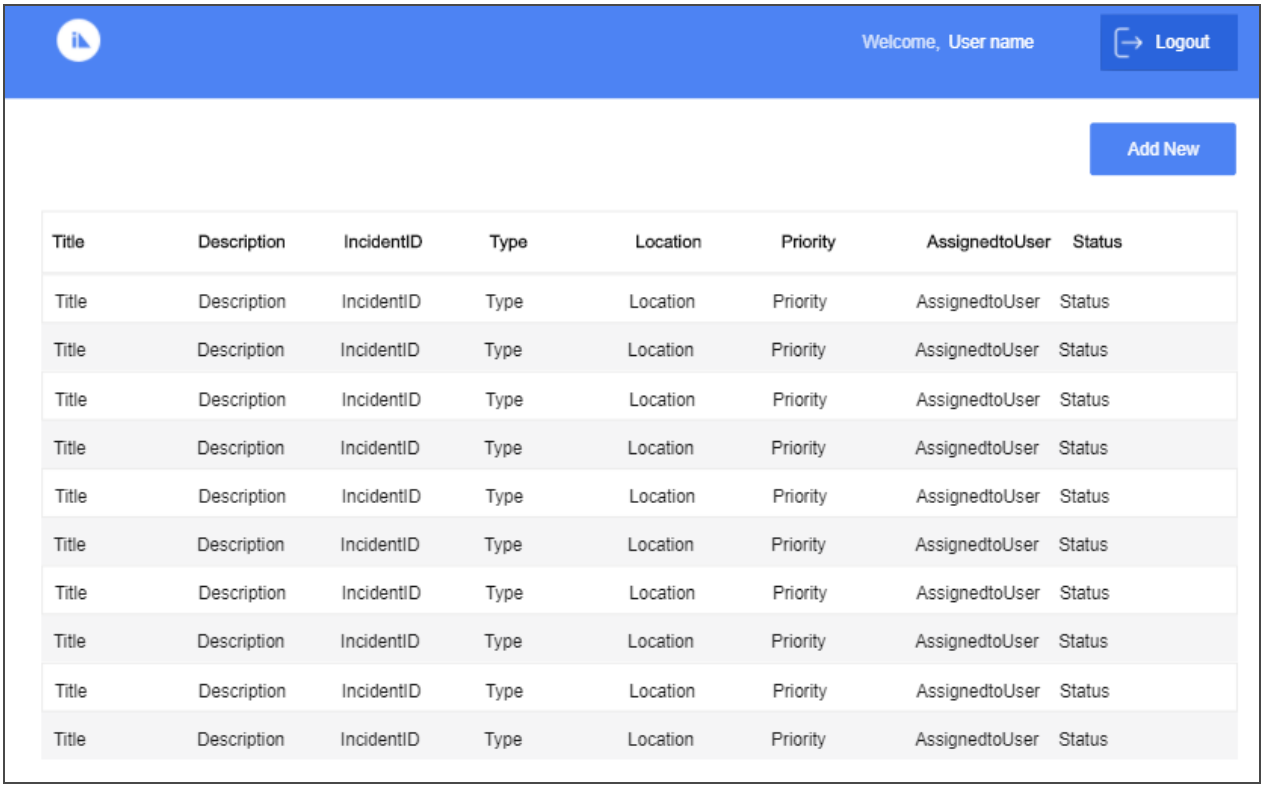
If you perform the **Generate Forms** action for the **create** or **update** methods, only the Create/Update entry form is generated.



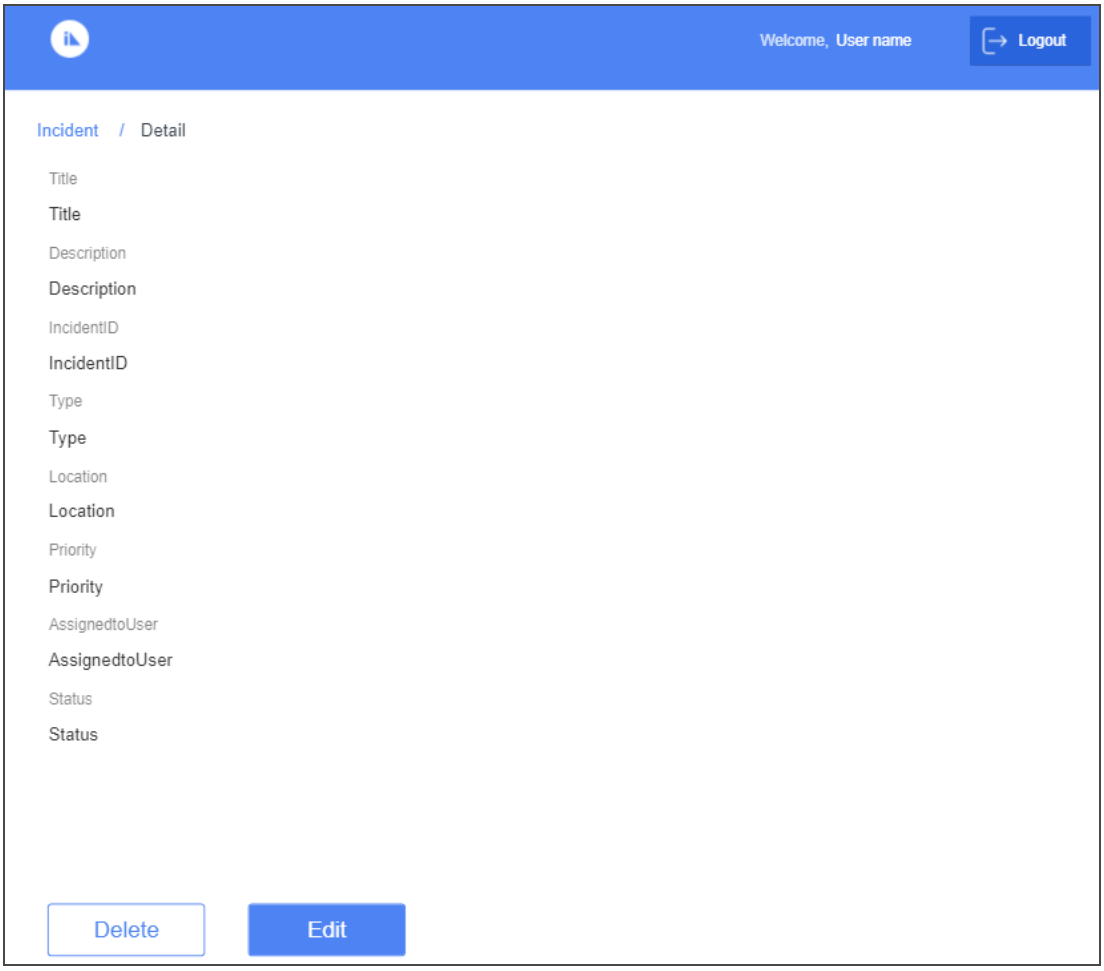


- **Object-level**: Right-click **Incident**, and then select **Generate Forms** > **Desktop**. Four types of forms (Grid, Details, Create entry form, and Update entry form) are generated with the respective navigation links among each other. An appropriate breadcrumb link is also displayed at the top of the Details, Create, and Update forms. Users can click the breadcrumb link to go back to the Grid screen of the app.

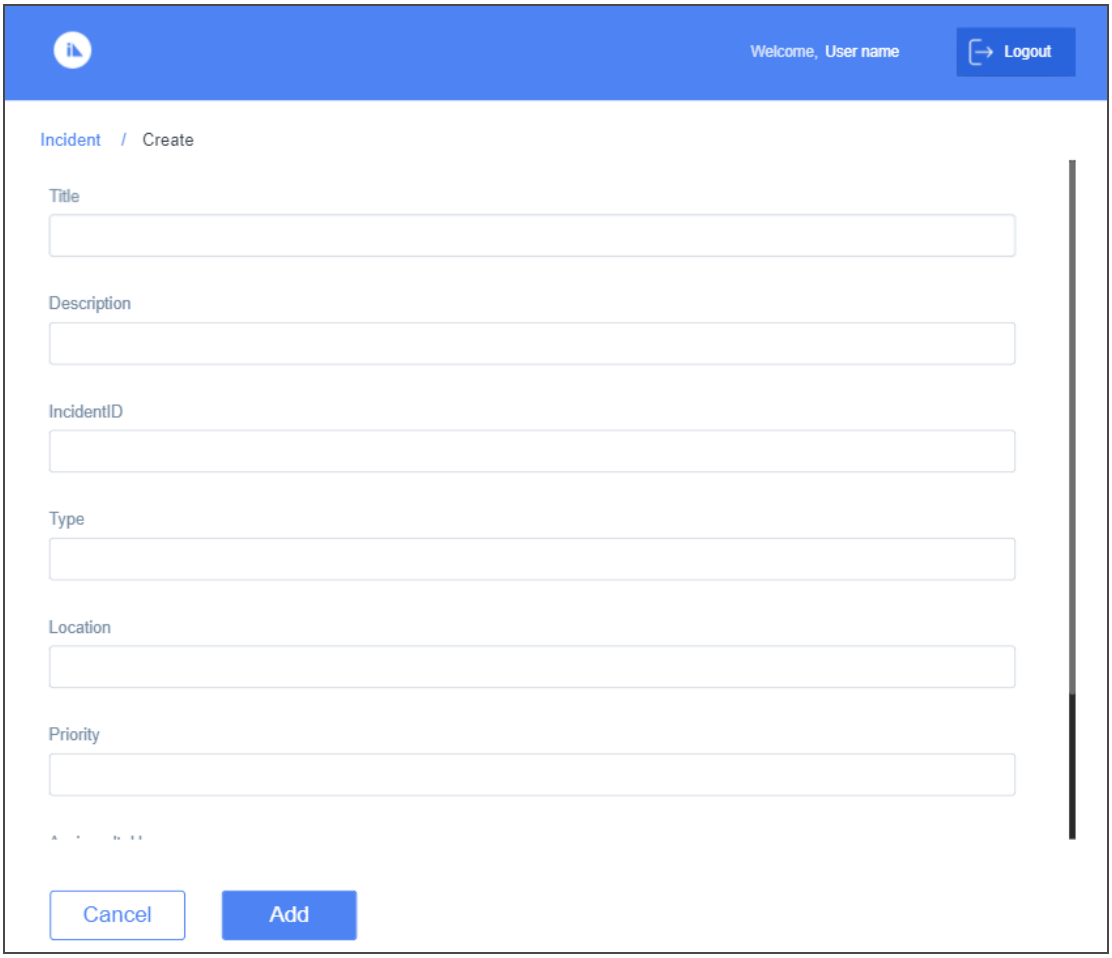
Grid Form



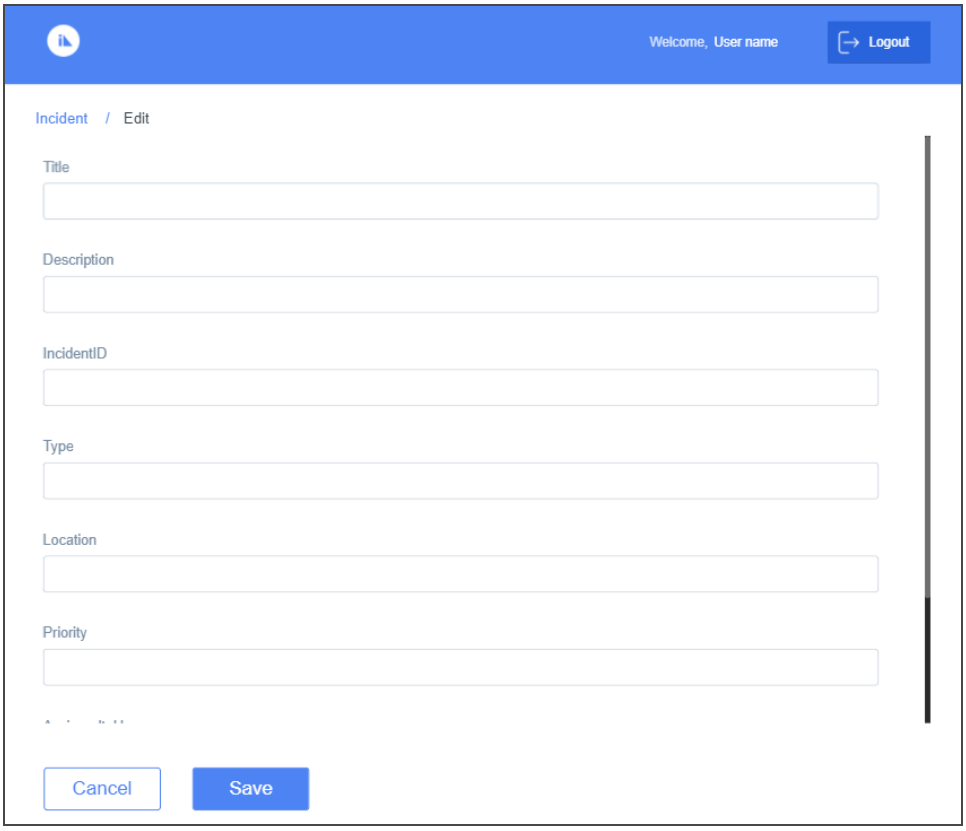
Details Form



Create Form



Update Form

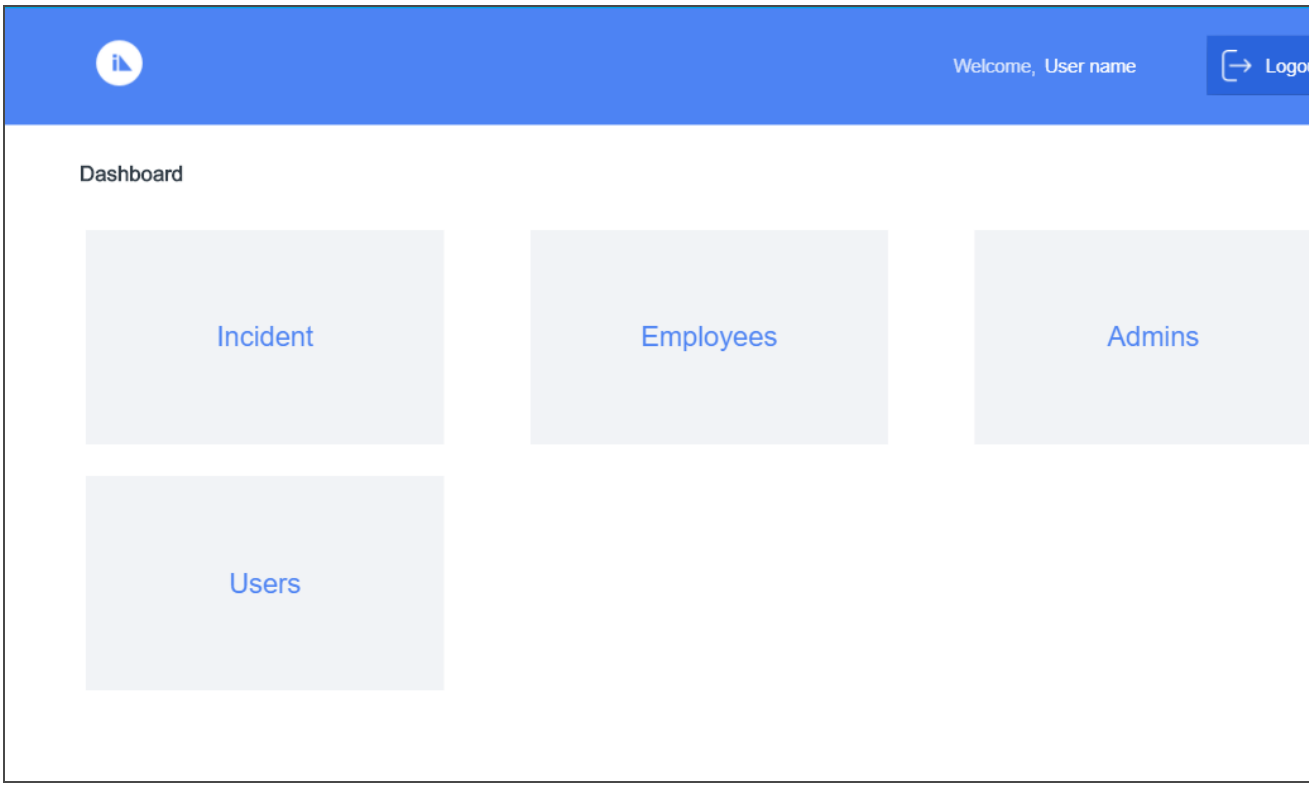


- **The Object service**: Right-click **CopyIncidentData**, and then select **Generate Forms** > **Desktop**. Five types of forms (Dashboard form, Grid, Details, Create entry form, and Update entry form) for each object in the Object service are generated, with the respective navigation links among each other.

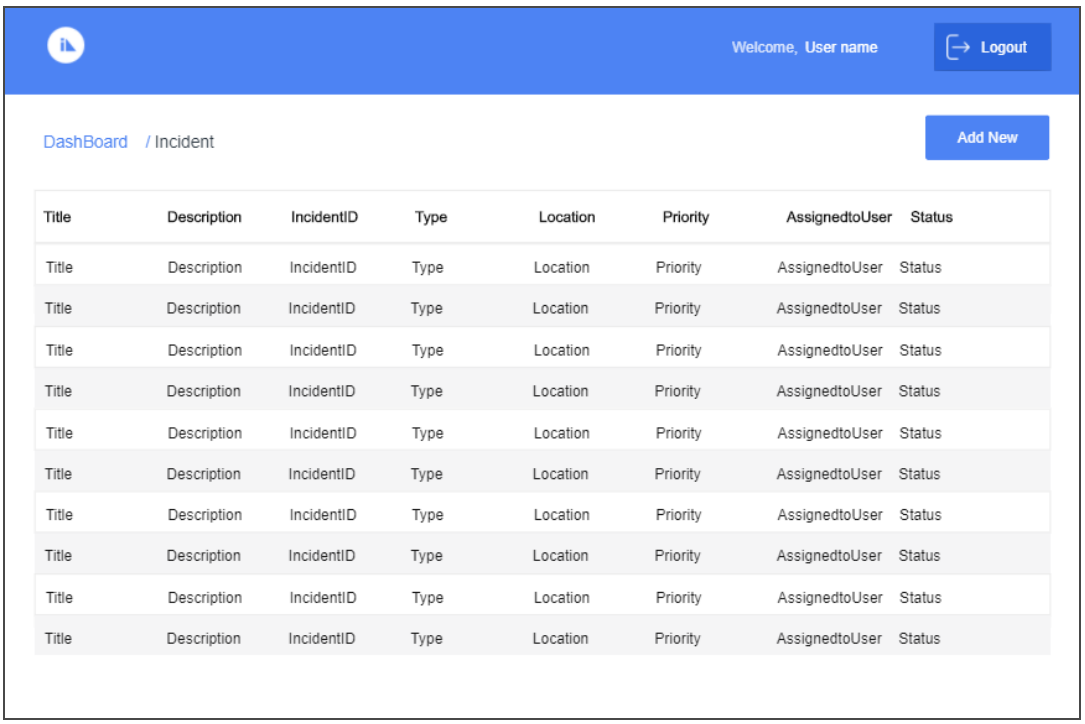
  The Dashboard form displays all the objects that are part of the selected Object service. From the Dashboard screen, app users can navigate to the respective Grid, Details, Create, and Update forms of the objects. If you have not defined the startup form of the app, the Dashboard form is made the startup form by default.

  An appropriate breadcrumb link is also displayed at the top of the Grid, Details, Create, and Update forms. Users can click the breadcrumb link to go back to either the Dashboard screen or Grid form of the app, as appropriate.

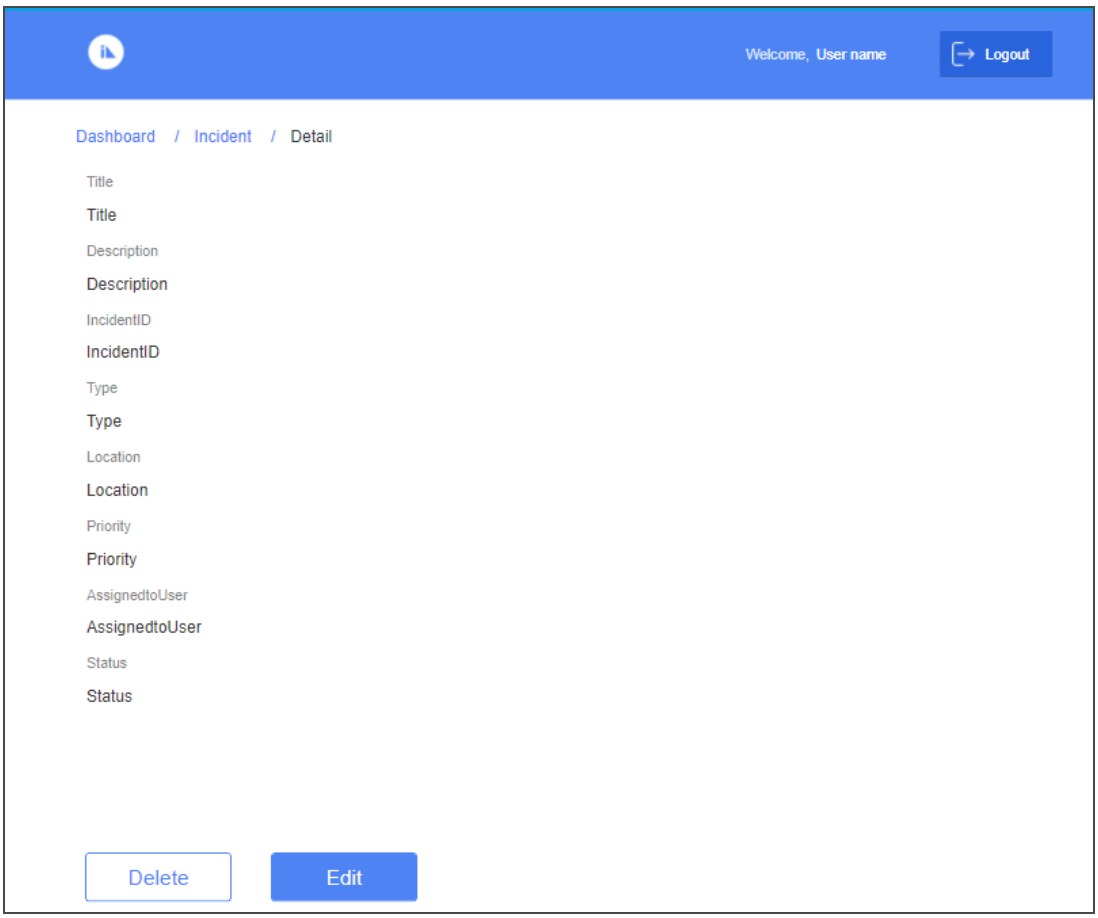
  Dashboard Form (with *Incident* object)



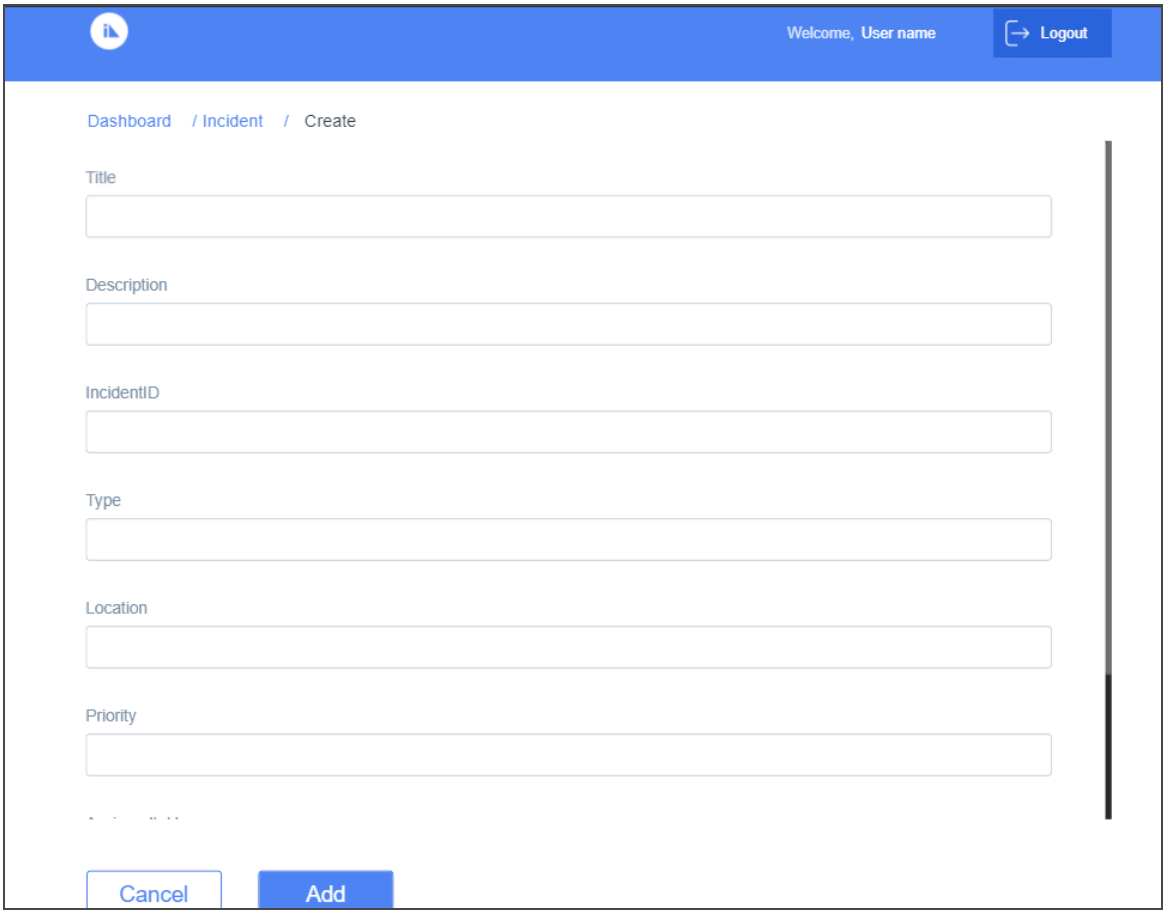
Grid Form



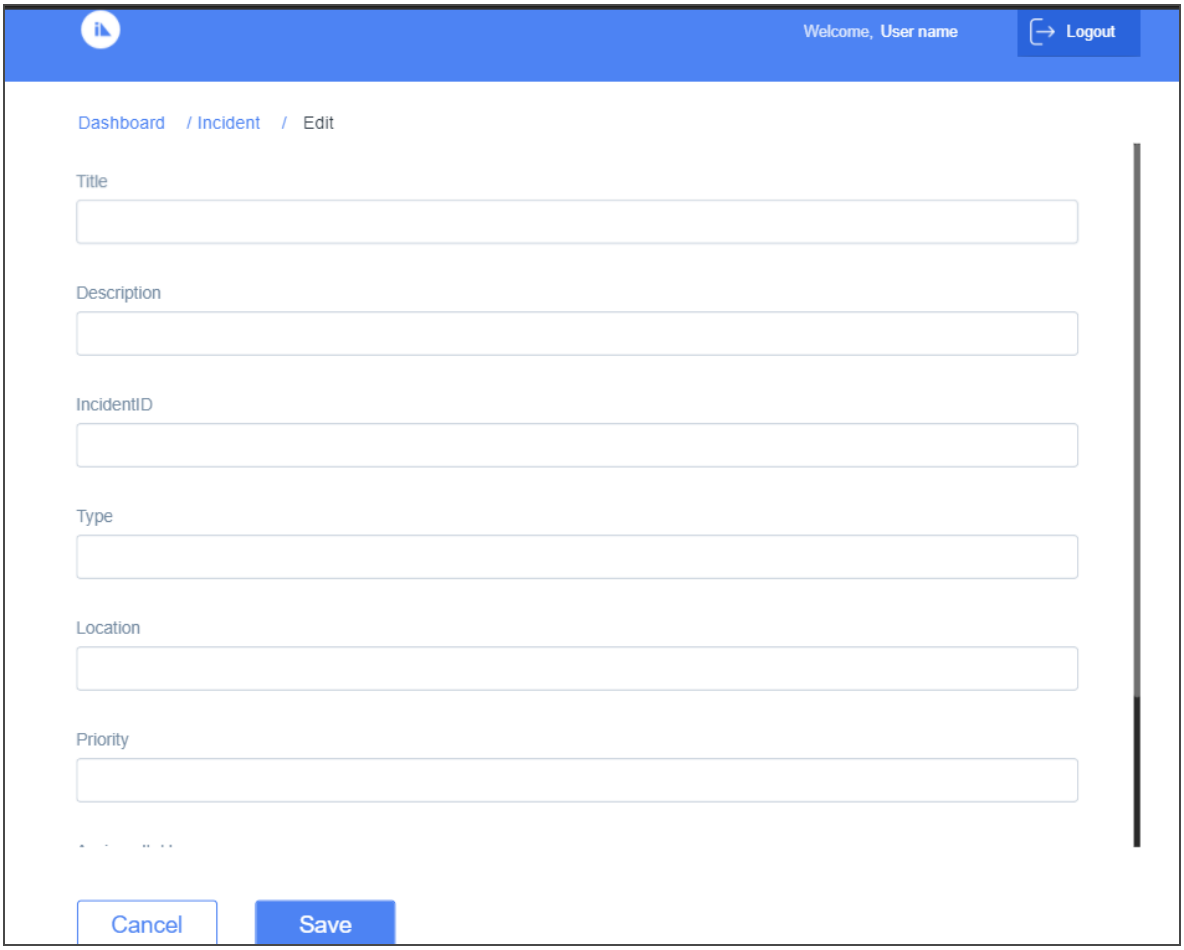
Details Form



Create Form

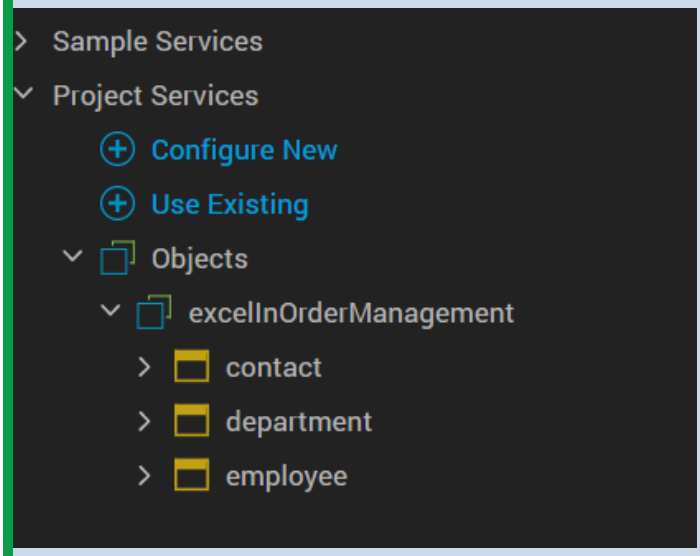


Update Form



Note: If an Object service contains multiple data model objects, then all available forms and navigation links for each data model object are generated.
For example, in the following image, the **excelInOrderManagement** Object service contains three data model objects: contact, department, and employee. So if you try to generate CRUD forms for the **excelInOrderManagement** Object service, all

available forms and associated navigation links are generated for the contact, department, and employee data model objects.



Create a Data Table for an Object Service

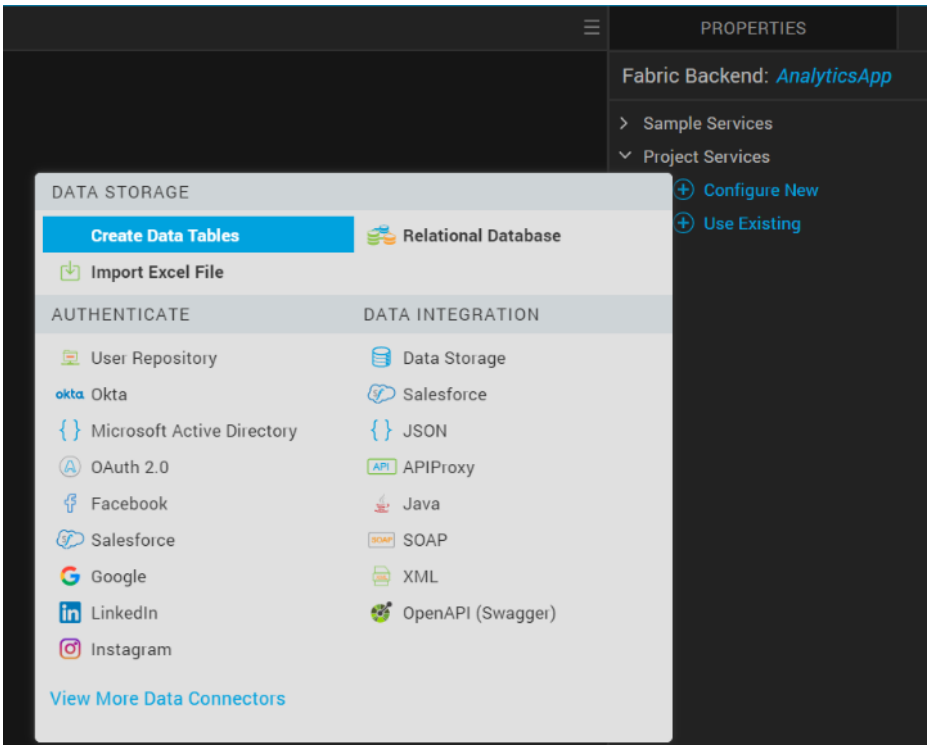
From Kony Visualizer V9, you can create a data table for an Object service. Several details such as data type, Primary key, nullable, and maximum length can also be specified for the fields of the Object Service.

> ***Note:*** You can create a data table for either an existing Kony Fabric app or for a new one.

To create a data table for an Object service from within Kony Visualizer, follow these steps:

1. In Kony Visualizer, open your project.
2. On the **Data & Services** panel, expand **Project Services**.
3. Click **Configure New**. A list of services appears.

4. Under the Data Storage section, click **Create Data Tables**. The **Edit Object Service** > **Data Model** > **Fields** window appears. A new Object is created in which you can quickly configure object fields. By default, every Object contains some system-generated fields.

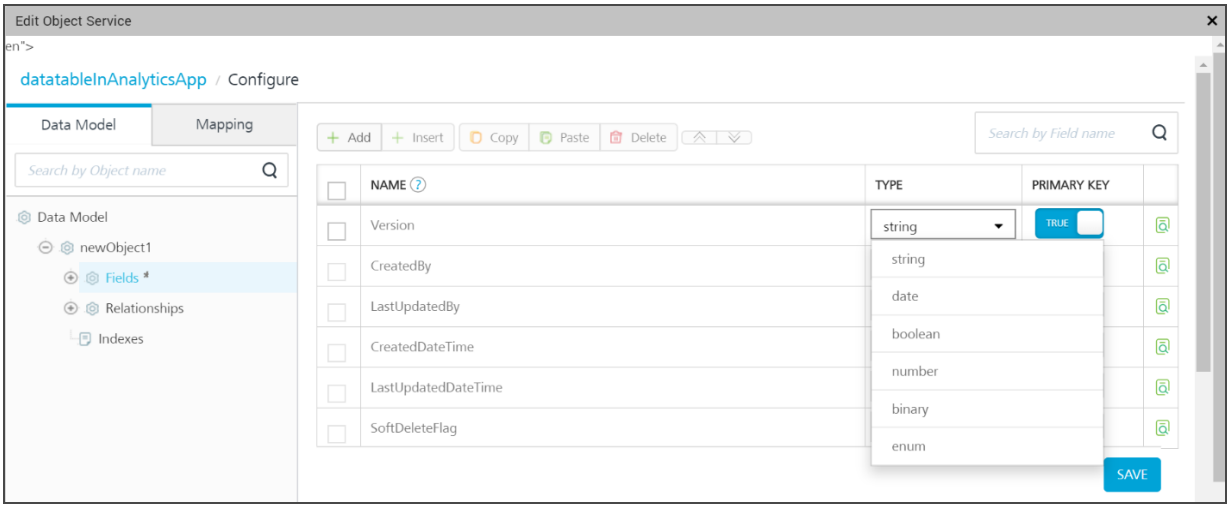


5. Click **Add**. A new field is created in the fields definition table.
6. Under the **Name** column, type the required name of the field. Here, **Version** is the field name.
7. Under the **Type** column, select the data type of the field. The available data types include string, date, boolean, number, binary, and enum. The string data type is selected by default.

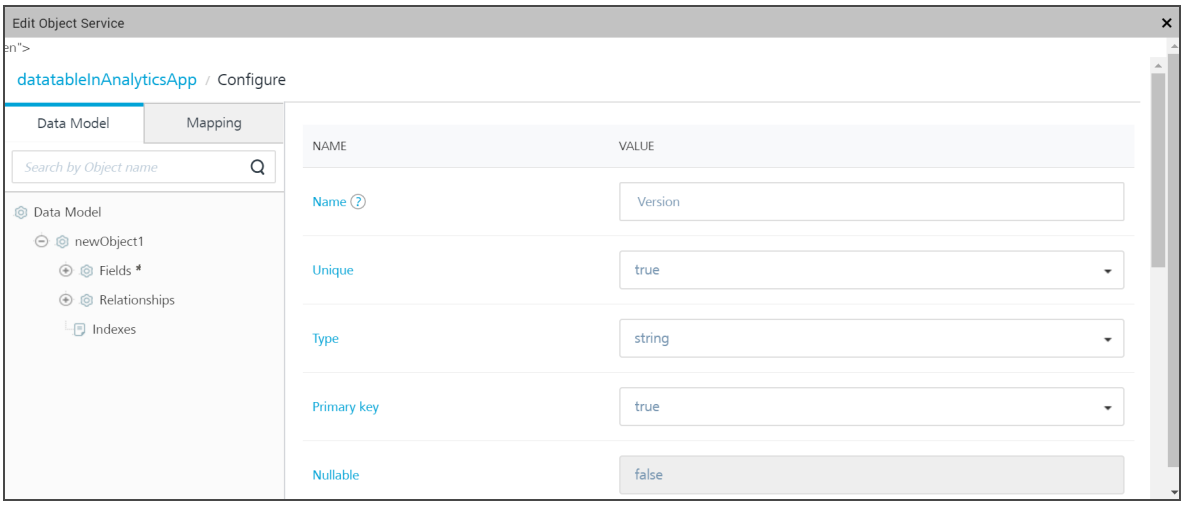
> ***Note:*** The Enumeration (**enum**) data type enables you to add a maximum of 50 comma-separated options as valid values in a field. For example, a field called **Priority** can have enumeration values such as Critical, High, Medium, Low and a field called **Gender** can

> have valid values such as Male, Female, Other. The enum data type is rendered as a drop-down list containing only those valid values that you defined, and users can choose only one option at a time for that field. This data type is available from Kony Visualizer V9. For more information on how to configure the enum data type, click here.

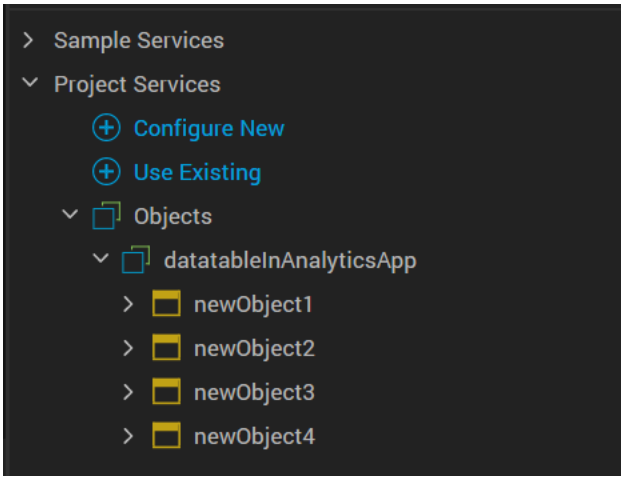
8. If you want to set the field as a Primary Key, select **Primary Key** as **True**. The Primary Key value is False by default.



9. If you want to view or modify any additional details of the field, click the View icon on the field entry row. The field details window appears. You can view or modify any editable details, and then click **Save**.



10. Click **Save**, and then close the Edit Object Service window. The data table and new data table objects are created, and are displayed under **Project Services** > **Objects** as shown here.



Note: You can create multiple objects in the data table of an Object service by following this procedure. These objects are then displayed as *newObject<X>* under Project Services on the Data & Services panel.

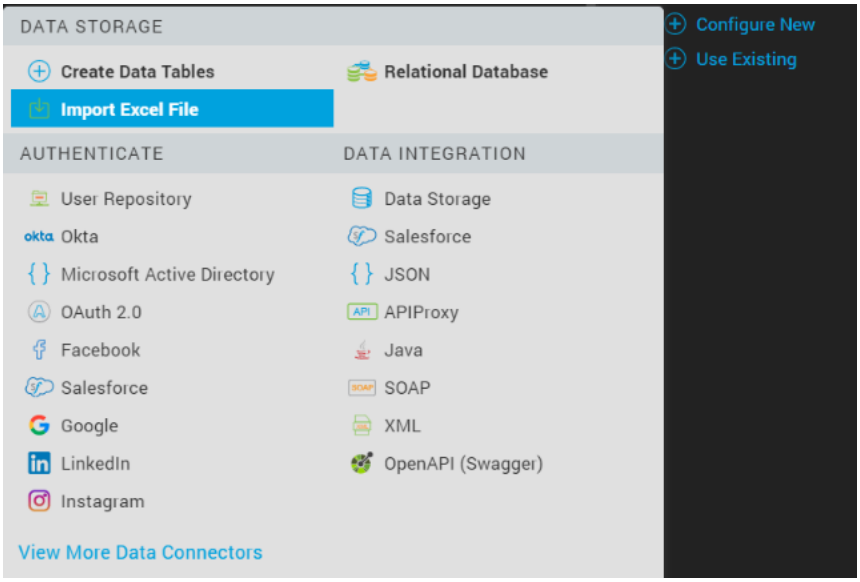
Configure an Object Data Model by Importing an Excel File

From Kony Visualizer V9, you can create an object data model for an Object service by importing a locally-stored Microsoft Excel file. With this feature, low-code developers have an easy and faster way to import an object data model from a locally-stored Excel file.

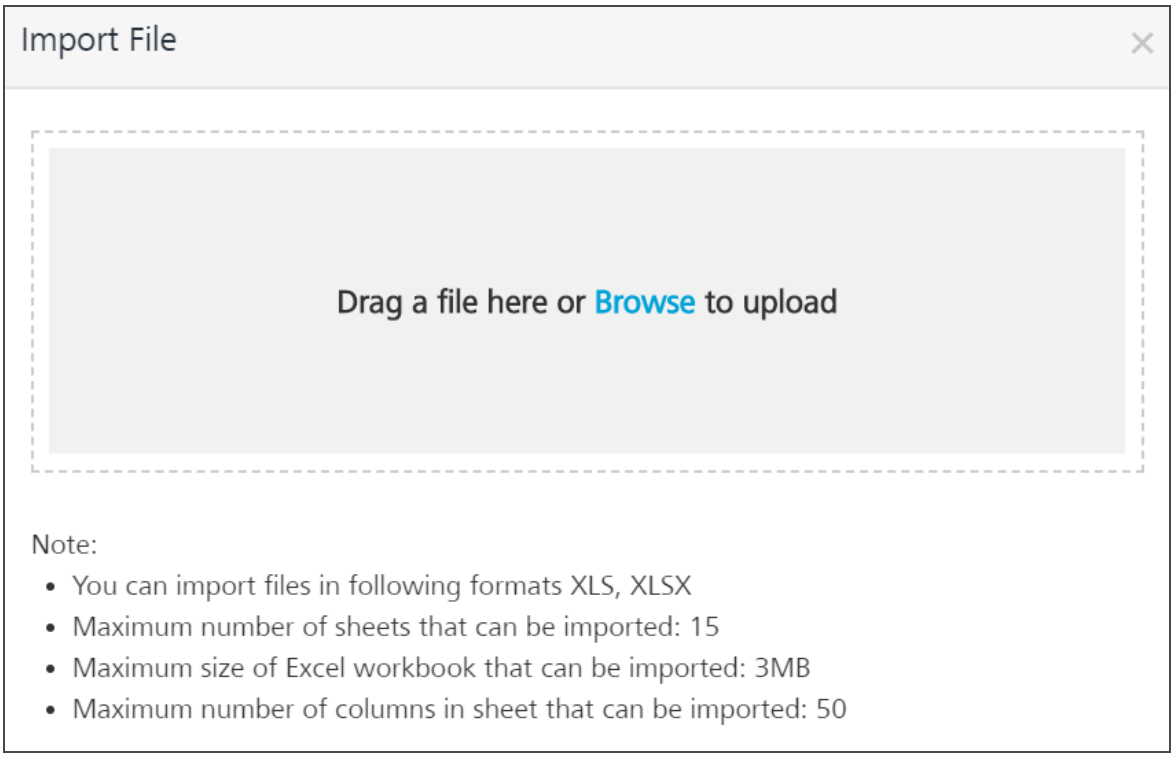
> ***Note:*** You can import an Excel file for either an existing Kony Fabric app or for a new one.

To create an object data model by importing an Excel file for an Object Service from within Kony Visualizer, follow these steps:

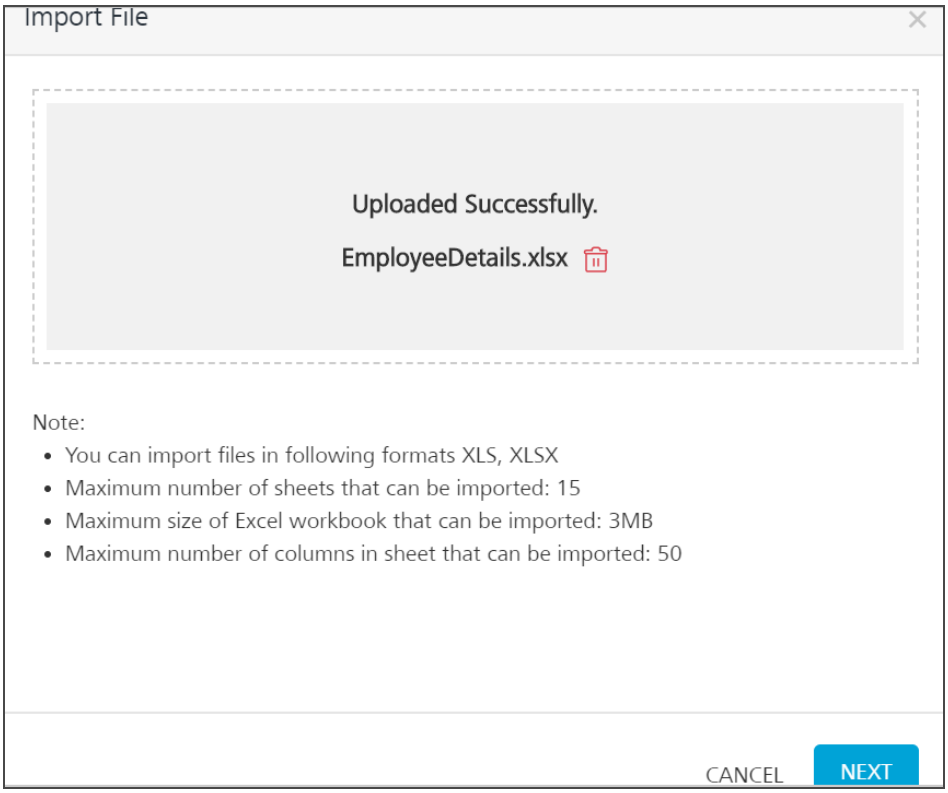
1. In Kony Visualizer, open your project.
2. On the **Data & Services** panel, expand **Project Services**.
3. Click **Configure New**. A list of services appears.
4. Under the Data Storage section, click **Import Excel File**. The **Import File** window appears.



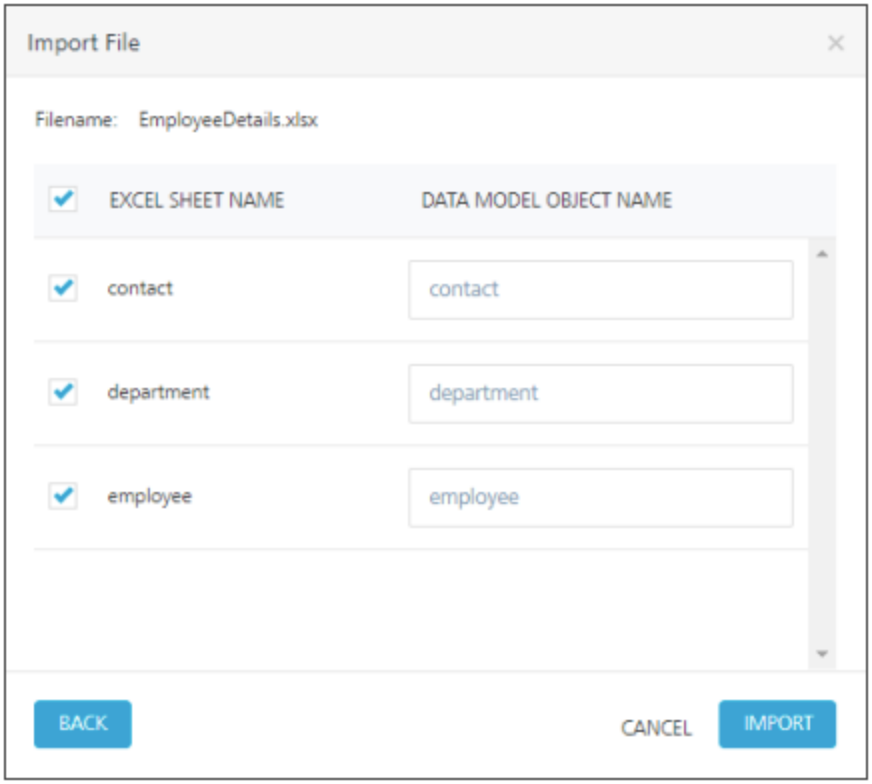
5. You can perform any one of the following actions:
   - Directly drag and drop the Excel file.
   - Click **Browse** to search for, and then select the Excel file from your local system.



6. After the Excel file has been successfully uploaded, a confirmation message appears. Click **Next**.
   If you want to delete the uploaded file and import a different Excel file instead, click the Delete icon.

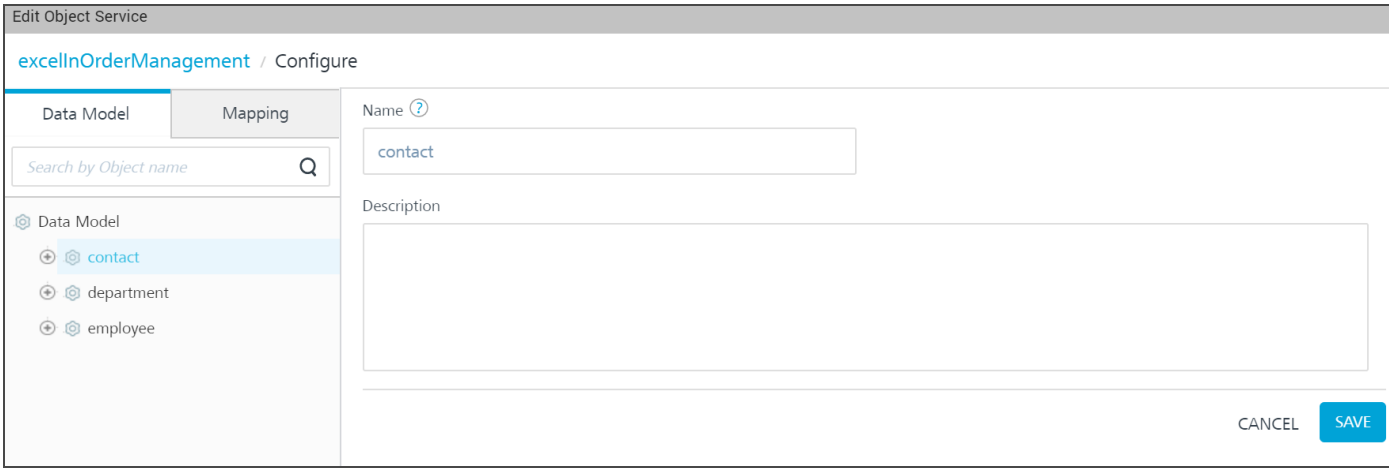


7. The uploaded Excel Sheet is selected by default. In the **Data Model Object Name** box, type the required name of the data model. If there are multiple sheets in the Excel file, you can add the name of each sheet.
   You can add the underscore (_) symbol in the data model object name. However, you cannot include any spaces or special characters in the data model object name.

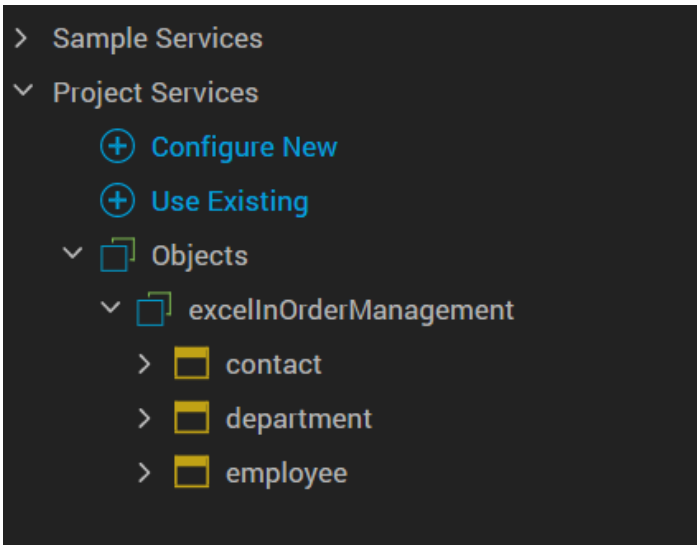


8. Click **Import**. After the Excel sheet has been successfully imported, a window with a confirmation message appears. Click **Done**.

9. The **Edit Object Service** window for the respective Kony Fabric app appears. You can use this window to configure various items of the Object service data model, such as the Name, Description, Fields, Relationships, Indexes, and Primary Keys. For more information on how to configure the data model of an Object Service, refer the Configuring a Data Model topic in the Kony Fabric User Guide.



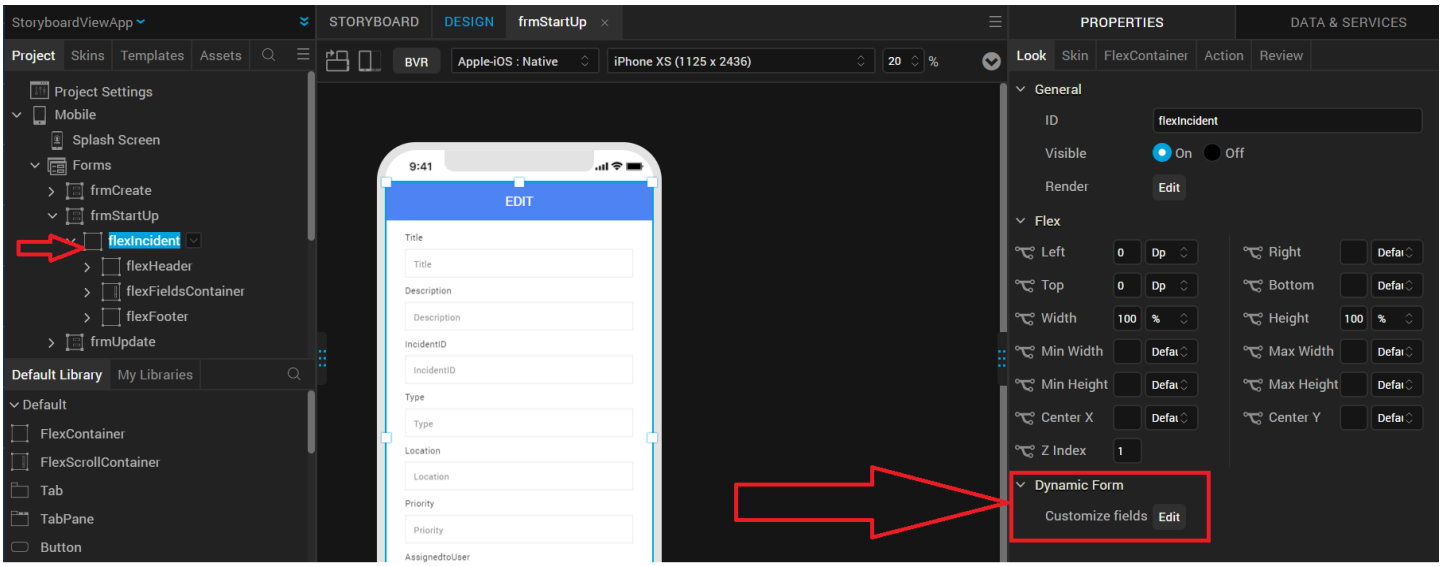
10. After you have configured the Object service data model, click **Save** and then close the **Edit Object Service** window. The object data model is created, and is displayed under **Project Services** > **Objects**.



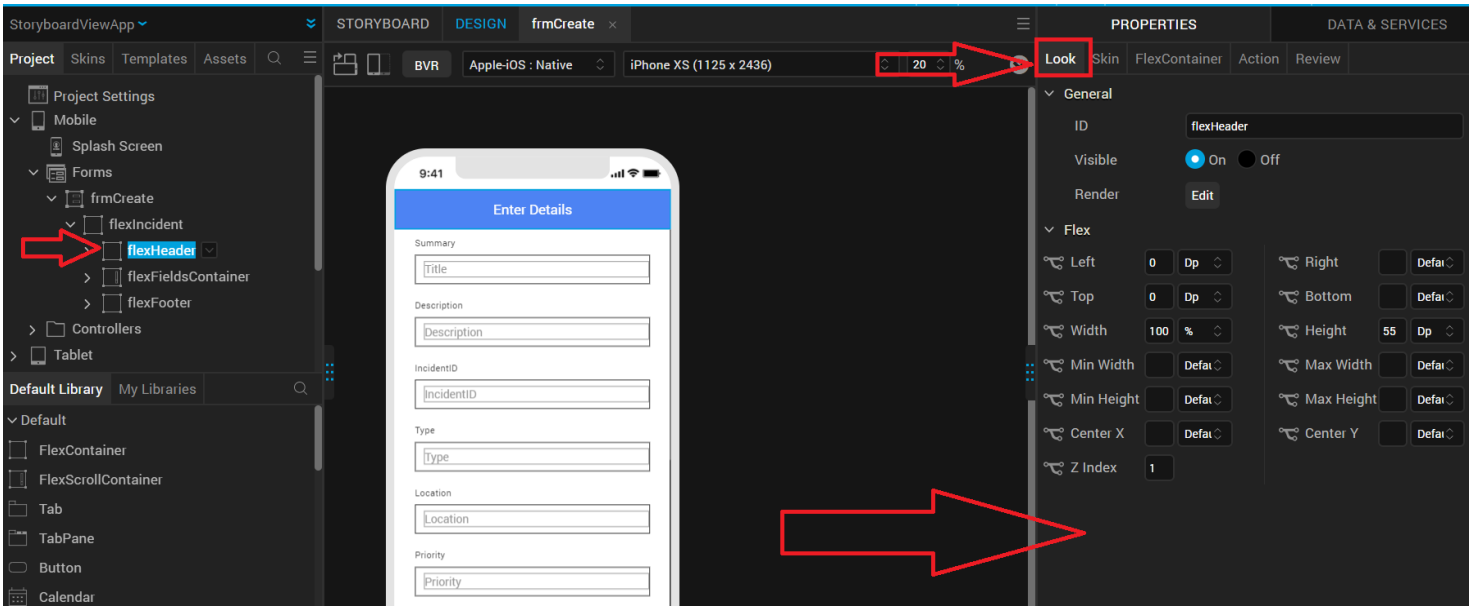
Customize the Generation of Data Model Objects

From Kony Visualizer V9, you can configure the generation of data model objects while designing your app. This enhancement enables you to select, reorder, and customize various properties of all the fields that you want to be displayed on the required screen of your app.

> ***Important:*** To dynamically configure the fields of the object model, you must click the parent container of the request/response parameter on the form. Then go to **Properties** panel > **Look** tab > **Dynamic Form** section, and click **Edit** beside **Customize fields**.
> In the following image, **flexIncident** is the parent container of the service parameter on the **frmStartUp** form. **Only while the parent container is selected, the Dynamic Form section appears under Properties panel > Look tab.**



> ***Important:*** If you do not click the parent container of the request/response parameter on the form, the Dynamic Form section does not appear under Properties panel > Look tab. **While any child container of the service parameter or the form itself is selected, the Dynamic Form section does not appear.**
> In the following image, **flexHeader** is a child container of the service parameter on the **frmStartUp** form. While this child container is selected, the Dynamic Form section does not appear under Properties panel > Look tab.



To understand this feature in detail, let us consider a scenario where Steve (a low-code developer) wants to create an app called **Incident Management**. He has already created the associated Kony Fabric app, and now wants to configure the app's UI.

The Admin of an airport can use the Incident Management Responsive Web app to view, create, update, and assign incidents/tasks to relevant staff members. In addition, staff members can use the Incident Management Mobile app to view and act upon their assigned tasks as well as to create new tasks.

The Responsive Web and Mobile apps comprise of four forms: list, detail, update, and create.

This topic consists of the following sections:

- Prerequisites
- Configure Dynamic UI Properties

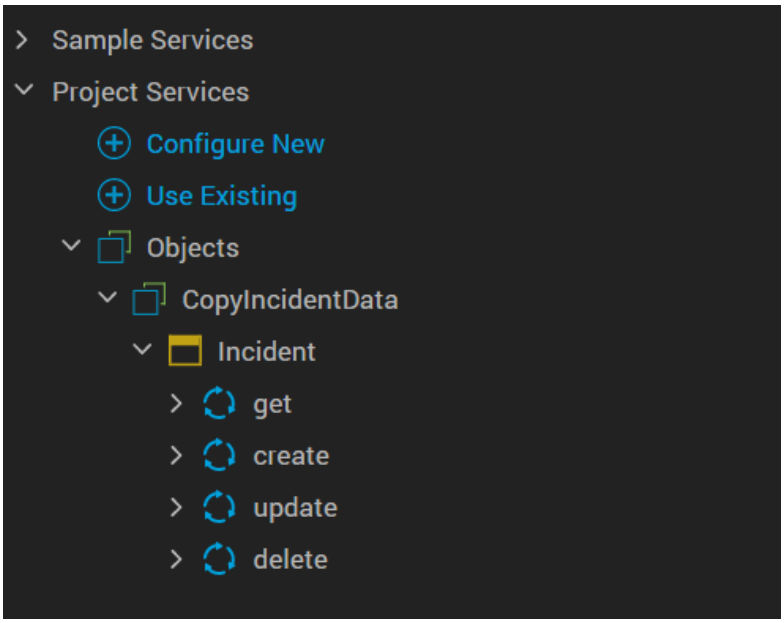
Prerequisites

- Create a Kony Fabric app.
- Link the Kony Fabric app to your Kony Visualizer project.

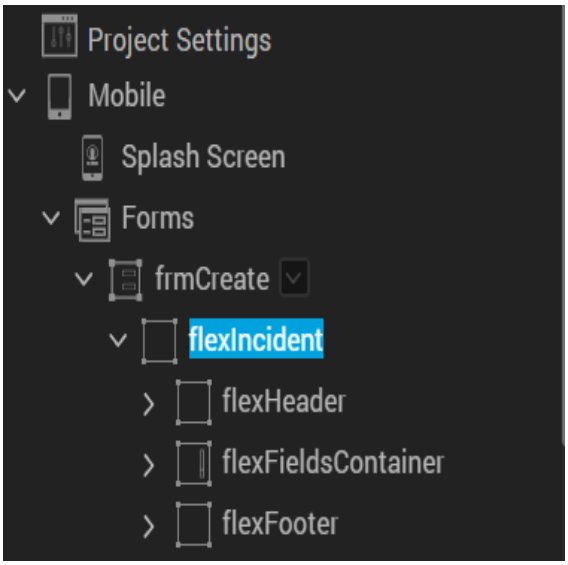
Configure Dynamic UI Properties

In this section, we will configure the dynamic properties of the fields on the Create Incident Entry screen of the Incident Management Mobile app:

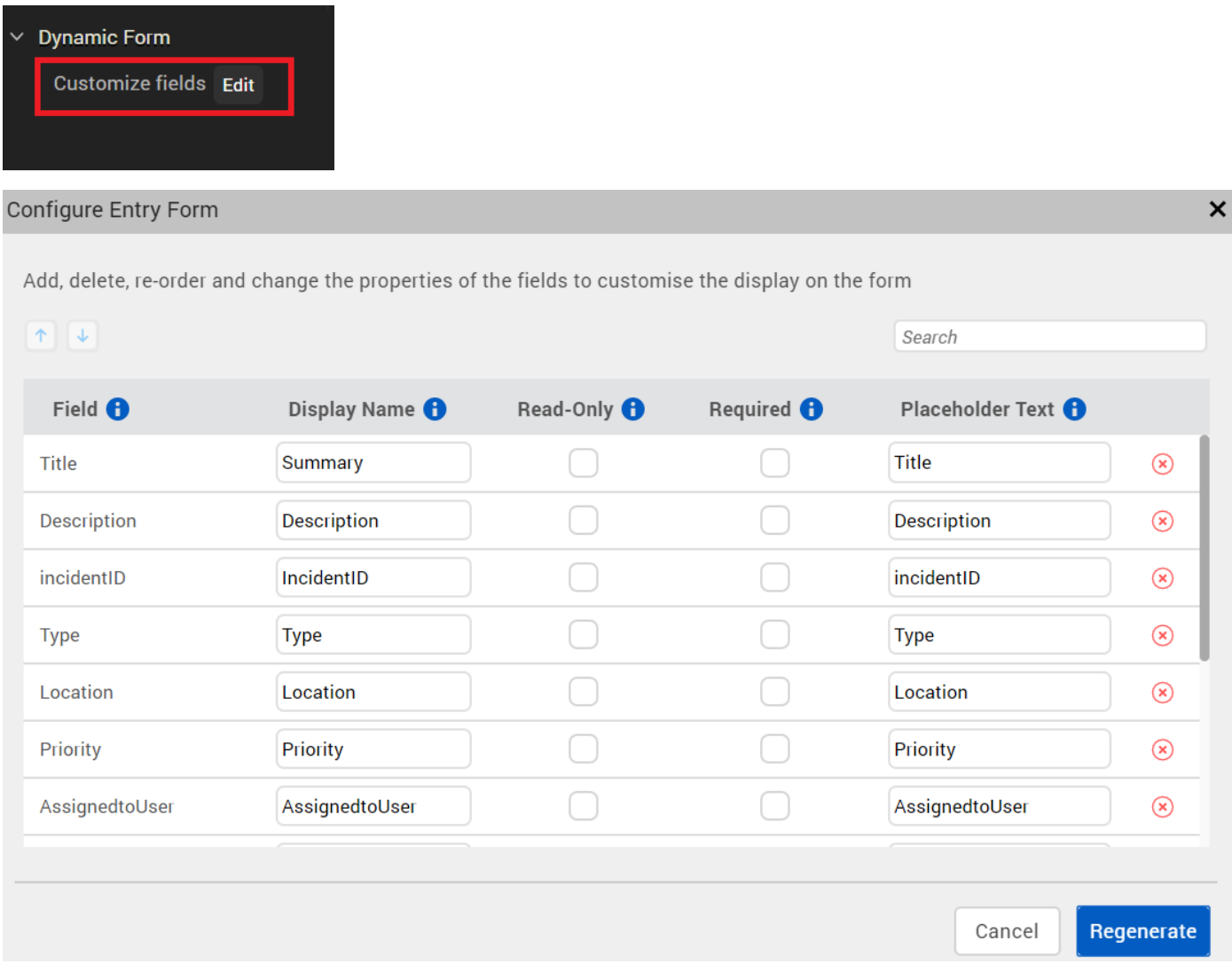
1. In your Kony Visualizer project, go to **Project Explorer** > **Mobile** > **Forms**, and then create a new form called **frmCreate**.
2. Go to the **Data & Services** panel > **Project Services** > **Objects** > **CopyIncidentData** > **Incident**.



3. Drag and drop the **create** method onto the Project Canvas. The entry form is generated.
4. On the Project Explorer, under **frmCreate**, click **flexIncident**. Here, flexIncident is the parent container of the create Request parameter.

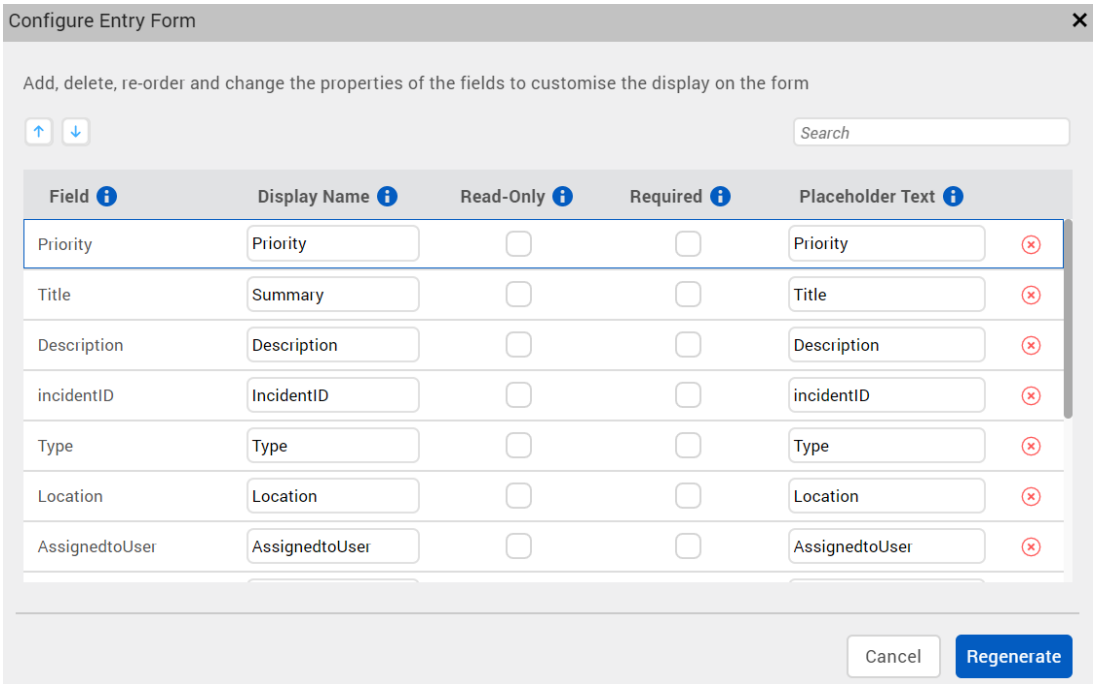


5. Go to **Properties** panel > **Look** tab > **Dynamic Form**, and then click **Edit** beside **Customize fields**. The **Configure Entry Form** window appears.

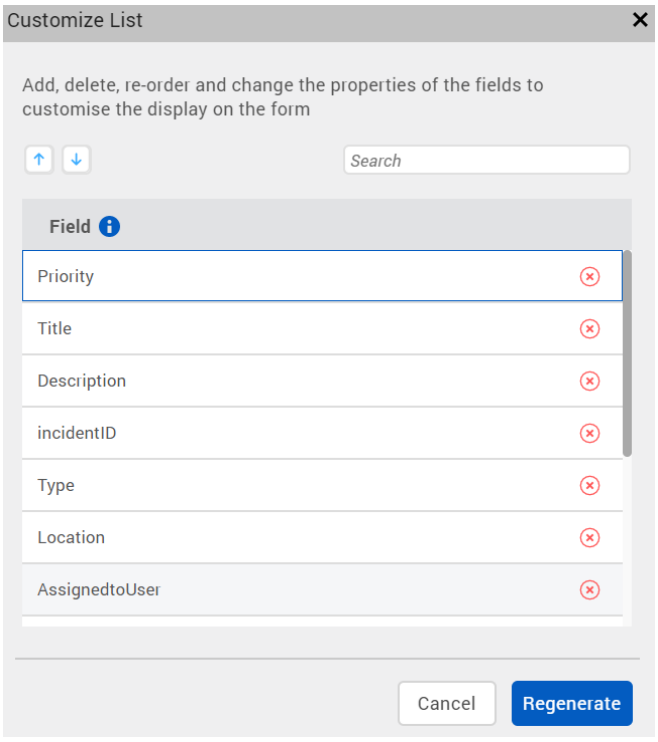


You can use the **Configure Entry Form** window to configure the dynamic properties of the fields. In this procedure, we will discuss the following operations that can be performed on a field called **Priority** on the **frmCreate** form:

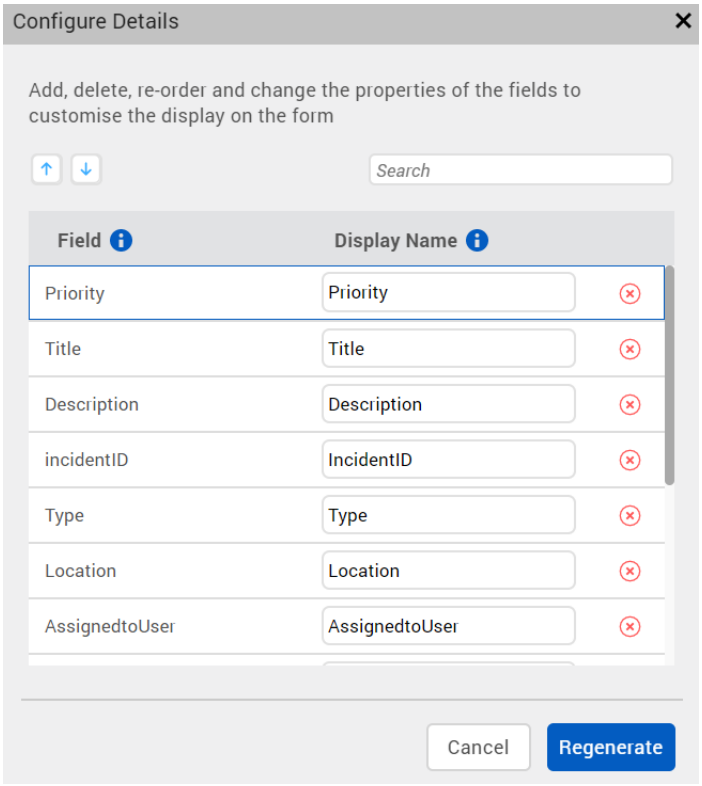
- **Reorder a Field**: In the **Configure Entry Form** window, click the **Priority** field. Keep clicking the Move Up icon until the **Priority** field is at the top of the list as shown.



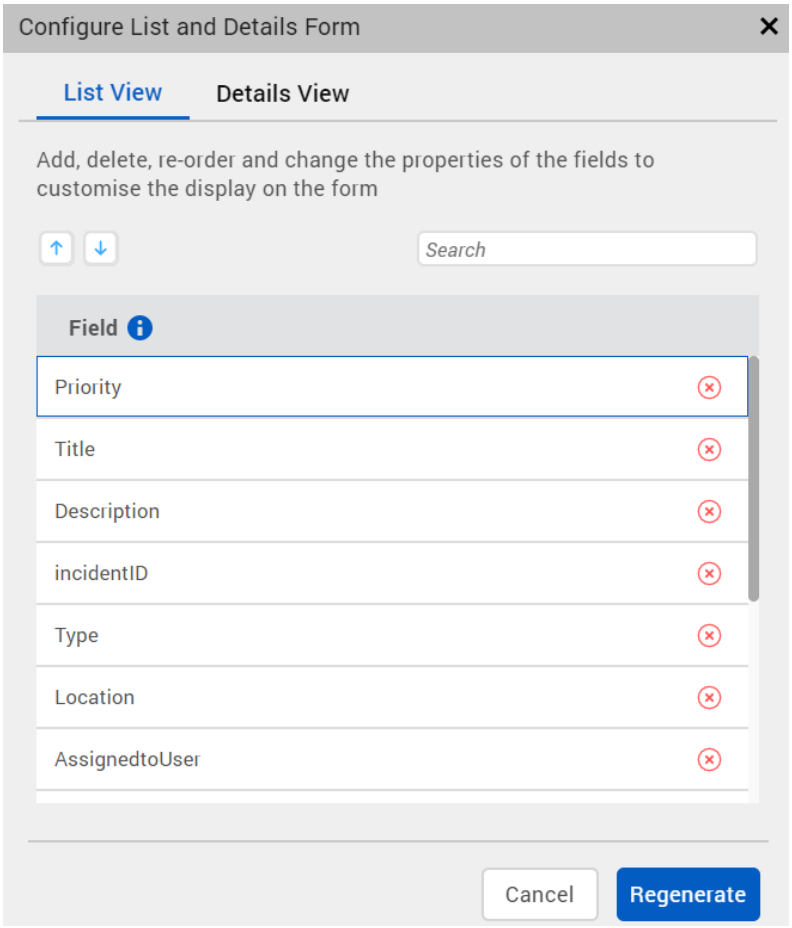
In List Using Details Forms

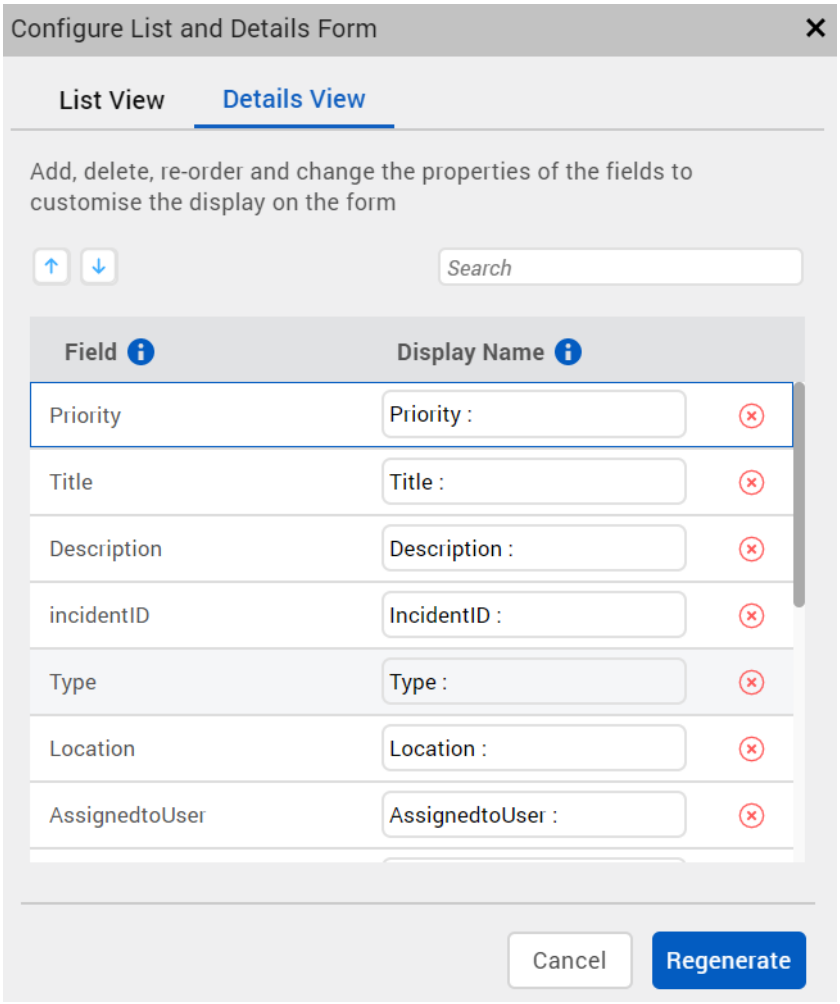


In Details Using Response Forms

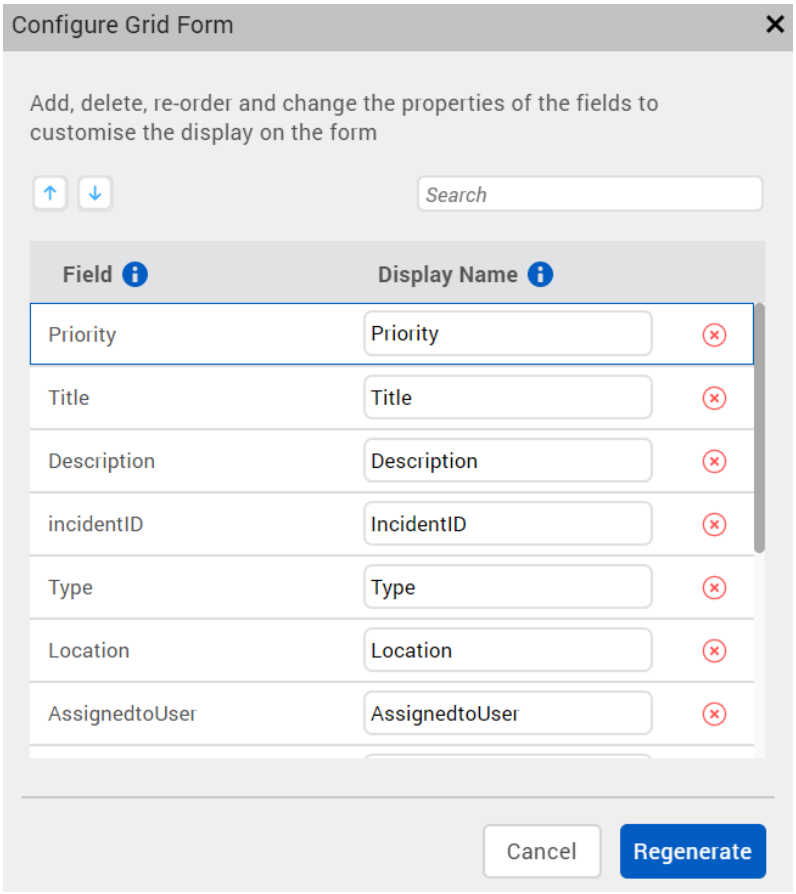


In List and Details Forms



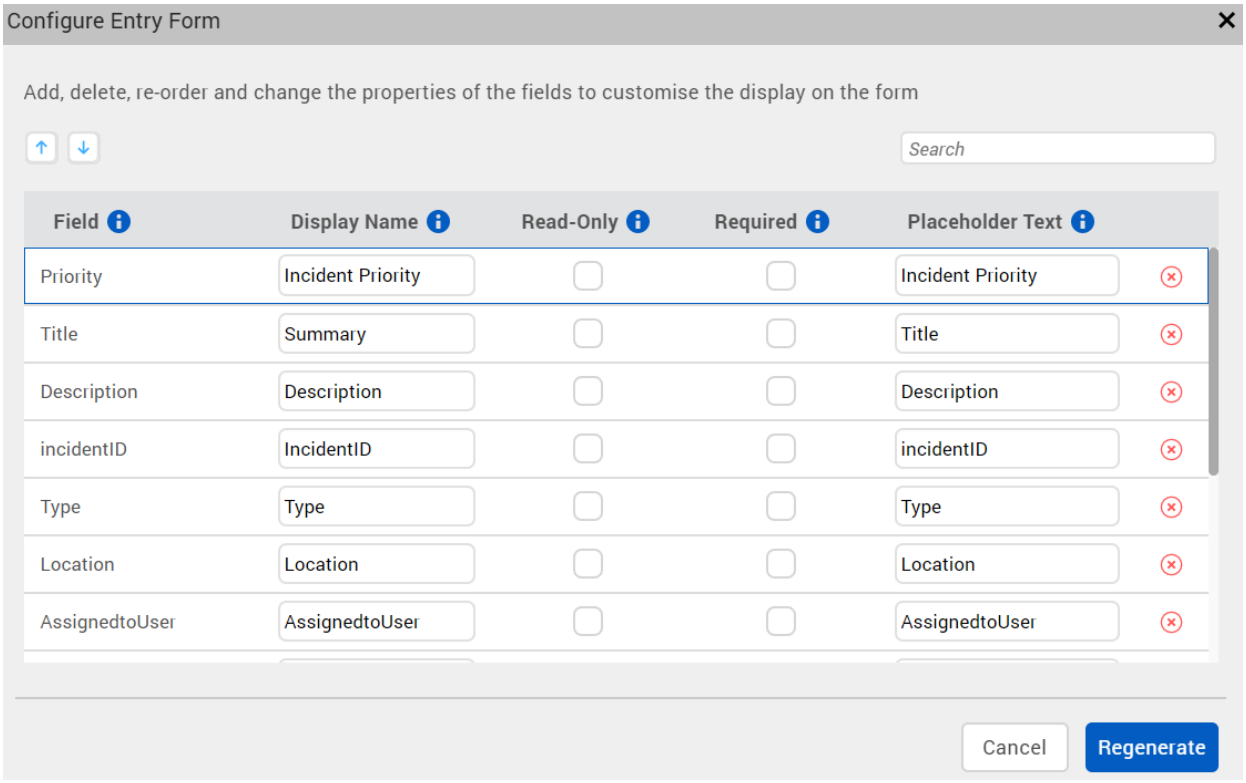


In Grid Forms

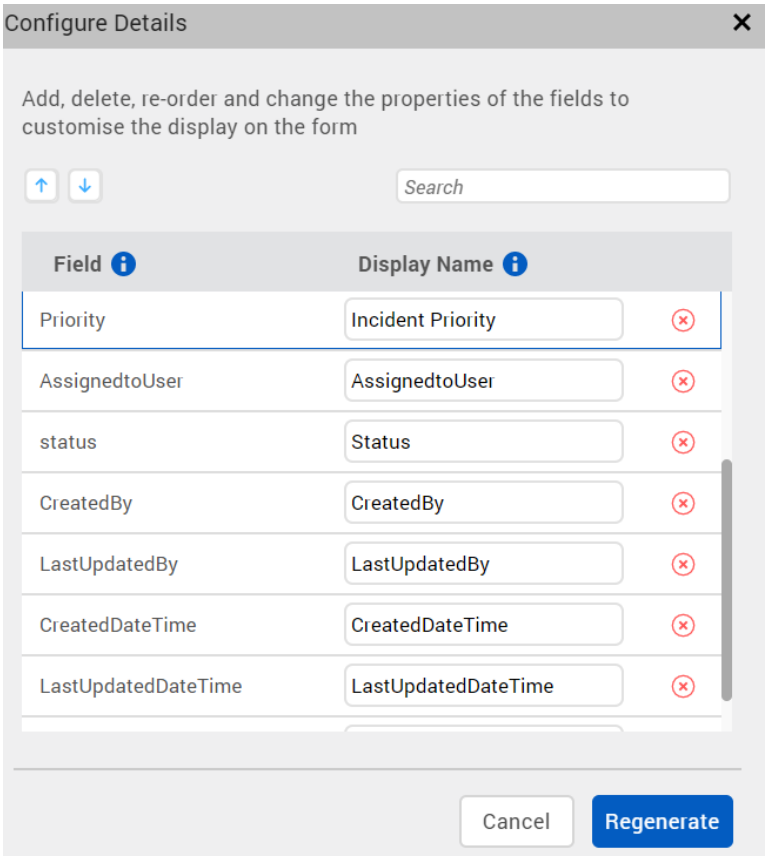


> ***Note:*** If you want to move a field down the order, select the field and click the Move Down icon as many times as necessary.

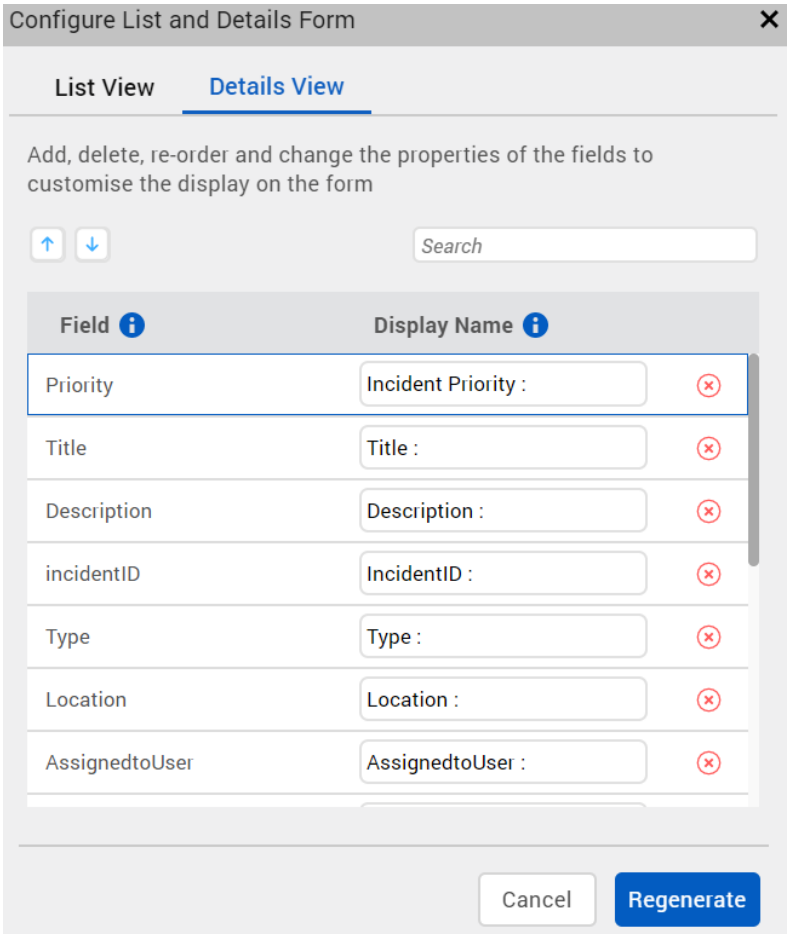
- **Add/Modify the Display Name and Placeholder Text**: In the **Configure Entry Form** window, for the **Priority** field, type the **Display Name** as **Incident Priority**. Similarly, type the **Placeholder Text** as **Incident Priority** as shown.
  Display Name signifies the name of the field that is displayed on the screen. And **Placeholder Text** enables you to provide a help text about the field.



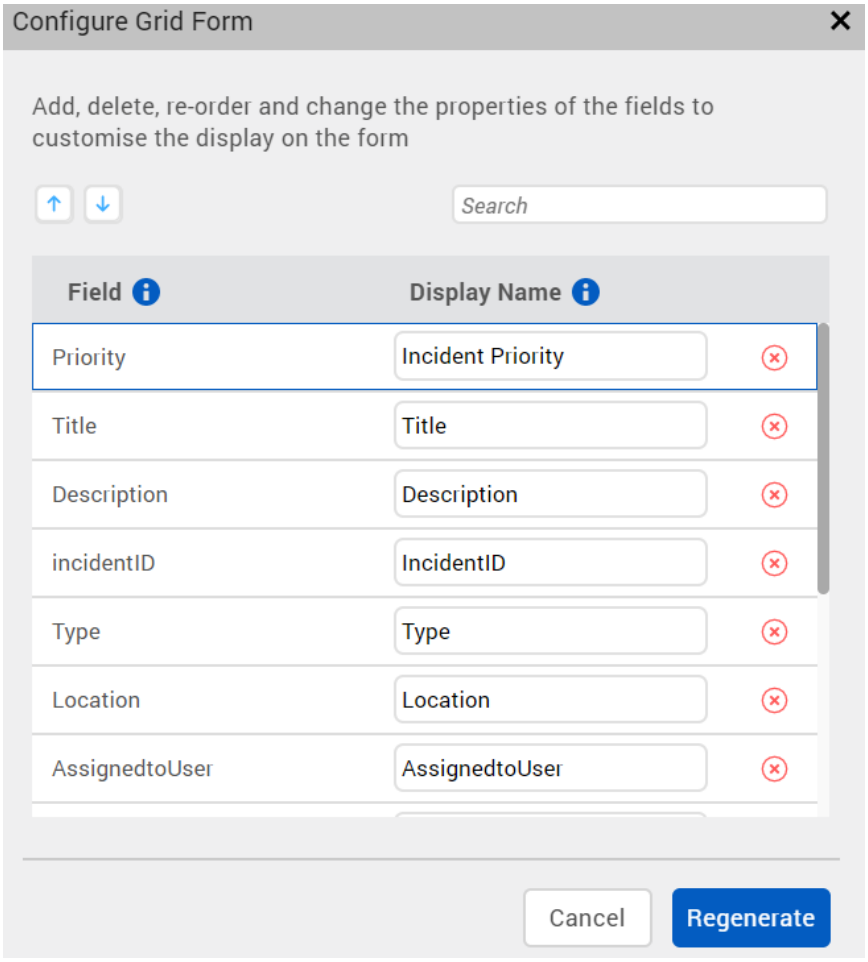
In Details Using Response Forms



In List and Details Forms

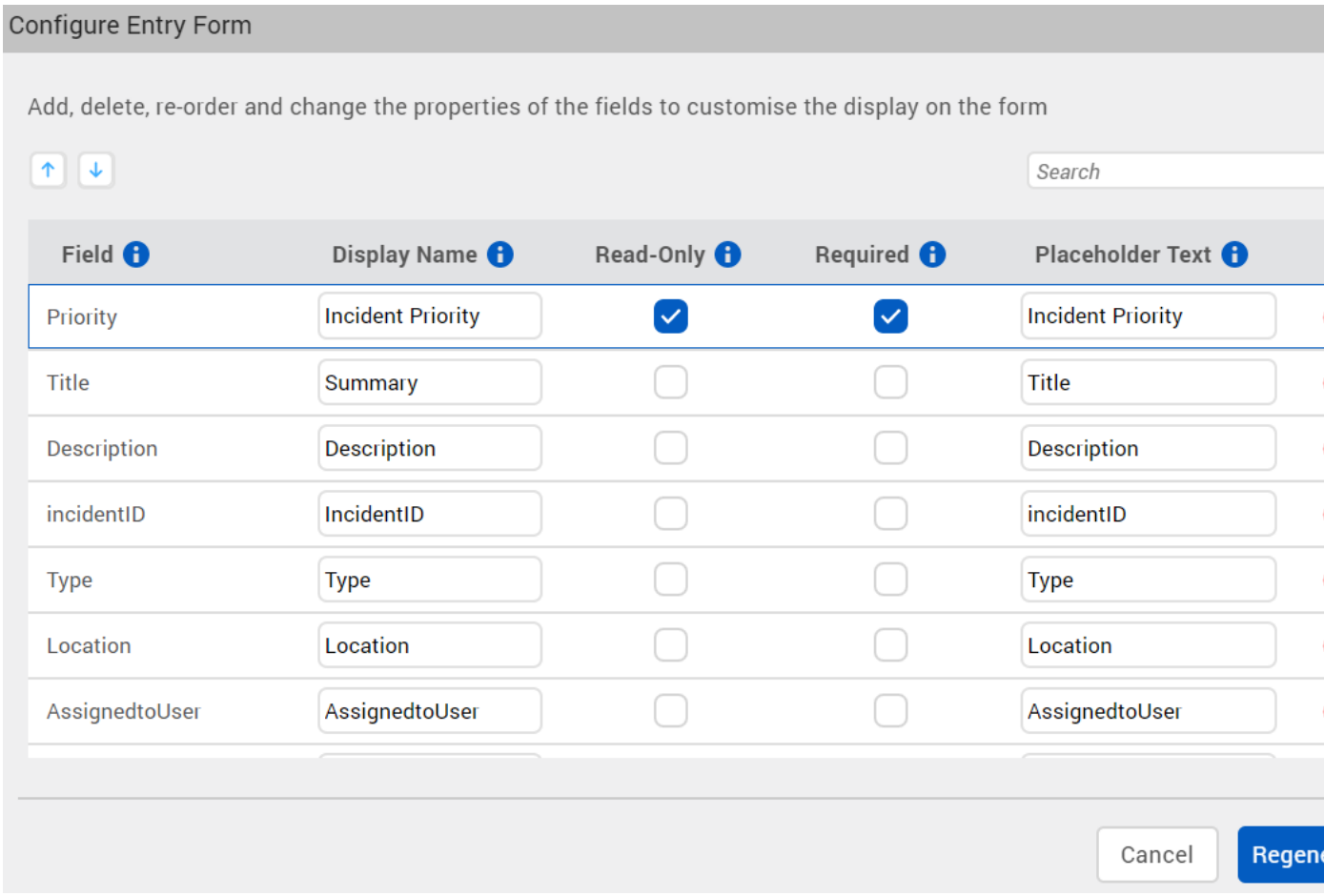


In Grid Forms



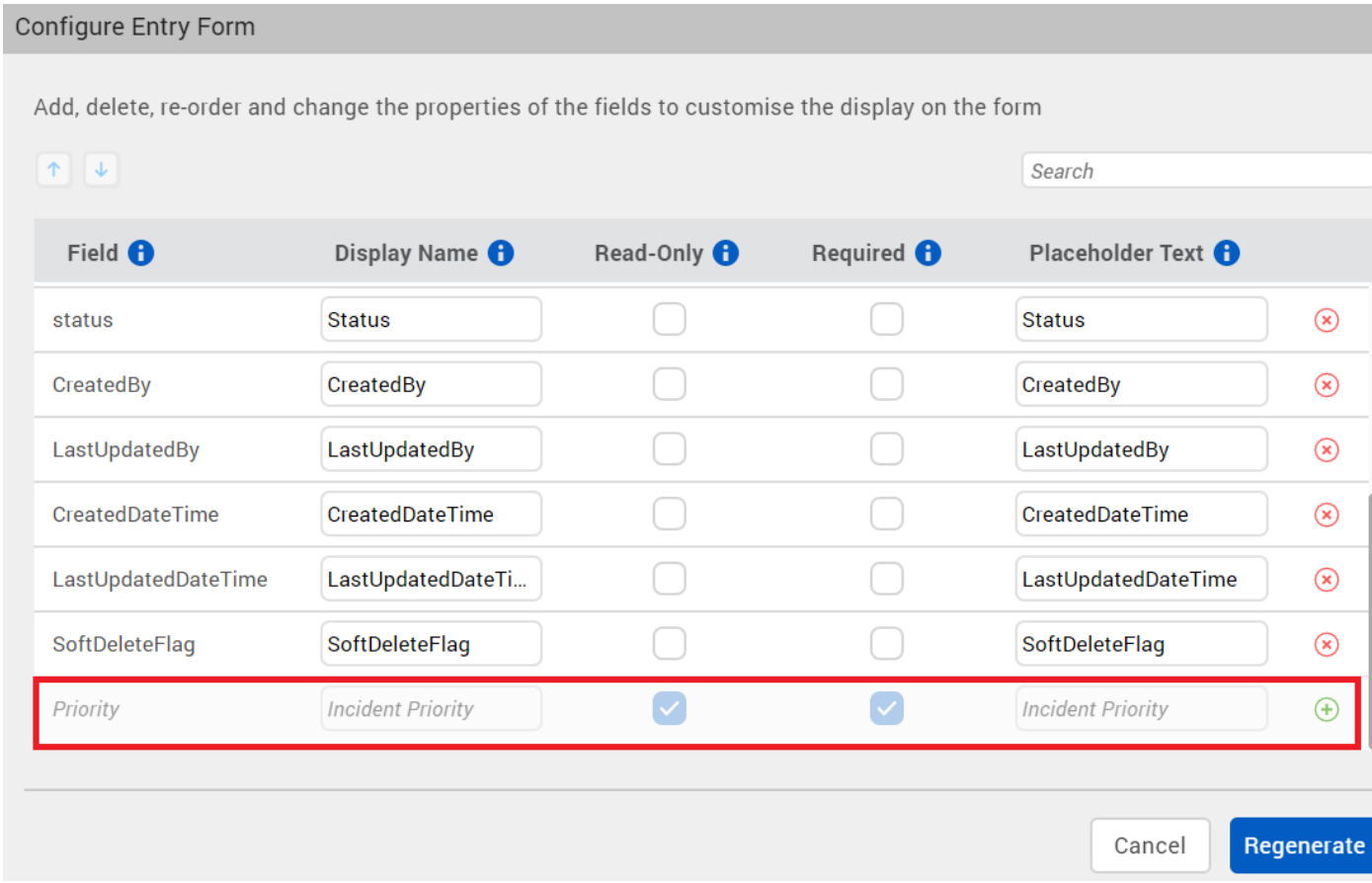
> ***Note:*** The Display Name and Placeholder Text fields are not applicable for list (List Using Response) forms. And the Placeholder Text field is not available for detail (Details Using Response) forms.

- **Configure the Read-Only and Required Properties**: In the **Configure Entry Form** window, for the **Priority** field, select the **Read-Only** and **Required** check boxes.
  Read-only fields signify that these fields cannot be edited by users. And **Required** fields signify that it is mandatory for users to specify the information for these fields; an asterisk is displayed on the UI for all **Required** fields.

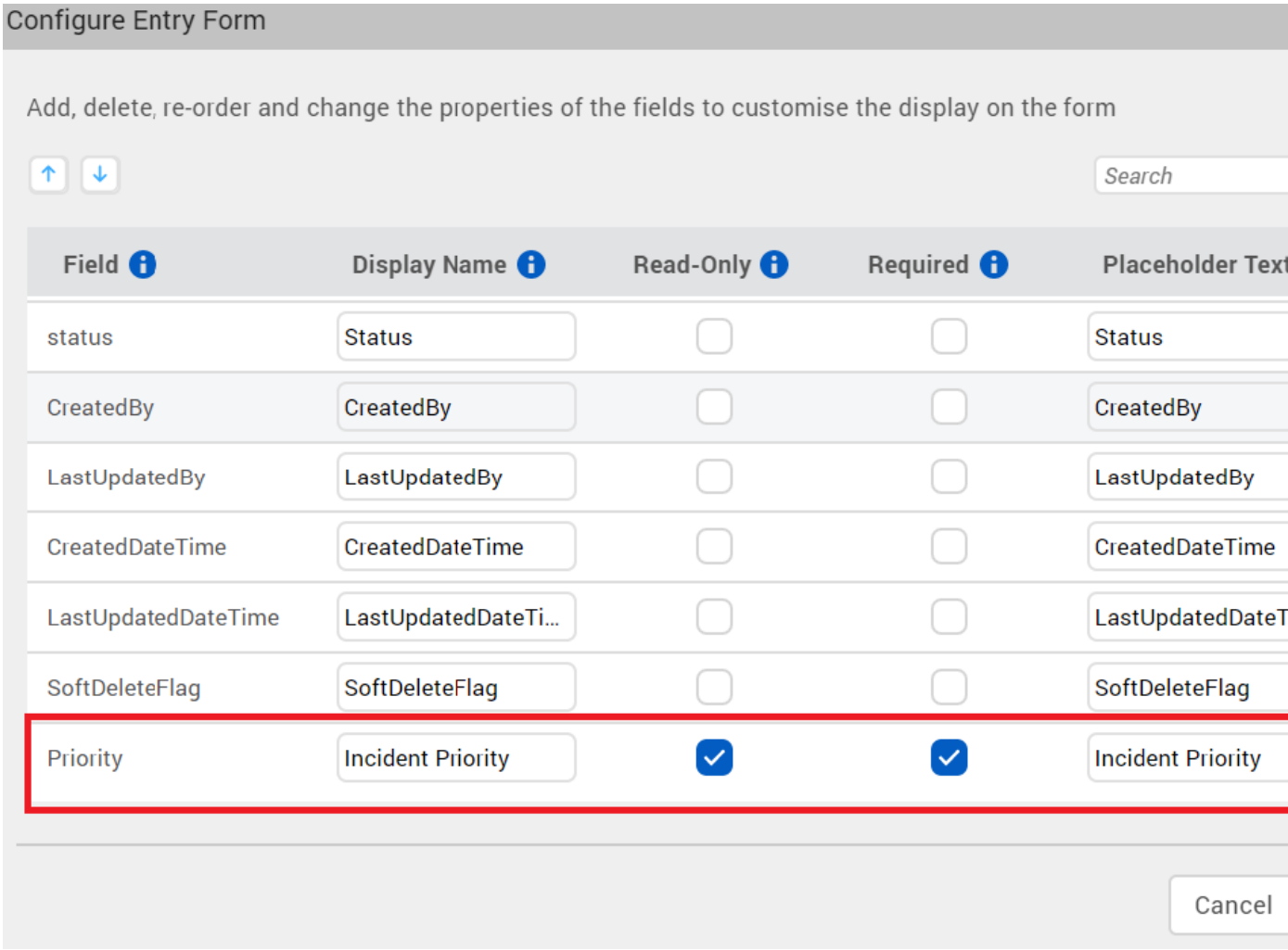


> ***Note:*** The Read-Only and Required check boxes are only applicable in Entry forms; as part of the **create** and **update** methods. These check boxes are not available in List Using Response and Details Using Response forms.

- **Delete/Add a Field:** In the **Configure Entry Form** window, for the **Priority** field, click the Delete icon. The Priority field is removed from the top of the list, moved to the bottom, and the text is displayed in gray color.

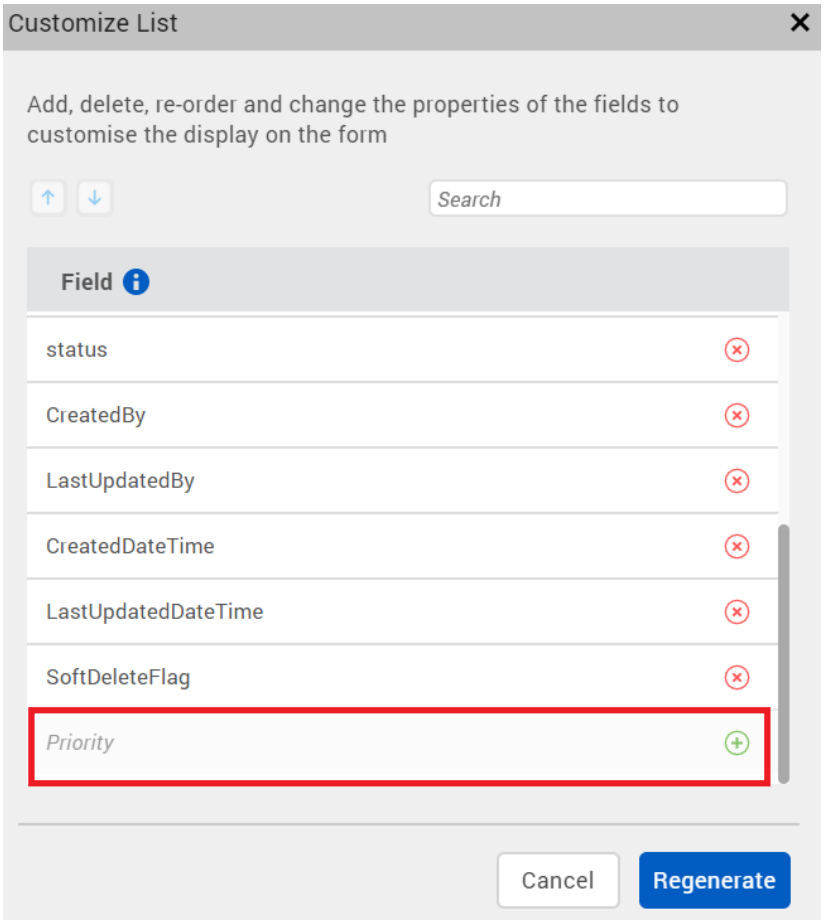


If you want to reinstate the **Priority** field, click the Add icon beside it. The Priority field becomes active again.

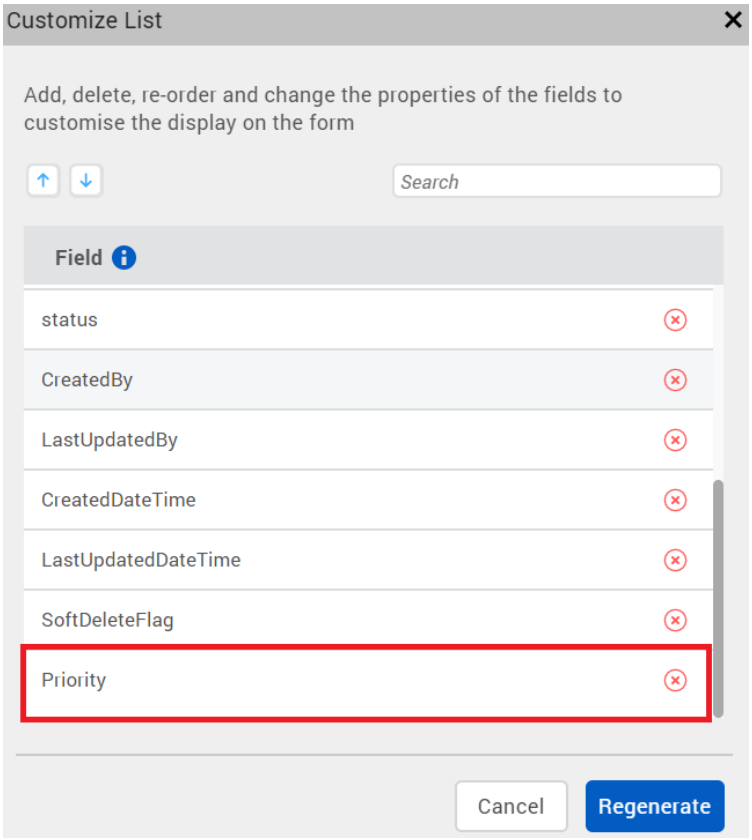


In List Using Response Forms

Delete Operation

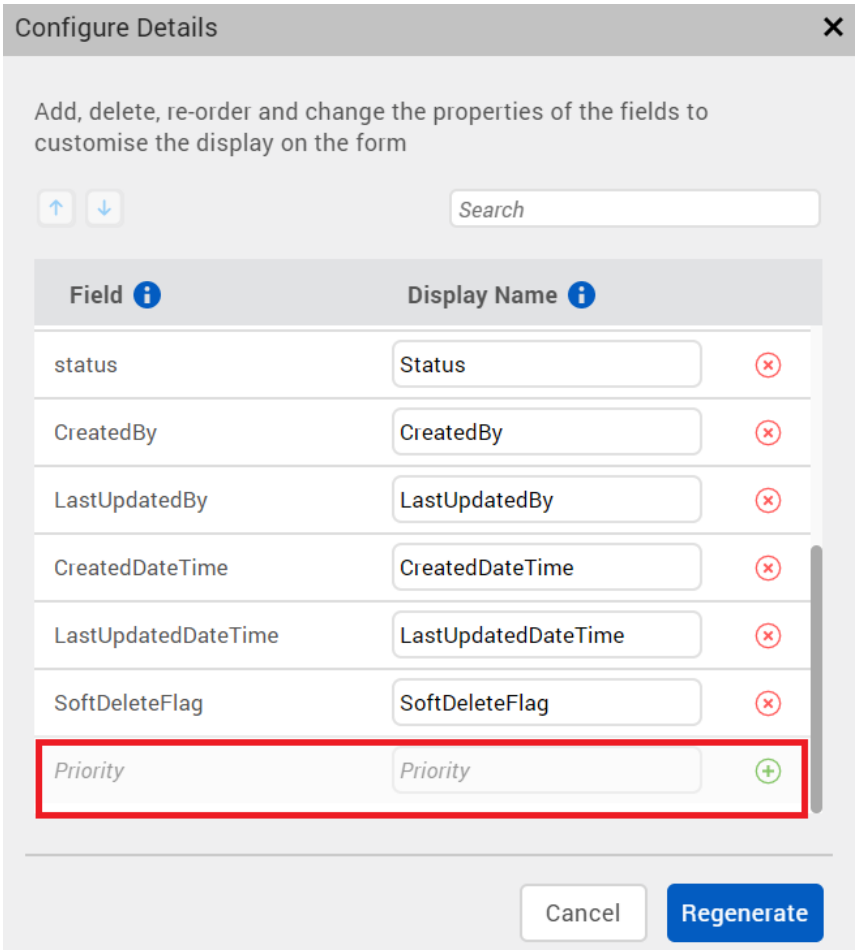


Add Operation

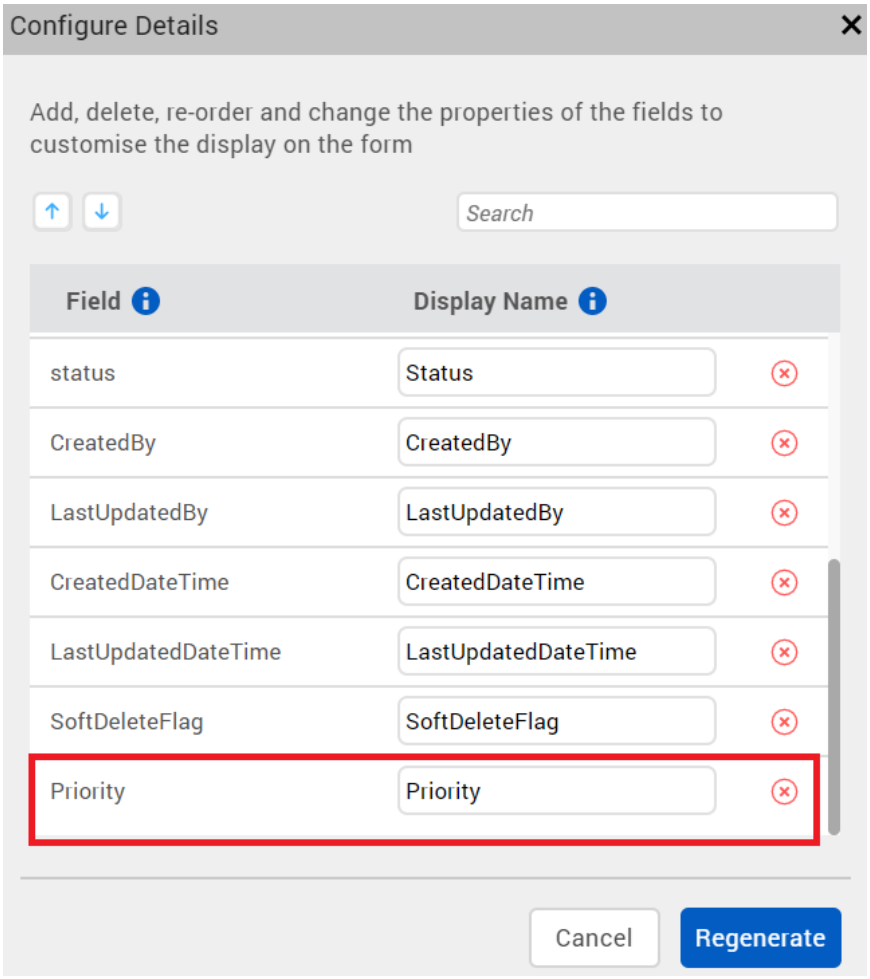


In Details Using Response Forms

Delete Operation

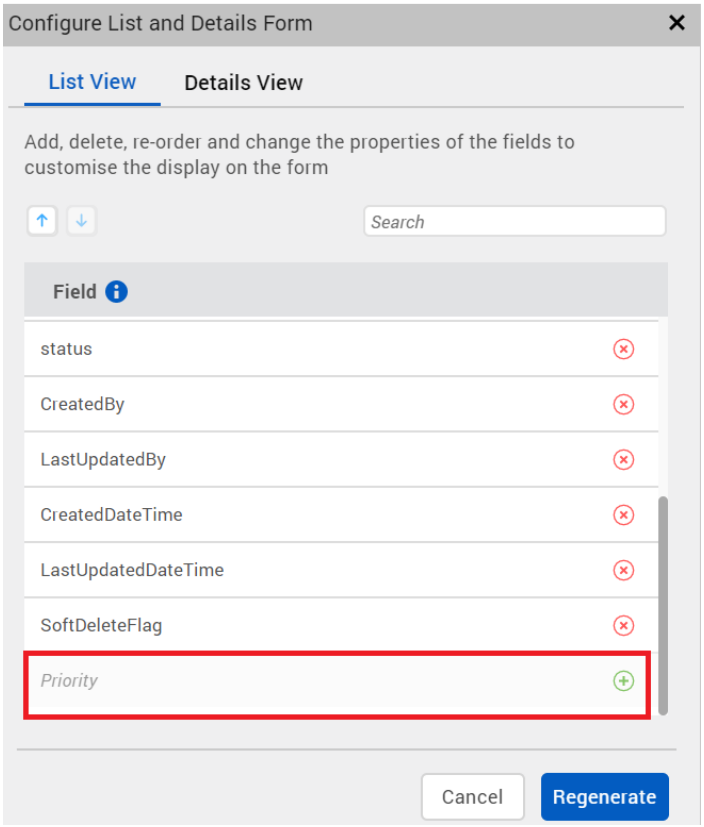


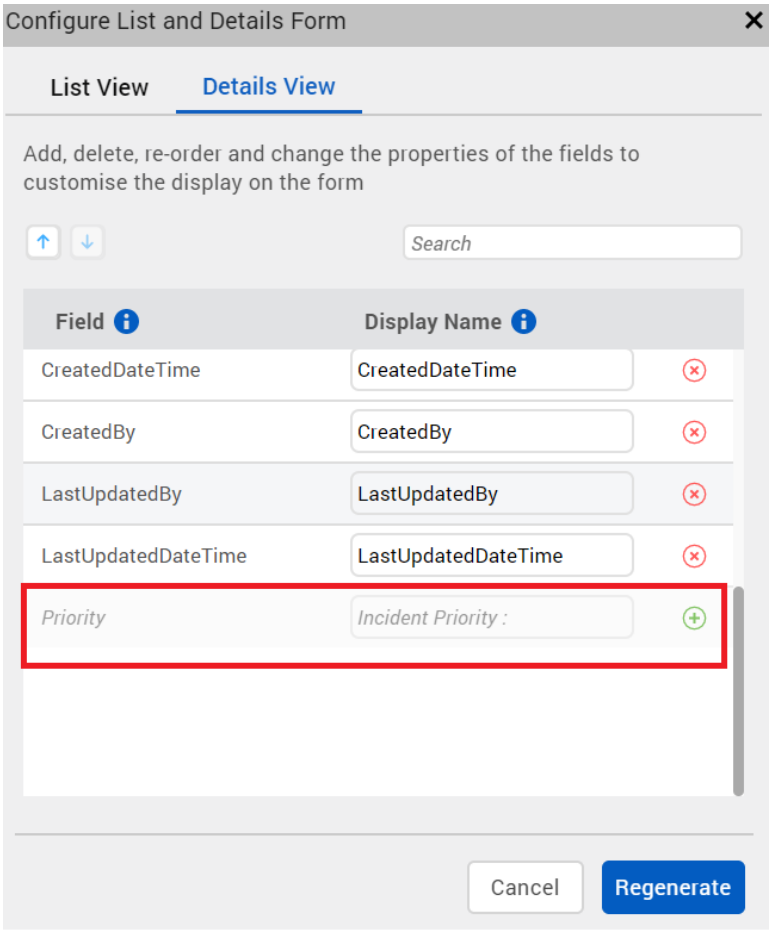
Add Operation



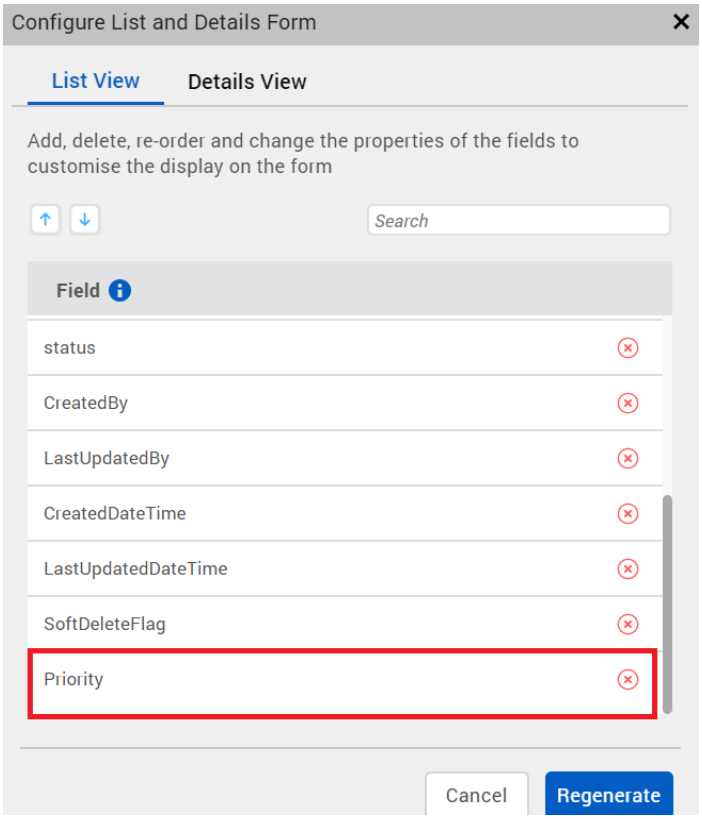
In List and Details Forms

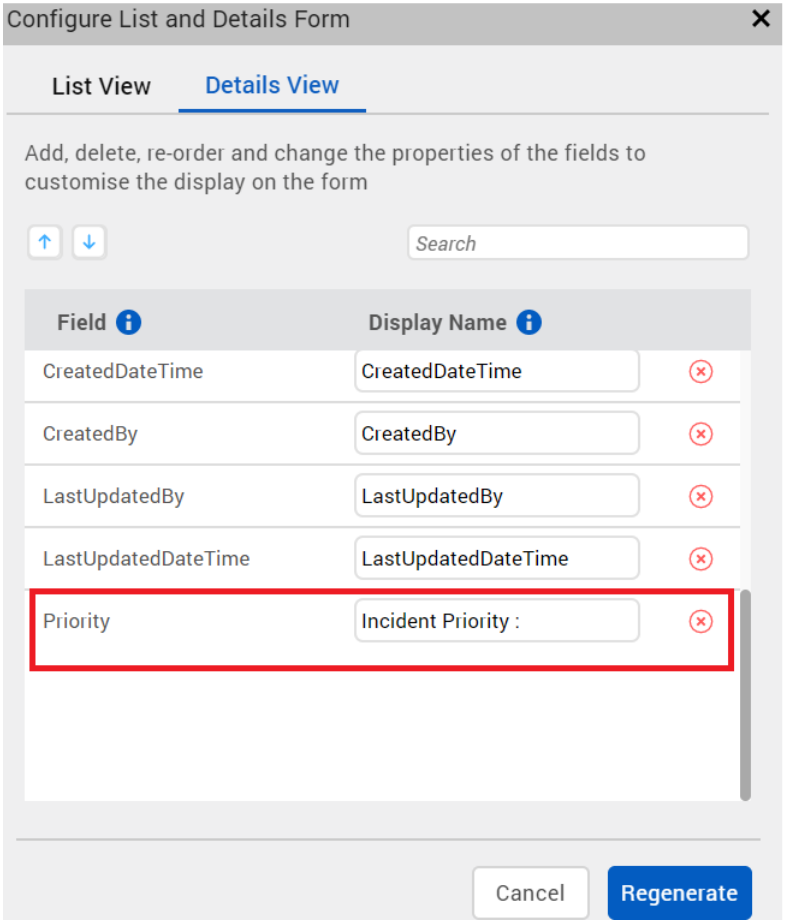
Delete Operation





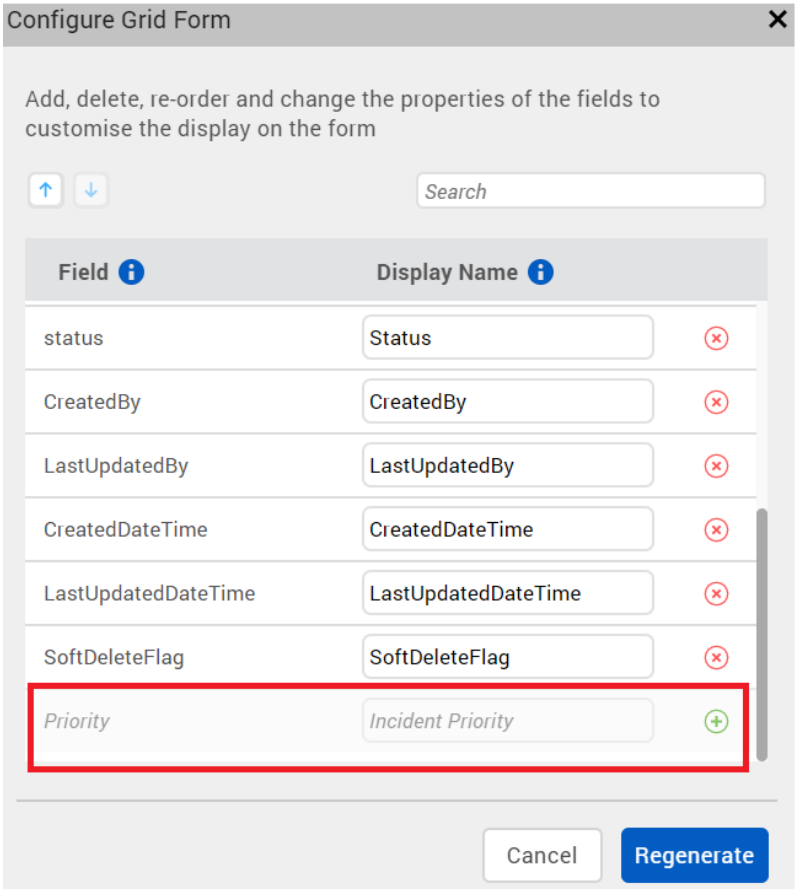
Add Operation



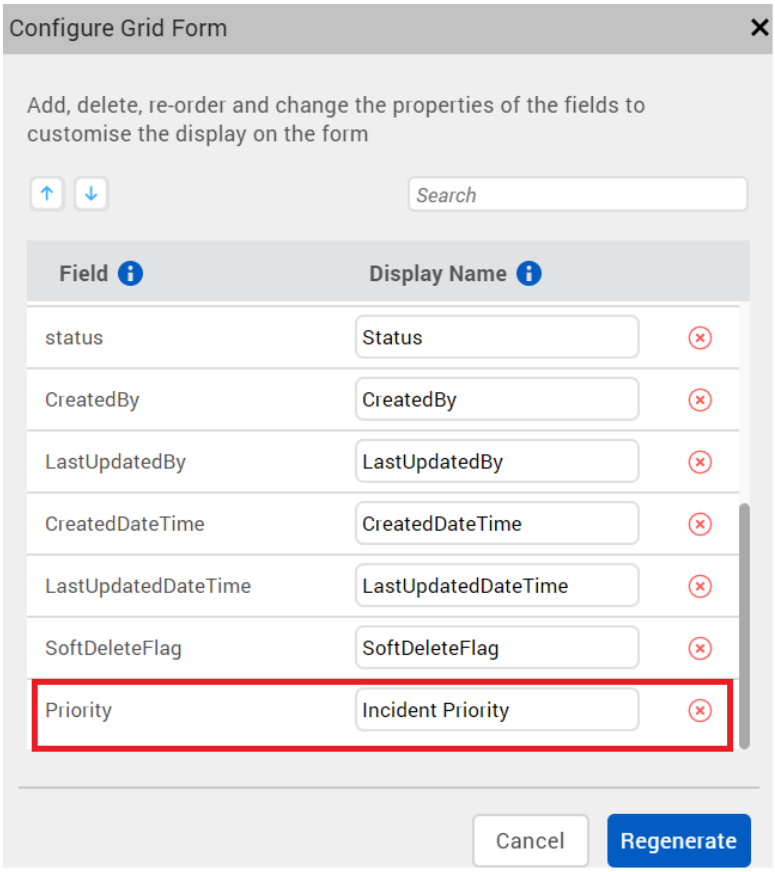


In Grid Forms

Delete Operation



Add Operation



After you have performed the necessary operations for the **Priority** field in the **Configure Entry Form** window, click **Regenerate**. The dynamic properties of the **Priority** field are saved, and the updated **frmCreate** form is generated on the Project Canvas.

Set Data for the Segment in a Component

To map data for the Segment in a component for different channels, follow these steps:

1. In Kony Visualizer, create a component.
2. Drag and drop a **get** method of an Integration or Object service (for example, *getCountries*) from the Data & Services panel onto the component that you just created. A dialog box appears asking you to select what type of form you want to generate: **List Using Response** or **Details Using Response**.

3. Select the **List Using Response** option. A Segment (for example, *seggetCountries*) is added to the component, a list form is generated on the Project Canvas, and the mappings between various elements are auto-generated in Mapping Editor.

4. On the Project Canvas, change the channel of the component from Mobile to Tablet, or vice-versa. For example, if you had been using the **iOS Mobile: Native** channel, change it to **iOS Tablet: Native**.

5. On Project Explorer, select the Segment of the component, and go to **Properties** panel > **Segment**.

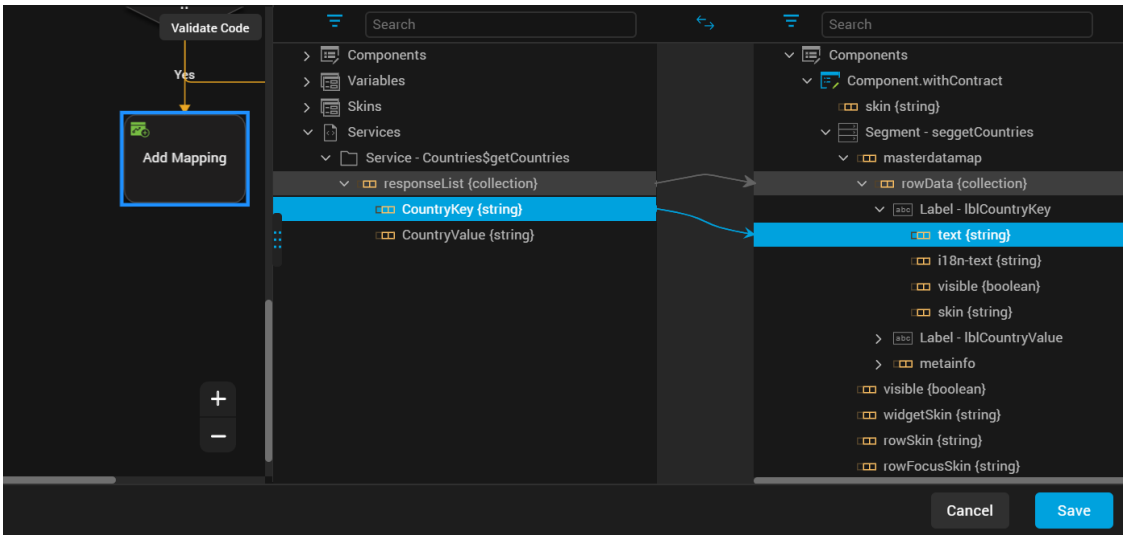
6. For the **Row Template** list box, select the **SampleRowTemplate** option.

7. On Project Explorer, select the parent FlexContainer of the component, and then go to **Properties** panel > **Action**.

8. Under the **Component** section, for the **onMapping** Event, click **Edit**. The **Action Editor** window appears, with **Diagram View** open by default.

9. In the flow diagram, click the **Add Mapping** action. Mapping Editor opens on the right pane of Action Editor.

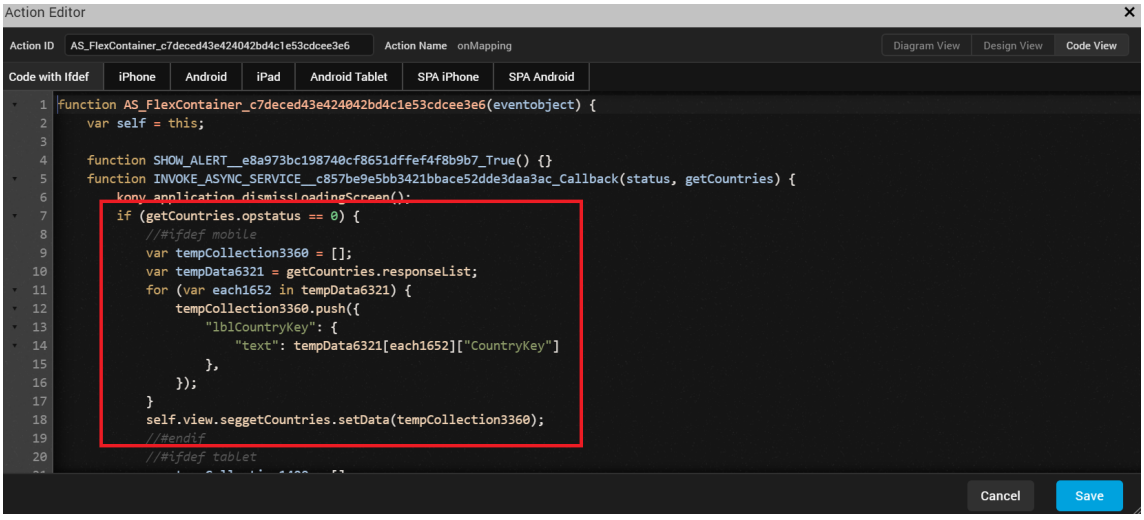
10. Map the required service parameters to the Segment **rowData** and associated rowData elements, as shown here.



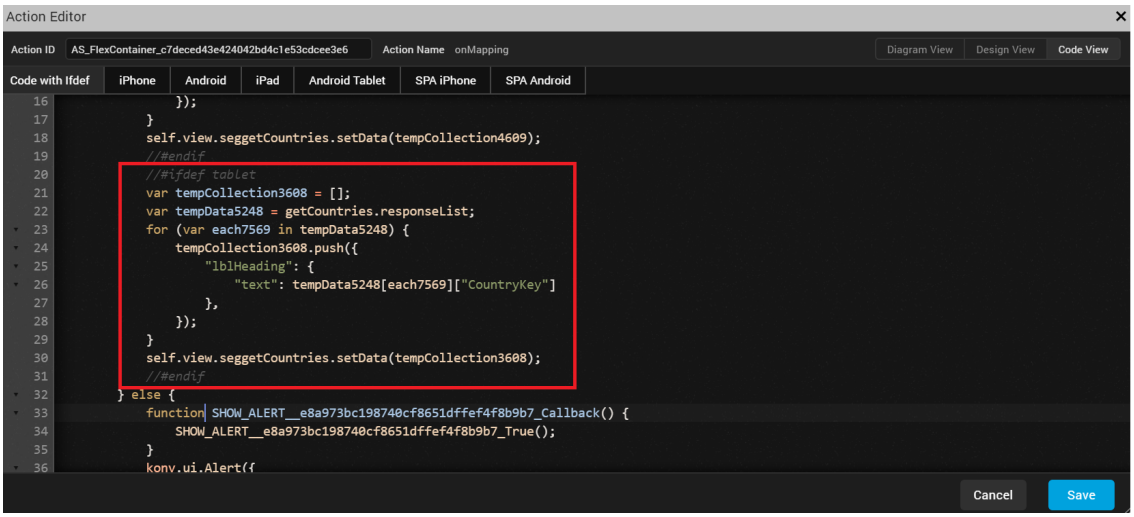
11. Click **Save**.

12. Click **Code View** to see the code details of the data mappings.

For Mobile



For Tablet



13. Click **Save**. You have successfully mapped service parameters to the Row Data widgets of the Segment in a component for the Mobile and Tablet channels.

14. If you build the project for Mobile channel, only the Mobile-related code will be used during run time. Similarly, if you build the project for Tablet channel, only the Tablet-related code will be used during run time.

Send Data between Two Forms

Previously, if you wanted to send data from one form to another, you had to save the data in a global variable first and then you would be able to use it. From Kony Visualizer V8 SP4, you can send data between two forms by using Action Editor. You can enter any data, such as an object or a variable, in the Action Editor UI and then send it to another form.

To send data between two forms, follow these steps:

1. In Kony Visualizer, create a new project.

2. Create two new forms; for example, say *frm1* and *frm2*.

3. Add a button to frm1.

4. Select the button, and then go to **Properties > Action**.

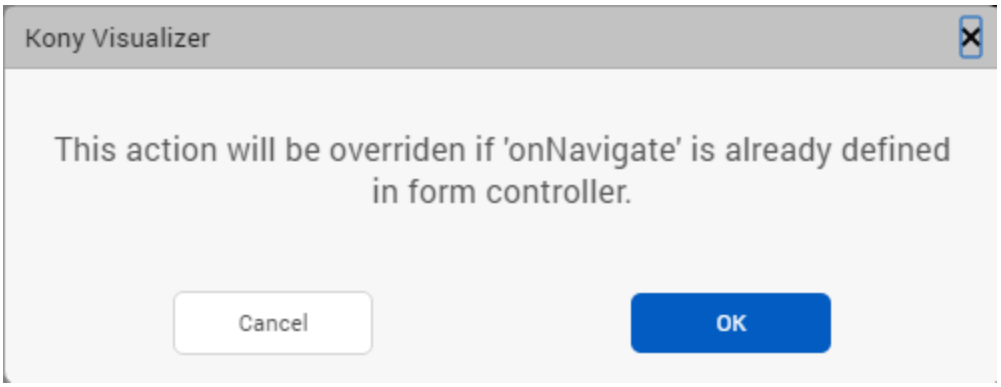
5. For the onClick Event, click **Edit**. The Action Editor appears.
6. From the list of actions on the left, scroll to the **Navigation** section, and then click **Navigate to Form**. The navigation action is added to the button.
7. In the lower section of the Action Editor, select **frm2**.

> ***Note:*** Alternatively, you can click **Create New** to create a new form; say, **frm3**. And set the onClick navigation for the button to **frm3**.

8. Select the **Pass data with navigation** check box. The form **frm1** and all of its widgets are listed in a nested structure along with defined local Variables, if any.
9. You can select the required widget data properties or local variables of **frm1** to pass data on navigation from **frm2**. As you select multiple widget properties, the Navigation Object is constructed as key-value pairs.
10. Click **View Code** to view the default code for the Navigation Object that has been constructed. Alternatively, you can select the Custom Input check box to create your customized Navigation Object or navigation flow data.
11. Click **Generate Code** to verify the data that is sent to **frm2**.

To use the data that is received from frm1, follow these steps:

1. Click **frm2**, and then go to **Properties > Action**.
2. For the onNavigate Event, click **Edit**. A confirmation dialog box appears stating that if you have already defined the onNavigate Event in FormController, the current action will not be respected.



3. Click **Ok**. The Action Editor appears.
4. From the list of actions on the left, scroll to the **Functions** section, and then click **Add Snippet**. The code snippet is added to **frm2**.
5. Click **Generate Code**. For this action sequence, you can access the data from the **eventobject** argument. This is the same argument that was passed from **frm1**.
6. Click **Close**.

Associated Data & Services Panel Features

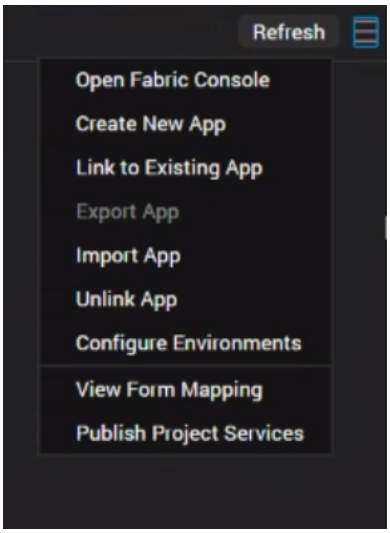
Kony Fabric Node on the Data & Services Panel

From Kony Visualizer V9, the **Kony Fabric node** and associated actions have been shifted from the Project Explorer to the Data & Services panel. This feature has been introduced to enable a unified and enhanced user experience of using Kony Fabric data and services from within Kony Visualizer.

Click the hamburger menu icon  at the upper-right corner of the Data & Services panel. From the list of options that appears, you can perform the following Kony Fabric-related actions:

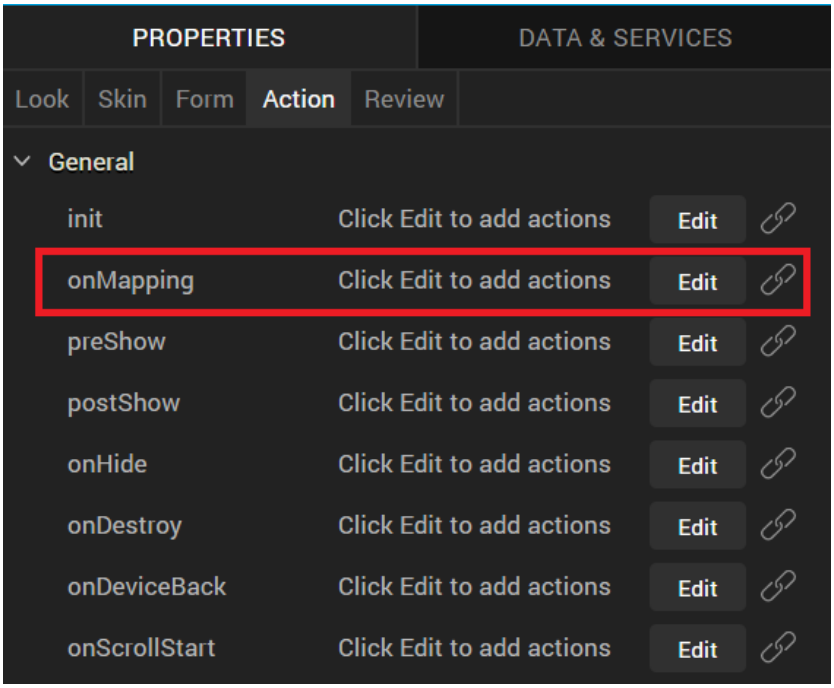
- Open Kony Fabric Console
- Create a Kony Fabric app
- Link your Kony Visualizer project to an existing Kony Fabric app

- Export an app from Kony Visualizer
- Import an app to Kony Visualizer
- Unlink a Kony Fabric app from your Kony Visualizer project
- Configure Kony Fabric environments
- View form mapping of linked Project Services
- Publish your Kony Visualizerapp to Kony Fabric



Mapping Event

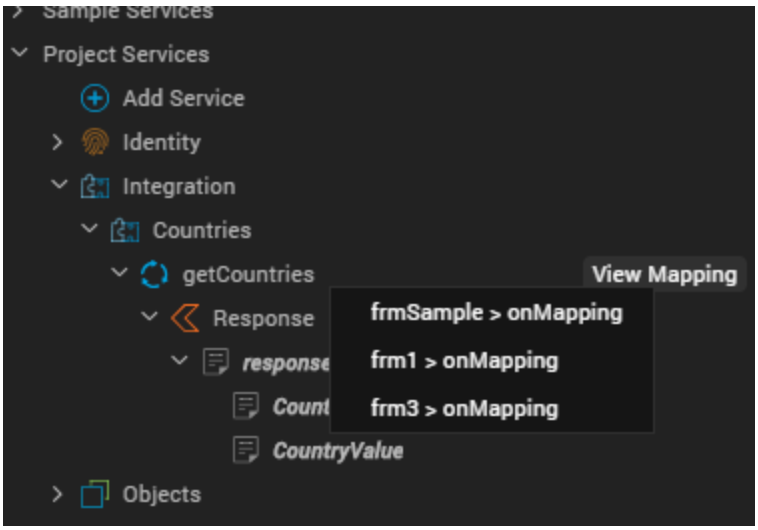
Mapping event is available in the Mapping Editor. Service invocations dropped on a form or a FlexContainer in a canvas are mapped to the mapping event .



> ***Note:*** If you add code to the preshow event dynamically, the new code added will override the mapping event.

View Mapping

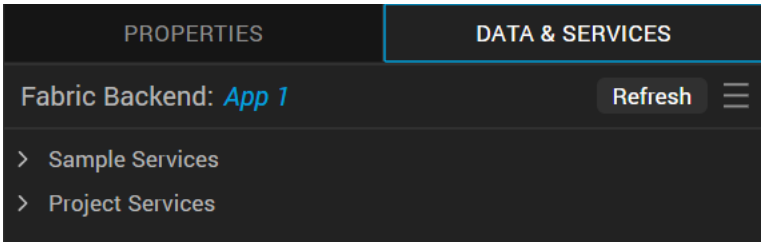
When **View Mapping** is selected for a service on the data panel, the associated action is displayed.



If more than one action sequence is associated to a service, a fly-out menu is displayed with a drop-down list. You can select the Mapping Action Sequence you want to view. This opens the Action Editor. For more information, refer Action Editor.

Manual Refresh

You can refresh the data in the Data & Services panel manually by using the Refresh button. When you click **Refresh**, the Data & Services panel fetches the latest data from your associated Kony Fabric application. You can use this Refresh button to update any changes you made to your Kony Fabric service by using the Kony Fabric console, from within Kony Visualizer.



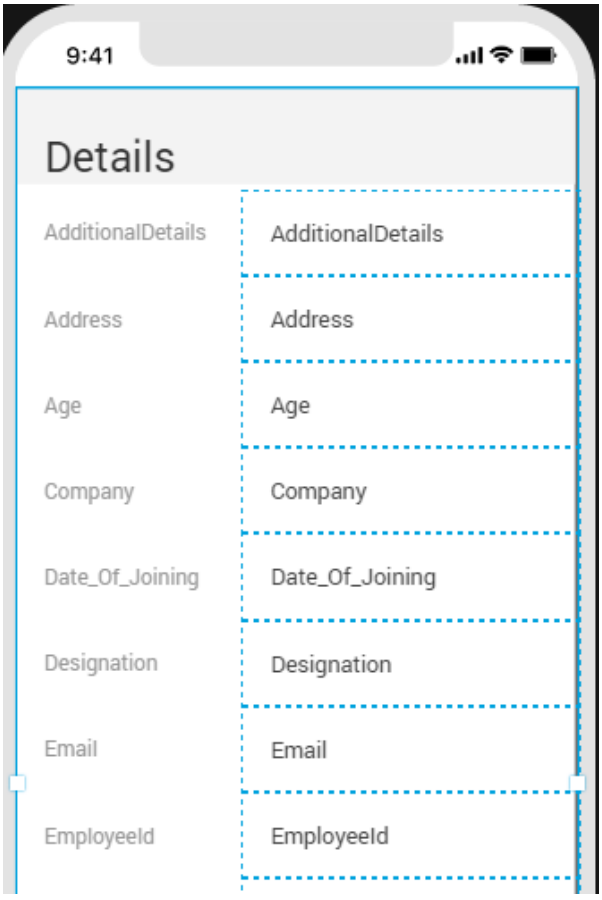
Service Invocation

When a network call is made to invoke an async service, the action sequence appears in the Data & Services panel. For containers, when you drag and a drop an operation or the response from the data panel, the async service invocation is added to the mapping event. Response parameters are auto-generated on the form. You can also invoke an Identity service by using the drag-and-drop feature onto a login form. Network actions are used to generate the service invocation.

Highlighted View of Mapped Widgets

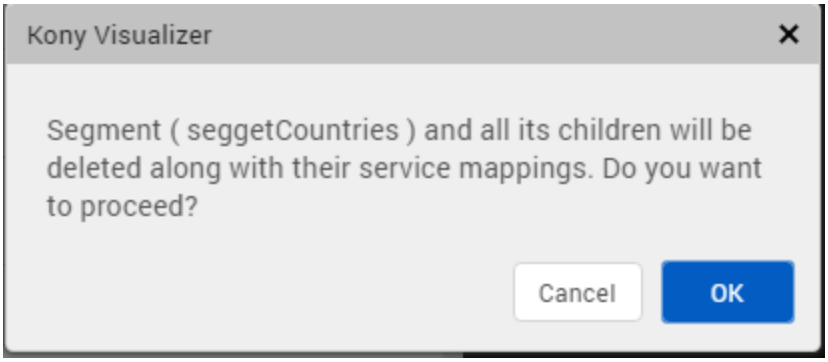
When you select a service or its associated operation from the Data & Services panel, the mapped UI elements are highlighted on the form.

When the Data & Services panel goes out of focus or if you select another form, the highlight effect disappears.



Delete Mapped Widgets

You can delete any mapped widget that is mapped to services by right-clicking the widget, and then selecting Delete. Visualizer displays a confirmation popup before you can proceed with the widget deletion. The display of this alert helps prevent any accidental deletions and service failures. Click Ok to delete the widget and all its children widgets along with the service mappings.



Action Sequences

The following action sequences are auto-generated for responses.

Loading Indicator

When enabled, the loading progress of a service call is displayed until the response is completed. You also have the option to not display (dismiss) the loading indicator.

Data Type Conversion

When you map a non-matching data type, a popup appears alerting you of the same. The non-matching data type is then implicitly converted before being mapped.

> ***Note:*** The Calendar Date data type is not converted. But if the *date* property of a Calendar widget is mapped to the *Date* field of an Object service, the date is converted to ISO format at run time and is sent to the back end. When this date is read during run time, the ISO format of the date is converted back to the DD-MM-YYYY format.

> ***Note:*** You cannot map Boolean and Date fields to widget positional properties such as left, top, right, and bottom.

If you want to view the data type conversion, click **Generate Code** in the Action Editor.

The following data types are converted:

- Boolean
- String
- Number (can be further converted as int and float)

Status Check

The Data & Services panel auto-generates actions to perform a service status check and map the widgets on the success of the service call.

You can also define your own IF condition blocks by using the Action Editor.

Search Engine Optimization for SPA

Applies to *Kony Visualizer Classic*.

Kony Visualizer supports search engine optimization (SEO) for single-page applications (SPAs) and Desktop web apps, making their cached web pages discoverable by search bots and spiders, such as Googlebot.

Implementing SEO in an SPA or Desktop web app involves the following tasks. Once you have completed them, building the app optimizes it for search engines.

Install PhantomJS

Create an SEO Configuration File

Create Sitemap.xml

Create SEO Functions

Call the Search Initialization as an App Service

Set the SEO Data Ready Flag

Enable SEO in Kony Visualizer

Install PhantomJS

Based on WebKit, PhantomJS is a headless web page browser that pre-renders the HTML of your SPA or Desktop web app so that its pages are accessible to search bots. To download PhantomJS, see the Download page on the PhantomJS web site.

> ***Note:*** After installing PhantomJS, you may need to ensure that its installation location is added to your computer's PATH environment variable.

Create an SEO Configuration File

This .json file provides properties to PhantomJS for pre-rendering an SPA's HTML. If both the SEO configuration file and sitemap.xml define values for properties of the same name, the values defined in the SEO configuration file to precedence over those in sitemap.xml.

You specify the location of the SEO configuration file in the project settings, as described in Enable SEO in Kony Visualizer.

> ***Important:*** The extension of the SEO configuration file must be .json. For example: `seo.json`

SEO Configuration File Properties

The properties that can be defined in the SEO configuration file are as follows:

Property	Description
`baseurl`	Provides you the option of changing the domain name or IP address of the published server specified in sitemap.xml.
`port`	Provides you the option of changing the port of the published server specified in sitemap.xml.

`channel`	Defines the channel(s) for which you want to run SEO. Possible values are:
	`mobile`
	`desktopweb`
	`both`

`maxiterations`	Each page is fetched multiple times until the data ready flag is set, otherwise, the build can fail. The default value is 5.
`contextpath`	Provides you the option of changing the contextpath specified in sitemap.xml.

SEO Configuration File Example

The following is an example of an SEO Configuration file.

```
{
    "channel": "mobile",
    "baseurl": "localhost",
    "port": "8888"
}
```

Create Sitemap.xml

Using the Sitemap Protocol, the sitemap for an SPA is an XML file that lists the SPA's various URLs, providing search bots a means of cataloging and listing them in search results.

In addition to URLs, a sitemap can indicate how often the content of a given URL changes, when it was last updated, and its importance relative to other URLs in the app.

A sitemap file always has the file name `sitemap.xml`.

You specify the location of the sitemap file in the project settings, as described in Enable SEO in Kony Visualizer.

The following is an example of a sitemap file.

```
< urlset xmlns = "http://www.sitemaps.org/schemas/sitemap/0.9" > < url
>             < loc > http: //localhost:8888/test/p?title=Home</loc>
```

```
        <changefreq>daily</changefreq>
   <priority>1.0</priority>        </url>
<url>           <loc>http://localhost:8888/test/p?title=products</loc>
          <changefreq>daily</changefreq>
   <priority>1.0</priority>        </url>
        <url>           <loc>http://localhost:8888/test/p?title=categ
ories</loc>           <changefreq>daily</changefreq>
   <priority>1.0</priority>        </url>                 <url>
   <loc>http://localhost:8888/test/p?title=list</loc>
   <changefreq>daily</changefreq>           <priority>1.0</priority>
     </url></urlset>
```

Create SEO Functions

Two functions are essential for optimizing the content of your SPA or Desktop web app for search engines. The first initializes the app's data for searching, and the second sets a flag indicating that the data is ready to be searched. Additionally, you can use a third function to clear the flag indicating that the data is ready to be searched.

These functions reside in a JavaScript file in the Modules folder of your project.

You call the initialization function at the Channel level as an App Service, and you set the Data Ready flag once all service calls are completed. Instructions for doing so are provided in the next two procedures.

The following is an example of a JavaScript file containing these two functions.

```
function initDeeplinkforSEO(eventObj) {
    var param = eventObj.launchparams;
    if (param.title == "home")      return frmHome;
    if (param.title == "products")      return frm01;
    if (param.title == "categories")      return frm02;
    if (param.title == "list")      return frm03;
}
```

```
function seoReady() {    //call this method after all service calls
are completed   return kony.application.setSEOdataReadyFlag;}
```

Call the Search Initialization as an App Service

Once you have created a function that initializes the app's data for searching, you create an action sequence that invokes it as a part of the App Service function. The App Service is the first event invoked when an application is launched, and so the logic for presenting search data needs to be added to the App Service function in an action sequence.

To call the search initialization as an app service, do the following:

1. On the Project tab of the Project Explorer, click the channel for which you want to initialize search, either Desktop, Tablet, or Desktop. Doing so displays the **App Events** tab on the Properties pane.
2. On the **App Events** tab on the Properties pane, click the **Edit** button that corresponds with App Service. An action sequence opens in the Action Editor.
3. Scroll down in the Actions pane to the Functions section, and then click **Add Snippet**. A code snippet opens in the Code Editor.
4. In the Code Editor, invoke the function that initializes the app's data for searching. Using the earlier example, if the function name is initDeeplinkforSEO, the code snippet would be as follows:

   ```
   return initDeeplinkforSEO (eventobject);
   ```

5. Save the action sequence by pressing **Ctrl+S**.

Set the SEO Data Ready Flag

You indicate that the data of your app is ready to be optimized for search by invoking the second function that you created, which sets the Data Ready flag. This takes place at the form level of your project.

To set the Data Ready flag for a form, do the following:

1. On the Project tab of the Project Explorer, navigate to the first form in your app, and click it. Doing so opens it in the Visualizer Canvas.
2. On the Properties pane, click the **Action** tab, and under the General section, click the **Edit** button that corresponds with the postShow event. An action sequence opens in the Action Editor.
3. Scroll down in the Actions pane to the Functions section, and then click **Invoke Function**. From the **Function Name** drop-down list in the bottom pane of the Action Editor, click the name of the function you created that sets the Data Ready flag. Using the earlier example, if the function name that would be selected is `seoReady`.
4. Save the action sequence by pressing **Ctrl+S**.
5. Repeat this procedure for every form in the app.

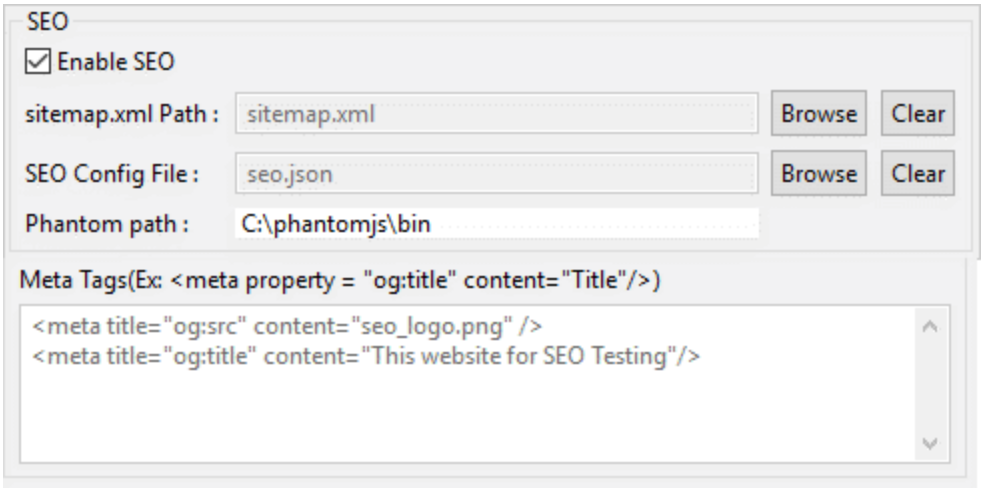
Enable SEO in Kony Visualizer

With your channels, app forms, and modules properly set up for optimizing your app for search engines, you need to enable SEO in the project settings of Kony Visualizer.

To enable SEO in Kony Visualizer, do the following:

1. On the **File** menu, click **Settings**, and then click the **Mobile Web** tab.
2. Configure the following settings, using the illustration below as a guide.
    - In the SEO section, click **Enable SEO**.
    - Click the **Browse** button corresponding to **sitemap.xml Path**, navigate to the location of your app's sitemap.xml file, click sitemap.xml, and then click **Open**.
    - Click the **Browse** button corresponding to **SEO Config File**, navigate to the location of your app's SEO configuration file, click it, and then click **Open**.
    - Using your computer's file explorer, navigate to the location of the executable for PhantomJS (i.e. `phantomjs.exe`), copy the folder path, and then in Kony Visualizer, paste it into the **Phantom path** text box of the **Mobile Web** tab.

- In the Meta Tags text box, enter meta tags for the web page. The content entered in this text box is added to the Meta tag under the head section of HTML for the associated Web app during its initial page launch.



3. Click **Finish**.
4. Build your app.

Publish a Kony Fabric App

Until an app is published, all its services and features are limited to just the Kony Visualizer development environment. To connect the app to live services, you have to publish the application to Kony Fabric. After you have signed in to Kony Fabric, you can select and publish your app to any of your cloud accounts. You can publish your app from the Kony Fabric Console by using the Publish tab.

Prerequisites

Before you can publish an app to Kony Fabric, you must meet the following prerequisites:

- Have a Kony Fabric account.
- Configure Kony Fabric in Kony Visualizer.
- Create a Kony Fabric app corresponding to the Kony Visualizer app that you are publishing.

For more information, refer Connecting to Services and Getting Started with Kony Fabric.

> ***Important:*** If your app contains deprecated widgets, it is possible that their skins may refer to the Helvetica font. If you are not explicitly using Helvetica in your app, you must verify your app's configuration and manually remove references to Helvetica before submitting it to the store.

This topic contains the following sections:

Directly Publish an App to Kony Fabric

From Kony Visualizer V8 SP4 FP 35 onwards, an option to directly publish your app to Kony Fabric has been provided. This feature is available on both Kony Visualizer and Kony Visualizer Classic.

To directly publish an app to Kony Fabric, follow these steps:

1. In Kony Visualizer, sign in to your Kony Cloud account. To do so, from the upper-right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Type your Kony Cloud email and password credentials, and then click **Sign in**. Kony Visualizer uses the configured Kony Fabric URL to sign in to Kony Fabric.

   > ***Note:*** In Kony Visualizer Classic, you can configure the Kony Fabric URL by going to: **Window > Preferences > Kony Visualizer > Kony Fabric**.

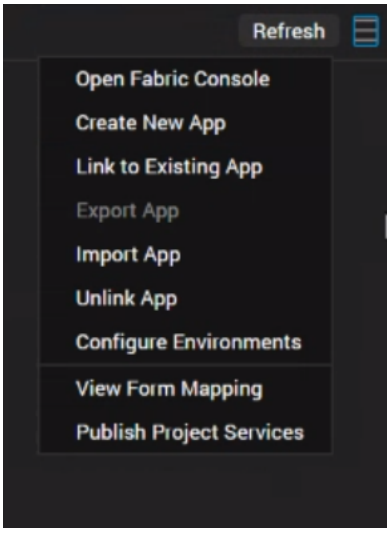
   > ***Important:*** If you are not able to get beyond the login page for the Kony Fabric Console, it could be because you set up Kony Fabricby using a **self-signed certificate**. The self-signed certificate allows you to install Kony Fabric, however, which Windows and Google Chrome do not trust the allow you to sign in. To resolve this issue, locate the certificate (you may need to contact your system administrator to do so), and then import it to the **Trusted Root Certification Authorities** folder of the Windows Certificate Store. For more information on how to import a certificate into the Windows Store, refer Import or export certificates and private keys on the Microsoft web site.

2. Select a default environment for Kony Fabric. To do so, on the **Project Explorer**, click **Project Settings**. Then, click the **Kony Fabric** tab. At the top of this tab, under Kony Fabric Environment, select an environment from the drop-down list. Click **Done**. If you do not see any environments listed, you need to create one. For more information, refer Environments in the Kony Fabric Console User Guide.

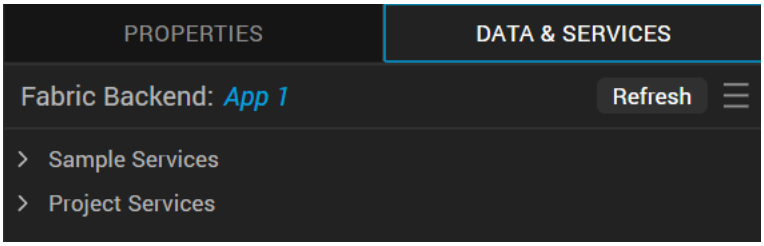
   > ***Note:*** If an environment contains an alias hostname, the alias is used in the generated App Service Document. The alias is used to make service calls from the client app.

3. To publish to Kony Fabric, your Kony Visualizer client app must be associated with a Kony Fabric app, which means that you need to either create a new Kony Fabric app or use an existing one. To do so, at the upper-right corner of the **Data & Services** panel, click the hamburger menu icon hamburger menu icon  , and then click either **Create New App** or **Link to Existing App**. From the Kony Fabric Application dialog box, select the app to which you

want to associate your Kony Visualizer app. For more information about how to create a new Kony Fabric app, refer How to Add Applications in the Kony Fabric Console User Guide.



Once you link your Kony Visualizer client app with a Kony Fabric app, the name of the linked Kony Fabric app is displayed beside **Fabric Backend** on the **Data & Services** panel. Here, *App 1* is the associated Kony Fabric app.



4. Right-click **Kony Fabric** for the linked app, and then click **Publish Project Services**. The app is successfully published to Kony Fabric.

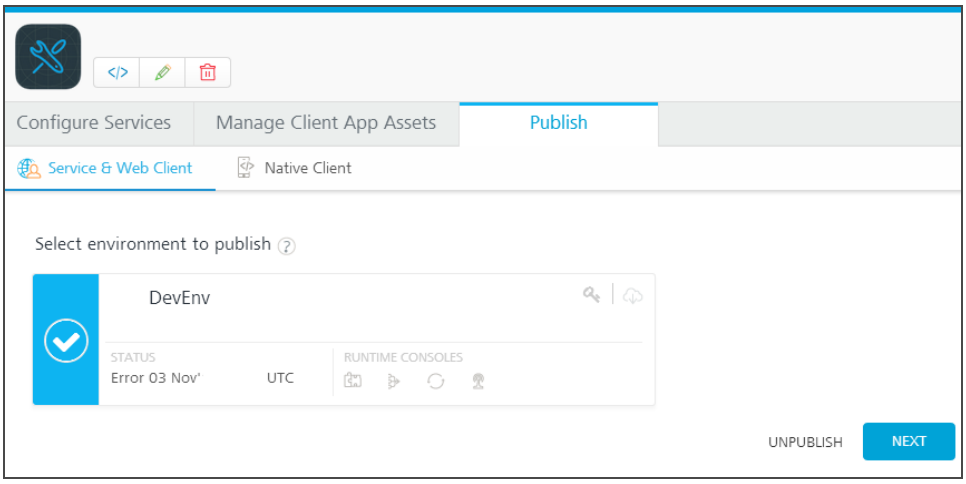
> ***Note:*** If you have not done so earlier, a dialog box appears asking you to select a valid Kony Fabric environment.

Publish the App from Kony Fabric

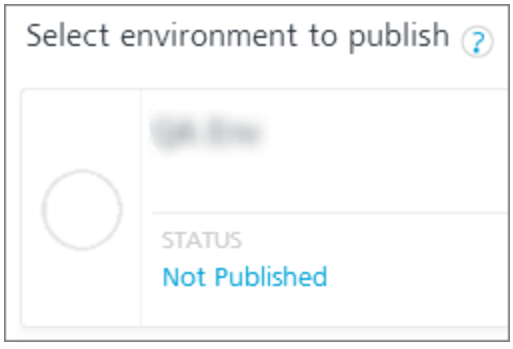
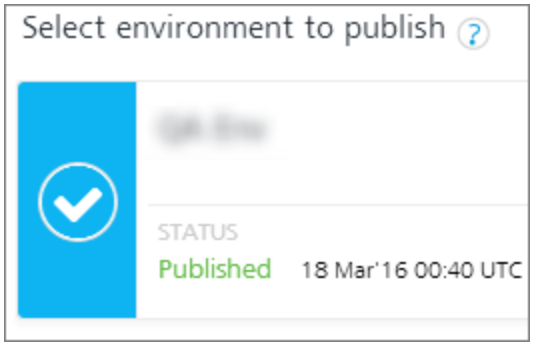
After you have published your app to Kony Fabric from Kony Visualizer, you can publish the app from Kony Fabric. This feature is applicable only in Kony Visualizer Classic.

To publish your app from Kony Fabric, follow these steps:

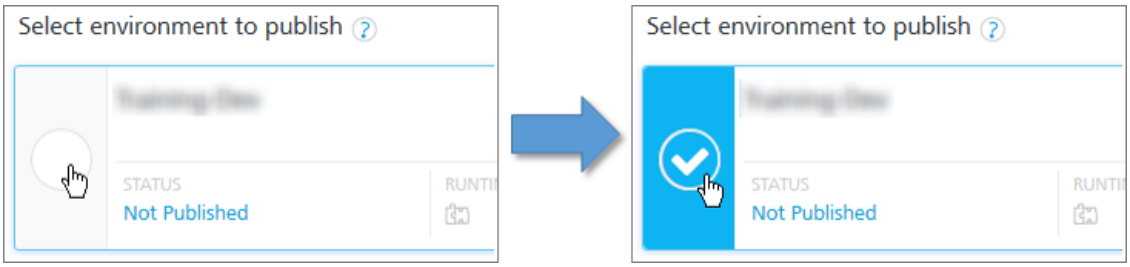
1. Once you have selected a Kony Fabric app to which you want to bind your Kony Visualizer app, the Kony Fabric Console opens. On the Kony Fabric console, click the **Publish** tab.



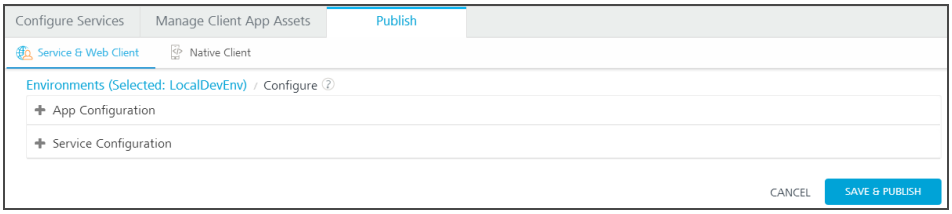
The status of the environment to which you want to publish the client app indicates whether the Kony Fabric app has been published.

Status when Not Published	Status when Published
	

2. If the Kony Fabric app to which you have bound the client app has not yet been published, select the app for publish.



3. Click **Next**. The **Environments** page appears.



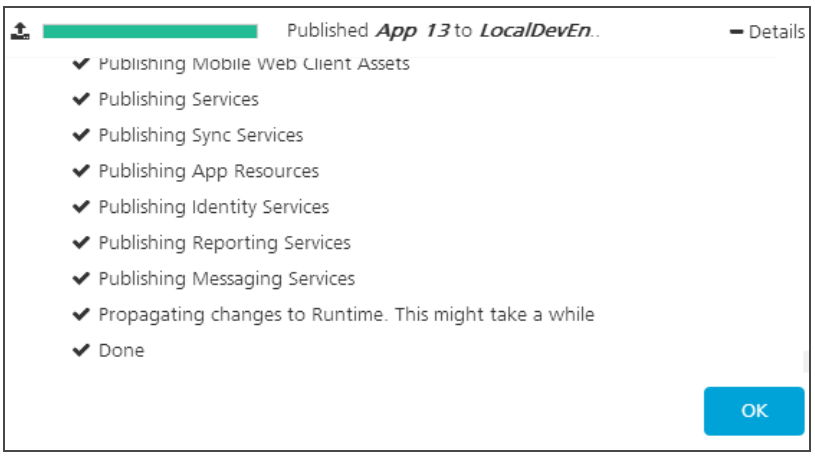
4. The **Configure** page displays options to change the configuration of your Services. To make any changes, click **+ Service Configuration**., and then click any of the blue-text fields.

> ***Note:*** If your app is an upload of web app (such as SPA, Desktop Web/Responsive Web, or Progressive Web App) binary , the Configure page displays a drop-down list of options specific to the web client environment.

5. If available, you can reconfigure the application-level settings and service configuration settings from this page .

6. Click **Save and Publish**. The app and any associated services are published to the selected environment.

> ***Note:*** If you are building a Kony Visualizer web app (such as SPA, Desktop Web/Responsive Web, or Progressive Web App), you must first define and publish any required Kony Fabric services. After you build any web application and upload its binaries to Kony Fabric, you cannot publish a service to it.
> If you encounter an error while publishing the Kony Fabric, try publishing again.



7. To close the Kony Fabric Console and return to Kony Visualizer, from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Visualizer icon  . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric apps and services.
8. Launch the app. To do so, on the **Product** menu, navigate to **Run As**, and then select an emulator to run the app on.

Environment List in Visualizer

Prior to Visualizer V9 release, it lists only the environments to which you have full access. From Visualizer V9 onwards, you can view the list of all environments associated with your Fabric account. This list includes the environment with both full and read-only access.

The different types of access that a cloud account admin can provide, are:

- **Full Access**: When your cloud account has **Full Access**, you can build your application and publish your services to Kony Fabric. The environment with full access is listed in Visualizer.
- **No Access**: When your cloud account has **No Access**, you cannot build and publish your application to Kony Fabric. The environment with no access is not listed in Visualizer.
- **Custom**: When your cloud account has **Custom** access, your cloud account is provided with permissions depending on the features/components. These features/components include

Server, Engagement, etc. You must have one of the following feature level permissions to build your application and see these environments listed in Visualizer:

- Build Client App
- Server

The following table shows the actions you can perform for each access.

User Access Type	Environment Access Type	View Environments in Visualizer	Build native Application	Build Web Application	Publish the Services
Full Access	Full Access	Yes	Yes	Yes	Yes
No Access	No Access	No	No	No	No
Custom	Build Client App	Yes	Yes	No	No
Custom	Server	Yes	Yes	Yes	Yes

For more information about the different environment permissions of your cloud account, click here.

Apply Application Security

Applies to *Kony Visualizer Classic*.

Kony protects all applications using methods like design time security, critical business logic security, and on-device encryption. All the applications developed through Kony are compliant with Payment Card Industry Data Security Standard (PCI DSS), HIPAA/HITECH, SOC2 Type II, ISO 270001:2013, and address OWASP top 10 mobile vulnerabilities, support Single Sign-On (SSO), and Multi-Factor Authentication (MFA), Federal Information Processing System (FIPS) 140-2. Standards

followed by Kony adhere to Microsoft Security Development Lifecycle (SDLC) for product development, Open Web Application Security Project (OWASP) for secure coding and testing, Web Application Security Consortium (WASC) guidelines for threat modeling, coding, and testing, and follow Application Security Verification Standard (ASVS) for testing security controls.

For enhanced security, Kony provides strong application protection solutions by using anti-tampering mechanisms, White Box Cryptography (WBC), Cryptography and Encryption APIs, and also can secure network communications. Following are two methods you can use to enhance security in your application.

- **Kony APIs** - You can make your application more secure by protecting the data in your application using the security APIs provided by Kony. Data can be encrypted using Cryptography APIs, offline data can be encrypted and stored using SQL Database Encryption APIs, and network communications can be secured using two-way SSL/Mutual Authentication (SSL Pinning + Client Authentication).
- **Protected Mode option** - Applications built in Kony Visualizer can use the additional security enhancements by building the application in the *Protected Mode*. Kony Platform code for iOS and Android is equipped with mechanisms that can protect your application by detecting attacks like tampering, swizzling, debugging, jail breaking (iOS), rooting (Android), and information disclosure. Additional security mechanisms are provided through the use of White Box Cryptography to protect application business logic and source code. Application reacts to the attack by exiting upon detecting attacks to prevent further attempts.

This topic covers the following:

Build Runtime Security in the Application

You can build runtime security in your application using the following Kony features and API's:

- Data can be encrypted using Cryptography APIs.
- Offline data can be encrypted and stored in SQL Database Encryption APIs.
- Network communications can be secured using **Two-way SSL/Mutual Authentication (SSL Pinning + Client Authentication)** which defends against man-in-the-middle attacks by authenticating both the client and server to each other.
    - **SSL Pinning** - This extra feature authenticates the server to which the application is communicating. For more information, see Allow Self-Signed Certificates (Android, iOS).
    - **Client Authentication** - This extra security feature uses a certificate to authenticate a client to a server.

SSL Pinning - Windows Limitation

For Windows devices, when SSL pinning is implemented, and the HTTPRequest initiates, if a Man in the Middle (MitM) attack accesses the HttpRequest (for example, fiddler), the HttpRequest goes to the server through the MitM and comes with a response. However, the client does not receive the response as the MitM attack corrupts the certificate of the response during the communication process. This applies to all windows channels.

In case of Android and iOS platforms, if there is a MitM attach, the HttpRequest aborts.

Protect the Application Binaries

In Kony Visualizer, *Protected Mode* option enables several security features that secure the binary at build time by including multiple self-protection security mechanisms. To use the option, enable the **Protected Mode** check box in the **Project Settings** dialog. If an application attack is observed, the security mechanism exits the application.

In this section, you will learn about:

1. Protection Mechanisms Provided by Kony
2. RSA Key Pair Generation, Encryption, and Usage
3. Configure Project Settings

Protection Mechanisms Provided by Kony

Kony provides the following application and code-level mechanisms to protect your application:

1. **Anti-tamper Protection** - Following are the application self-protection security mechanisms used by Kony that react by exiting application on detecting an attack:
    - **Tamper Protection** - Fights against application compromise by detecting modifications in the application.
    - **Jailbreak / Root detection** - Resists runtime attacks by preventing app from running on a rooted or jailbroken device. For more information on, click here.
    - **Swizzling detection** - Prevents the abuse/misuse of the swizzling feature to override methods at runtime. This attack is specific to Objective C in iOS.
    - **Anti-debugging** - Prevents debugging of a production application to prevent attackers from analyzing the application at runtime.
2. **Protecting Cryptographic Keys using White Box Cryptography (WBC)** - Cryptographic keys are critical to securing systems such as applications and communications, and therefore must be protected at all times. Kony provides powerful secure cryptographic capability beyond the native operating system's capabilities. Kony's encryption and decryption uses a secure process known as White Box Cryptography to perform encryption and decryption while keeping the keys safe. The keys are never present in static form or in memory at runtime. WBC is a secure implementation of cryptographic algorithms in a system that employs cryptographic algorithm and keys. Strong algorithms are used for encryption and decryption, insecure, and deprecated algorithms are not used.

RSA Key Pair Generation, Encryption, and Usage

Prerequisites

For OpenSSL command to work, for the Windows environment, you can use a couple of different third-party tools, such as Git Bash, which is available here, and Cygwin, which is available here.

To generate, encrypt, and use the RSA key pair, follow these steps:

1. Open a terminal (Git Bash or Cygwin terminal in Windows) and type **openssl**.
2. Generate RSA public/private key pair using OpenSSL.
    a. Generate a 2048-bit RSA key using this command.

    ```
    openssl genrsa -out private_key.pem 2048
    ```

    b. Extract public key from RSA key pair using this command.

    ```
    openssl rsa -pubout -in private_key.pem -out public_key.pem
    ```

    c. View the private key using this command.
    `openssl rsa -text -in private_key.pem`

    d. To use private keys use the following commands.

        i. `less private_key.pem` to verify that it starts with a `-----BEGIN RSA PRIVATE KEY-----`.

        ii. `less public_key.pem` to verify that it starts with a `-----BEGIN PUBLIC KEY––`.

3. Send your public key and Kony Visualizer version to licensing@kony.com. This step is applicable for Android and iOS platforms.
    For the Responsive Web/SPA platform, you must raise a Kony customer service ticket and provide your public key and Kony Visualizer version details in the ticket.

    > ***Important:*** Public key must not be shared with anyone except Kony.

4. For Android and iOS platforms, Kony's security team validates the details and encrypts your public key.
    For the Responsive Web/SPA platform, Kony's security team validates the information and

shares the unique **clientID** and **clientSecret** in the same customer service ticket that you had raised. You must then use these details to create a postbuild task.

5. Kony's security team then returns the encrypted public key to you through email.
    a. For iOS, Kony provides a set of `fin` keys along with the public key to protect iOS applications.

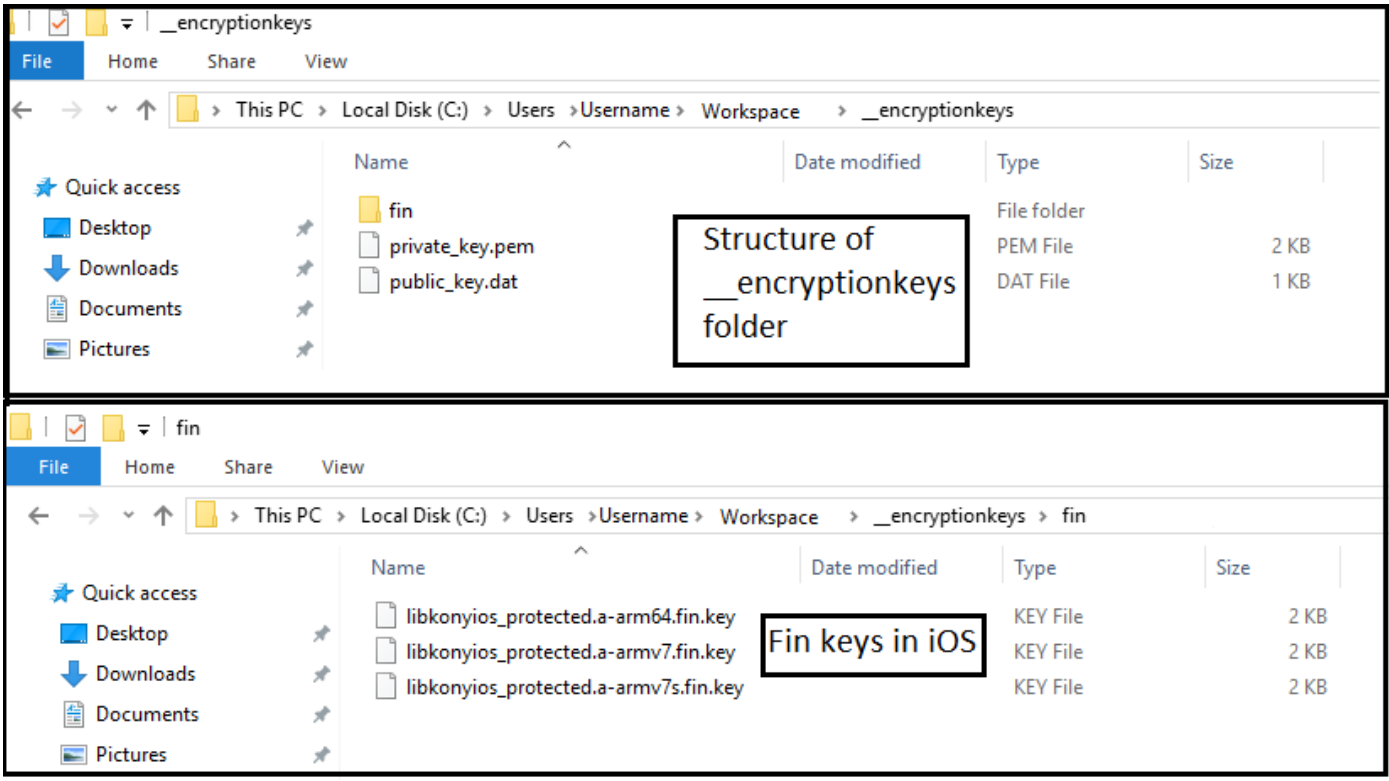
> ***Note:*** Starting with Kony Visualizer V8 SP3, `fin` keys are not applicable. Kony will not provide the `fin` keys folder if your Visualizer version is V8 SP3 or later.

> ***Important:*** You must not share your private key with anyone including Kony. In case of a key compromise for the Android and iOS platforms, generate a new set of keys and send the public key to licensing@kony.com. If a key compromise occurs for the Responsive Web/SPA platform, generate a new set of keys and send the public key via a Kony customer service ticket.

6. Navigate to your Kony workspace and create a **__encryptionkeys** folder.
7. Place the following keys received from Kony in **__encryptionkeys** folder.
    a. Your private key. The private key must be named as `private_key.pem`.
    b. Encrypted public keys provided by Kony.

> ***Note:*** Starting with Kony Visualizer V8 SP3, `fin` keys are not applicable. Once you have updated your Visualizer version to V8 SP3 or later, you can delete the `fin` keys folder from your **__encryptionkeys** folder.

    c. For iOS, the `fin` keys provided by Kony. These keys are provided to protect iOS applications. `fin` keys are applicable until the Kony Visualizer V8 SP2 version.

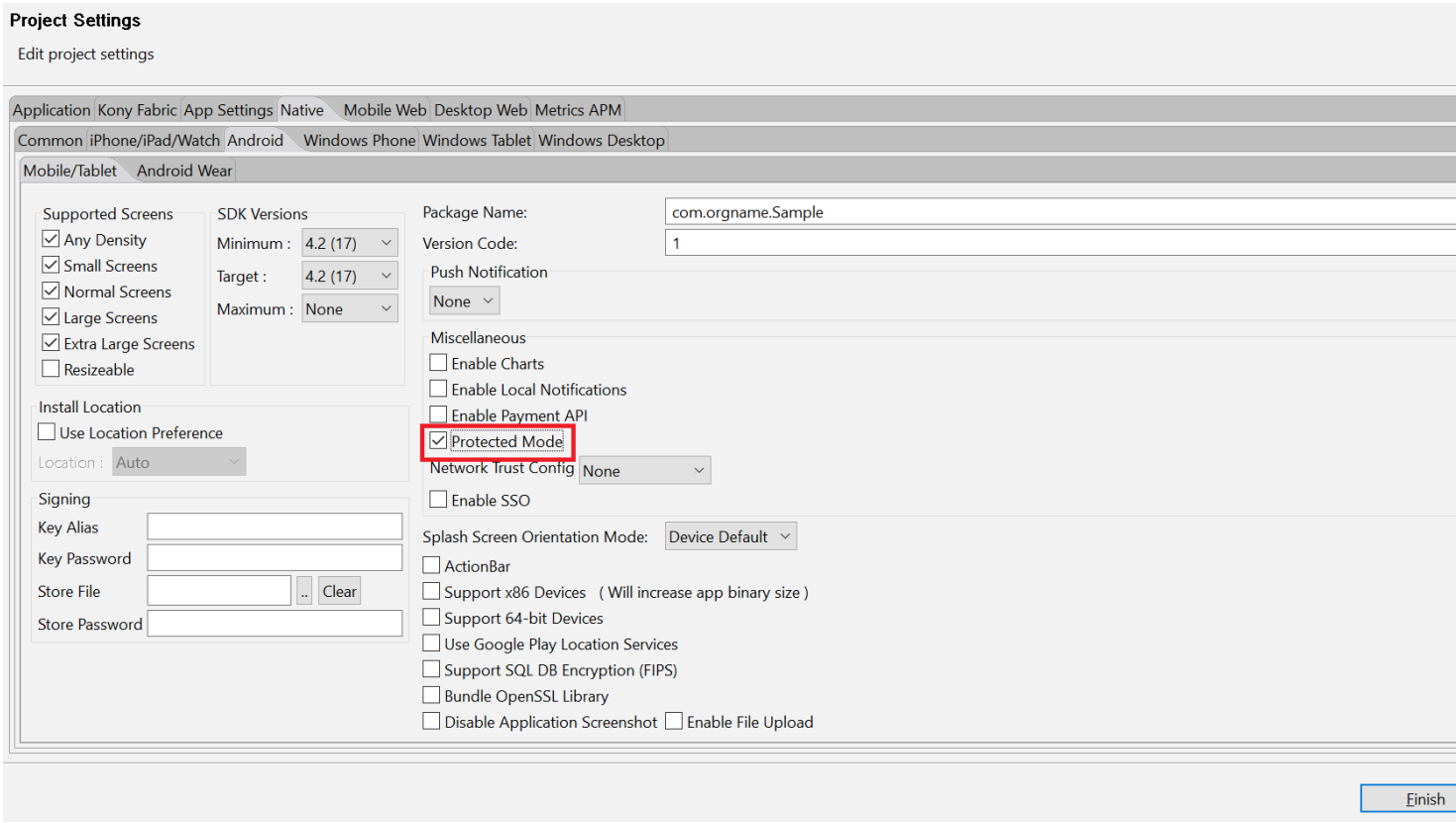


Configure Project Settings in Kony Visualizer

To enable Protected Mode in Project Settings, follow these steps:

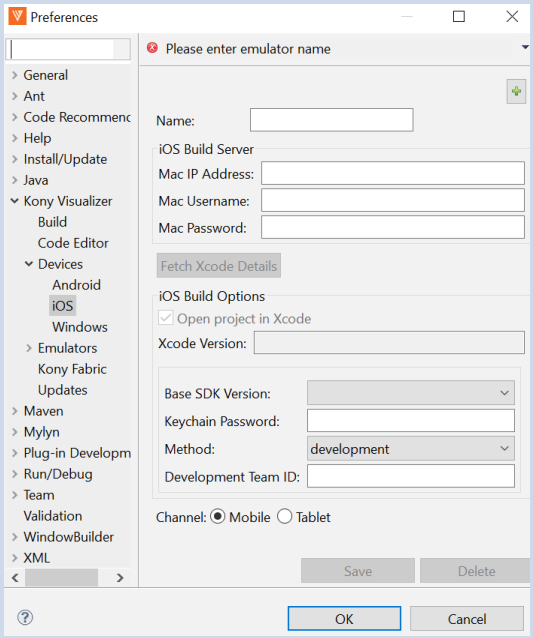
1. In Kony Visualizer, click **Project Settings**.
2. Go to **Native > iPhone/iPad/Watch** or **Android** tab.
3. Select the **Protected Mode** option.

> ***Note:*** The *Protected Mode* option works only if the application is built in *Release* mode.



4. Click **Finish**.

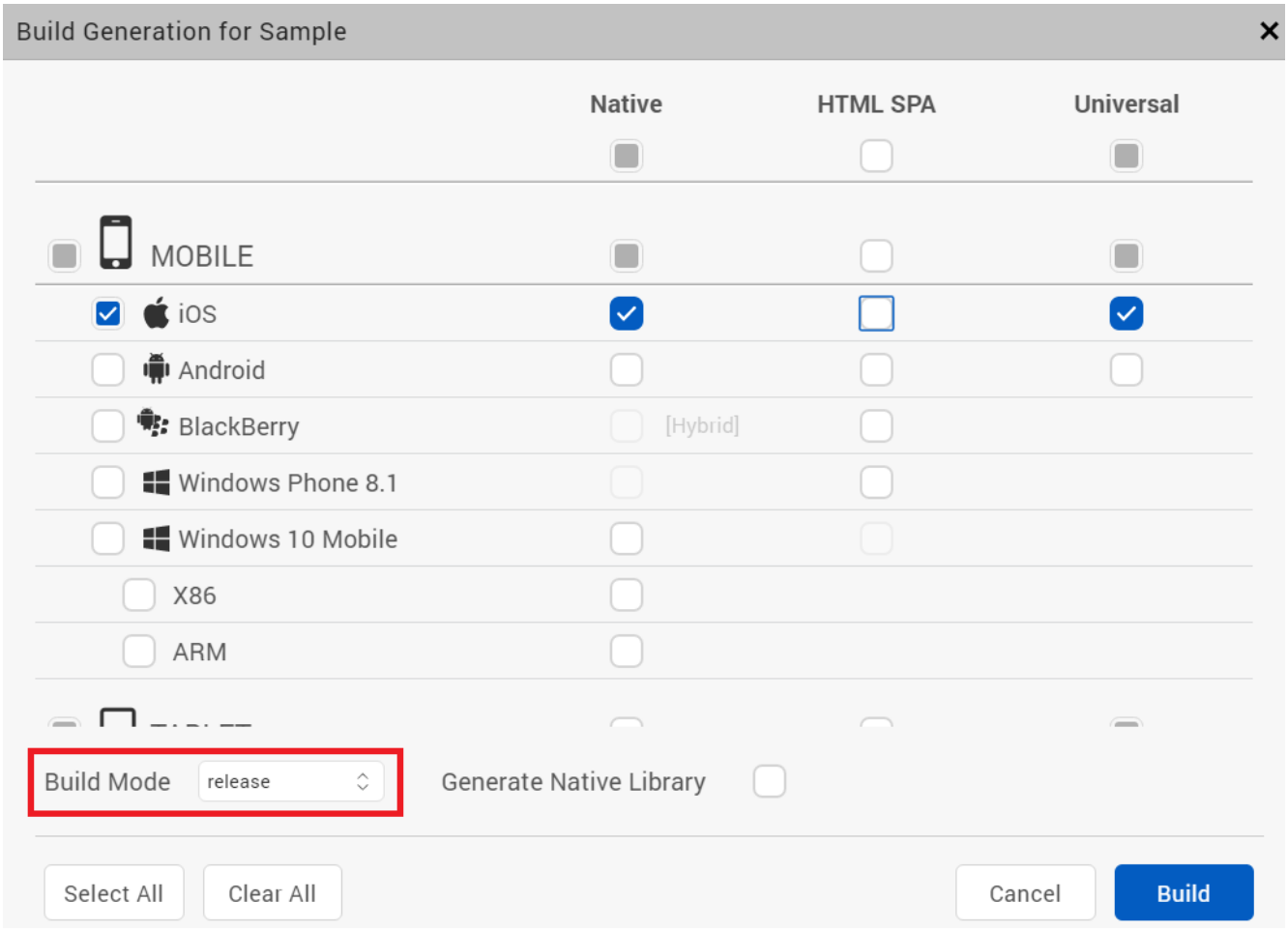
Note: Before you proceed to build your iOS application, you must first go to **Window** > **Preferences** > **Kony Visualizer** > **Devices** > **iOS** and enable automatic builds for iOS. This action ensures that the native XCode project is updated for the app.



Note: In XCode, under **Targets**, you must select the **KProtected** option; not the **KRelease** option.

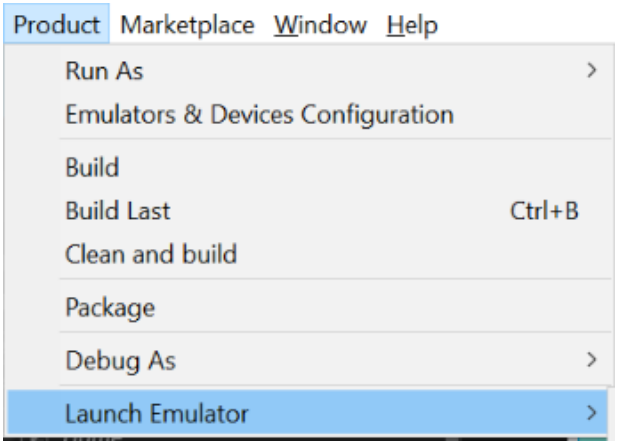
5. Go to **Product** > **Build**. The **Build Generation for Sample** dialog box appears.
6. Select the required channels and platforms.

7. In the **Build Mode** drop-down list, click **release**.



8. Click **Build**.

9. After the application is built, go to **Product** > **Launch Emulator**, and then click the required emulator or device.



Impact on App's Performance

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

Application Bootup Time Comparisions in milli seconds (iOS)			
Device Model Specs & OS	**Release Mode**	**Protected Mode**	**Delta time between Release Mode and Protected Mode**
iPad 3	2026	3794	**1768**
iPhone 5c	1220	2503	**1283**
iPhone 6+	624	1208	**584**

Application Bootup Time Comparisions in milli seconds (Android)			
Device Model Specs & OS	**Release Mode**	**Protected Mode**	**Delta time between Release Mode and Protected Mode**
Micromax MI-3 (Android OS v6.0.1)	1439 ms	4351 ms	**2912 ms**
Samsung S6 (Android OS v7.0)	585 ms	4186 ms	**3601 ms**
Google Pixel (Android OS v8.0)	473 ms	2296 ms	**1823 ms**

Application Security Guidelines

It is recommended that you follow the security guidelines to ensure the application is fully protected.

1. Data at Rest
2. Data Protection
3. Database Security
4. Input Validation
5. Output Encoding
6. Data in Transit
7. Authentication
8. Session Management
9. Access Control
10. Error Handling and Logging
11. File Management

Data at Rest

- Do not store sensitive information on a device (especially on jailbroken and rooted devices).
- Avoid hard coding sensitive data (cryptographic keys, passwords).
- Consider removing any data stored and cached on the device when a user logs out of the application or removes application from the device.

> ***Note:*** If data must be stored, ensure that it is encrypted by leveraging Kony's Cryptographic APIs. In addition, Kony recommends building the application in *Protected Mode* for additional protection of the binary. Although Kony provides many security features and layers of defense, we cannot guarantee that your application will remain secure if best practices and secure design principles are not followed.

Data Protection

- Protect server-side source code from being downloaded by a user.
- Do not store passwords, connection strings, or other sensitive information in clear text or in any non-cryptographically secure manner on the client side. This includes embedding in insecure formats like Microsoft Viewstate, Adobe Flash, or compiled code.
- Remove comments in user accessible production code that may reveal back-end system or other sensitive information.
- Remove debug code and functionality from production code.
- Do not include sensitive information in HTTP GET request parameters.
- Disable autocomplete features on forms containing sensitive information including authentication.
- Disable client side caching on pages containing sensitive information. Cache-Control - no-store, may be used in conjunction with the HTTP header control *Pragma: no-cache*, which is less effective, but is HTTP/1.0 backward compatible.
- The application should support the removal of sensitive data when the data is no longer required. (personal information or certain financial data), such as upon a logout.

Database Security

- Use strongly typed parameterized queries.
- Use input validation and output encoding and ensure you address meta characters. If these fail, do not run the database command.

- Ensure that variables are strongly typed.
- The application should use the lowest possible level of privilege when accessing the database.
- Ensure that only cryptographically strong one-way salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. (Do not use the MD5 algorithm if it can be avoided)

Input Validation

- Identify all data sources and classify them into trusted and untrusted. Validate all data from untrusted sources (for example, databases, file streams, and so on.)
- There should be a centralized input validation routine for an application.
- Specify proper character sets, such as UTF-8, for all sources of input.
- Encode data to a common character set before validating (canonicalize).
- All validation failures should result in input rejection.
- Determine if the system supports UTF-8 extended character sets. If so, validate after UTF-8 decoding is completed.
- Validate all client provided data before processing, including all parameters, URLs and HTTP header content (for example, cookie names and values). Include automated postbacks from JavaScript, Flash, or other embedded code.
- Verify that header values in both requests and responses contain only ASCII characters.
- Validate data from redirects (an attacker may submit malicious content directly to the target of the redirect, thus circumventing application logic and any validation performed before the redirect).
- Validate for expected data types.
- Validate data range.
- Validate data length.

- Validate all input against a white list of allowed characters, when possible.
- If any potentially hazardous characters must be allowed as input, implement additional controls like output encoding and secure task-specific APIs. You should also account for the use of data throughout the application . Examples of common hazardous characters include:

```
< > " ' % ( ) & + \ \' \"
```

- If your standard validation routine cannot address the following inputs, then they should be checked discretely
    - Check for null bytes (%00).
    - Check for new line characters (%0d, %0a, \r, \n).
    - Check for *dot-dot-slash" (../ or ..\)* path alterations characters. In cases where UTF-8 extended character set encoding is supported, address alternate representation like `%c0%ae%c0%ae/`. Utilize canonicalization to address double encoding or other forms of obfuscation attacks.

Output Encoding

- Conduct all encoding on a trusted system (for example, the server)
- Use a standard, tested routine for each type of outbound encoding.
- Contextually output encode all data returned to the client that originated outside the application's trust boundary. HTML entity encoding is one example, but does not work in all cases.
- Encode all characters unless they are known to be safe for the intended interpreter.
- Contextually sanitize all output of untrusted data to queries for SQL, XML, and LDAP.
- Sanitize all output of untrusted data to operating system commands.

Data in Transit

- Implement encryption for the transmission of all sensitive information. This should include TLS for protecting the connection and may be supplemented by discrete encryption of sensitive files

or non-HTTP connections.

- TLS certificates should be valid and have the correct domain name. The certificates also should be current and be installed with intermediate certificates when required.
- Failed TLS connections should not fall back to an insecure connection.
- Use TLS for connections to external systems that involve sensitive information or functions.
- Use a single standard TLS implementation that is configured appropriately.
- Use TLS connections for all content requiring authenticated access and for all other sensitive information.
    - If SSL/TLS is used, do not mix use of encrypted and unencrypted communication (i.e. do not use HTTP and HTTPS in the application) because you may inadvertently disclose information over an unencrypted channel.
- Use mutual authentication (Two-Way SSL) to mitigate man-in-the-middle attacks.
- Use certificate and public key pinning (https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning) to mitigate man-in-the-middle attacks.
- Specify character encodings for all connections.
- Filter parameters containing sensitive information from the HTTP referer, when linking to external sites.

> ***Note:*** If certificate pinning is used, be aware that when the SSL certificate expires or is revoked, a timely application update is required to ensure that users can communicate with the server and avoid interrupted service. Plan an application update for updating certificates.

Authentication

- Authenticate each API call and resource.
- Authentication failure responses should not indicate which part of the authentication data is incorrect. For example, instead of Invalid username or Invalid password, just use Invalid username and/or password for both. Error responses must be identical in both display and source code.
- Use only HTTP POST requests to transmit authentication credentials.
- Enforce password complexity requirements established by policy or regulation. Authentication credentials should be sufficient to withstand attacks that are typical of the threats in the deployed environment. (requiring alphanumeric and/or special characters).
- Enforce password length requirements established by policy or regulation. Eight characters is commonly used, but 16 is better. Also, consider the use of multi-word pass phrases.
- Password entry should be obscured on the user's screen. (for example, on web forms use the input type password).
- Enforce account disabling after an established number of invalid login attempts (for example, a limit of five attempts is common). The account must be disabled for a period of time sufficient to discourage brute-force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed.
- Password reset and changing operations require the same level of controls as account creation and authentication.
- If using email-based resets, only send email to a pre-registered address with a temporary link/password.
- Temporary passwords and links should have a short expiration time.
- Enforce the changing of temporary passwords on the next use.
- Notify users when a password reset occurs.
- Prevent re-use of passwords.

- Disable *Remember Me* functionality for password fields.
- The last use (successful or unsuccessful) of a user account should be reported to the user upon the next successful log-in.
- Re-authenticate users before performing critical operations.
- Use multi-factor authentication for highly sensitive or high value transactional accounts.
- If using third-party code for authentication, inspect the code carefully for malicious code.

Session Management

- Use the server or framework's session management controls. The application should only recognize the session identifiers as valid.
- Log-out functionality should be available from all pages protected by authorization.
- Set the domain and path for cookies containing authenticated session identifiers to an appropriately restricted value for the site.
- Session management controls should use well-vetted algorithms that ensure sufficiently random session identifiers.
- Establish a session-inactivity timeout that is as short as possible, based on balancing risk and business functional requirements. In most cases, the timeout should be no more than several hours.
- Disallow persistent log-ins and enforce periodic session terminations, even when the session is active. Especially for applications supporting rich network connections or connecting to critical systems. Termination times should support business requirements, and a user should receive sufficient notification to lessen negative impacts.
- If a session was established before log-in, close that session and establish a new session after a successful log-in.
- Generate a new session identifier on any re-authentication.
- Do not allow concurrent log-ins with the same user ID.

- Do not expose session identifiers in URLs, error messages, or logs. Session identifiers should only be located in the HTTP cookie header. For example, do not pass session identifiers as GET parameters.
- Periodically generate a new session identifier if the connection security changes from HTTP to HTTPS, as can occur during authentication. Within an application, it is recommended to consistently use HTTPS rather than switching between HTTP to HTTPS.
- Generate a new session identifier and deactivate the old one periodically. (This action can mitigate certain session hijacking scenarios where the original identifier was compromised).
- Set HttpOnly (https://www.owasp.org/index.php/HttpOnly) attributes on all cookies.
- Set the `secure` attribute for cookies transmitted over a TLS connection. (https://www.owasp.org/index.php/SecureFlag)

Access Control

- Access controls should fail securely.
- Deny all access if the application cannot access its security configuration information.
- Enforce authorization controls on every request, including those made by server-side scripts, includes, and requests from rich client-side technologies like AJAX and Flash.
- Segregate privileged logic from other application code.
- Restrict access to files or other resources, including those outside the application's direct control, to only authorized users.
- Restrict access to protected URLs to only authorized users.
- Restrict access to protected functions to only authorized users.
- Restrict direct object references to only authorized users.
- Restrict access to services to only authorized users.
- Restrict access to application data to only authorized users.

- Restrict access to user and data attributes, and policy information used by access controls.
- Restrict access to security-relevant configuration information to only authorized users.
- Server-side implementation and presentation layer representations of access control rules must match.
- If state data must be stored on the client, use encryption and integrity checking on the server side to catch state tampering.
- Enforce application logic flows to comply with business rules.
- Limit the number of transactions a single user or device can perform in a given period of time. The transactions/time should be above the actual business requirement, but low enough to deter automated attacks.
- Use the referer header as a supplemental check only. The referer header should never be the sole authorization check, because it can be spoofed.
- If long authenticated sessions are allowed, periodically re-validate a user's authorization to ensure that the user's privileges have not changed. If the privileges have changed, log the user out and force the user to re-authenticate.
- Implement account auditing and enforce the disabling of unused accounts. (for example, after no more than 30 days from the expiration of an account's password).
- An application must support disabling of accounts and terminating sessions when authorization ceases. (for example, changes to role, employment status, business process, and so on).

Error Handling and Logging

- Do not disclose sensitive information , such as system details, session identifiers or account information,
- Use error handlers that do not display debugging or stack trace information.
- Implement generic error messages, and use custom error pages.
- An application should handle application errors and not rely on the server configuration.

- Properly free allocated memory when error conditions occur.
- Error handling logic associated with security controls should deny access by default.
- All logging controls should be implemented on a trusted system (that is the server)
    - Logging controls should support successes and failures of specified security events.
    - Ensure logs contain important log event data.
    - Ensure log entries that include un-trusted data will not execute as code in the intended log viewing interface or software.
    - Restrict access to logs to only authorized individuals.
- Use a master routine for all logging operations.
- Do not store sensitive information in logs, including unnecessary system details, session identifiers, or passwords.
- Ensure that a mechanism exists to conduct log analysis.
- Log all input validation failures.
- Log all authentication attempts, especially failures.
- Log all access control failures.
- Log all apparent tampering events, including unexpected changes to state data.
- Log attempts to connect with invalid or expired session tokens.
- Log all system exceptions.
- Log all administrative functions, including changes to the security configuration settings.
- Log all back-end TLS connection failures.
- Log cryptographic module failures.
- Use a cryptographic hash function to validate log entry integrity.

File Management

- Require authentication before allowing a file to be uploaded.
- Limit the type of files that can be uploaded to only those types that are needed for business purposes.
- Validate uploaded files are the expected type by checking file headers. Checking for file type by extension alone is not sufficient.
- Do not save files in the same web context as the application. Files should either go to the content server or in the database.
- Prevent or restrict the uploading of any file that may be interpreted by the web server.
- Turn off execution privileges on file upload directories.
- When referencing existing files, use a white list of allowed file names and types. Validate the value of the parameter being passed, if the value does not match one of the expected values, either reject it or use a hard-coded default file value for the content instead.
- Do not pass directory or file paths, use index values mapped to pre-defined list of paths.
- Never send the absolute file path to a client.
- Ensure application files and resources are read only.
- Scan user uploaded files for viruses and malware.

> ***Note:*** For a more comprehensive list, Kony recommends the secure coding checklists and best practices maintained by the Open Web Application Security Project (OWASP).

Jailbroken and Rooted Device Detection

Android

On Android devices, root detection dialog displays when the app is launched in the foreground. The detection happens through the Kony auto-generated launcher activity using KonyMain. If a user writes custom activities using FFI and Framework, root detection does not happen.

If an app is triggered by background sources (Push, GeoFence, SMS, etc.,) root detection dialog does not display. However, a process is created in the background for this app with no application data loaded in memory.

Default Error Message: This device does not meet the minimum security requirements for this application. Please contact the app publisher for more details. The application will exit when you press OK.

You can modify the **Default Error Message** using the i18N string defined with key ROOT_DETECTION_MESSAGE. If you do not have an i18n string, default standard error message appears.

Visualizer tries to respect the i18N key defined in a locale which matches current device locale and then the i18N key defined in a locale which matches Visualizer/IDE set default locale and then the Default Error Message in order of preference.

The following APIs are not respected as they are available only when JavaScript is loaded and JavaScript is never loaded when rooting of a device is detected.

- kony.i18n.setCurrentLocaleAsync
- kony.i18n.setDefaultLocaleAsync

> ***Important:*** You must define your locate specific error message with i18n key ROOT_DETECTION_MESSAGE.

To modify the default error message for Android, do the following:

1. Create a **stringconstants.xml** file with the following content.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="default_root_detetection_message">Custom Root
Detection Message</string>
</resources>
```

2. Modify the Custom Root Detection Message as per your requirement
3. Copy the file in the following folders as required

   For mobile - **<WorkSpace>\<Application>\resources\mobile\native\android\values**

   For tablet - **<WorkSpace>\<Application>\resources\tablet\native\androidtab\values**

iOS

Visualizer tries to respect the i18N key defined in a locale which matches current device locale and then the i18N key defined in a locale which matches Visualizer/IDE set default locale and then the Default Error Message in order of preference.

Whenever an app is launched on a jailbroken iOS device, a jailbroken detection dialog will appear.

To customize the jailbroken dialog in i18n keys manually, do the following:

1. In Kony Visualizer, on the **Edit** menu (the **Project** menu for Kony Visualizer), click **Internationalization (i18n)**. The Configure Internationalization dialog box displays.
2. In the Configure Internationalization dialog box, an initial row for an i18n key and the locales you want your app to support is displayed. In the Key field, type the name for your jailbroken key, **ROOT_DETECTION_MESSAGE**.
3. In each locale's field for the ROOT_DETECTION_MESSAGE key, enter the word or phrase. Here, for English locale, define your message.
4. Click **Finish**.

For more information on how to manually add keys in i18N, click here.

Certificate Pinning

To check trust for communication between an app and a server, server certificates are bundled with the application. Pinning is a process of associating a host with their expected certificate or public key. Once a certificate or public key is known or seen for a host, the certificate or public key is associated or pinned to the host.

Pinning makes use of knowledge of the pre-existing relationship between the user and an organization or service to make the security-related decisions better. As the information is already on the server or service, you do not need to rely on generalized mechanisms meant to solve the key distribution problem. You do not need to turn to DNS for name/address mappings or CAs for bindings and status.

In this document, you will learn about the following topics:

1. Get the Certificate that you want to Pin
2. Enable Certificate Pinning in iOS
3. Enable Certificate Pinning in Android
4. Enable Certificate Pinning in Windows

Get the Certificate that you want to Pin

To get the certificate that is to be pinned, follow any of these two procedures which are common for all platforms:

Generate by using the KonySSLPinningTool.jar Tool

Follow these steps to use the KonySSLPinningTool.jar file:

1. Download the zipped KonySSLPinningTool.jar file to your local system, and then unzip it.

> ***Note:*** You must run the KonySSLPinningTool.jar file with Java 8 or later.

2. Run the following command to save the entire certificate chain from leaf to root individual cert files in der format:
   java -jar KonySSLPinningTool.jar --cert_format DER --ssl_host HOST_NAME --ssl_port PORT
   here, HOST_NAME: The host name of your server.
   PORT: HTTPS port on which your server is listening. It defaults to 443, if --ssl_port option is not specified.

All the certificates are saved with their common name (CN) as the file names after replacing the non-alpha numeric characters with underscore (_)

> ***Note:*** The Windows platform respects only .cer extension for certificates. So, you must change the extension of the generated certs from .der to .cer before performing pinning in the application.

You can get additional information on the supported command line arguments by using this command:
java -jar KonySSLPinningTool.jar -help

Retrieve Certificates through openssl Command or Browser

Follow these steps to retrieve the certificate by using either the openssl command or by using any web browser:

- Use the following openssl command to get the leaf certificate for a site.

  openssl s_client -servername <HOST_NAME> -showcerts -connect < HOST_NAME >:<PORT> </dev/null 2>/dev/null|openssl x509 -outform PEM >mycertfile.pem
  where, HOST_NAME: The host name of your server.

  PORT: HTTPS port on which your server is listening. The port number is usually 443, unless configured differently.

> ***Note:*** This command may return a different certificate when the server supports Server Name Indication (SNI).
> So, while using with servers that support SNI, ensure that the openssl version is 1.1.1a or 1.0.2q or later, which has SNI extension enabled by default.

> ***Note:*** You can verify the Open SSL version by using the following command: **openssl version**

- Alternatively, you can retrieve the certificates by using browser by opening any HTTPS URL of the required domain in your web browser and export the certificate (leaf or any intermediate) by inspecting the certificates fetched from the HTTPS connection. Note that the certificate-exporting procedure may vary from browser to browser and from version to version of the same browser. You can consult the respective usage guides for the relevant procedure.
- Use the following command to convert the certificate to .der format:
  - For iOS and Android: **openssl x509 -outform der -in mycertfile.pem -out certificate.der**
  - For Windows: **openssl x509 -outform der -in mycertfile.pem -out certificate.cer**

    > ***Note:*** Windows platform respects only .cer extension for certificates.

> ***Note:*** The certificate that you choose for pinning impacts the level of security you achieve. The security level decreases as you navigate up the certificate chain from leaf to root certificate. You can pin either the leaf certificate in chain, or the intermediate CA certificate, or pin both the leaf and intermediate CA certificates simultaneously. Typically, the intermediate CA certificate is your organizational CA certificate.

Enable Certificate Pinning in iOS

Follow these steps to enable Certificate Pinning in iOS:

1. Bundle the certificate in the application.
    a. Navigate to the application resources folder and create a **certs** folder in it.
    b. **certs** folder need to be created in the following path:
        i. <workspace>/<app>/resources/mobile/native/iphone/
        ii. <workspace>/<app>/resources/tablet/native/ipad/
    c. Place server certificates inside the **certs** folder.
2. Configure SSL Pinning.
    a. In **infoplist_configuration.json** file, add the entry **{ "allowbundledonly" = true }**. For more information on how to configure custom key value pairs in iOS platform, click here.

Enable Certificate Pinning in Android

Follow these steps to enable Certificate Pinning in Android:

1. Navigate to the application resources folder.
2. Copy the server certificate to the certs folder as shown. Create the folder hierarchy, if required.
    - For mobile - <workspace>/<app>/resources/mobile/native/android/assets/certs/
    - For tablet - <workspace>/<app>/resources/tablet/native/androidtab/assets/certs/
3. In Kony Studio, right-click your application and go to **Properties > Native > Android**.
4. From the **Network Trust Config** (or) **Allow Self Signed/Untrusted Certificates** drop-down list, select **Allow Pinned**.
5. Build the application for Android platform.

Enable Certificate Pinning in Windows

Follow these steps to enable Certificate Pinning in Windows:

1. Navigate to the application resources folder.
2. Copy the server certificate to the certs folder as shown. Create the folder hierarchy, if required.
    - For mobile - <workspace>/<app>/resources/mobile/native/winphone8/assets/certs/
    - For tablet - <workspace>/<app>/resources/tablet/native/windows8/assets/certs/
3. For the mobile channel in Kony Studio, right-click your application and go to **Properties > Native > Windows Phone > Common**.
4. For tablet channel in Kony Studio, right-click your application and go to **Properties > Native > Windows Tablet > Application UI**.
5. From the **Network Trust Config** (or) **Allow Self Signed/Untrusted Certificates** drop-down list, select **Allow Pinned**.
6. Build the application from Windows platform.

Public Key Pinning

SSL Pinning

SSL Pinning is the process of associating a host with their expected X509 certificate or a public key. Once a host's certificate or public key is known or identified, the certificate or public key is associated or 'pinned' to the host. This offers protection against certificate forgery.

You have to take the following decisions in the pinning process:

- **What must be pinned**: Either pin the certificate or pin the public key. Kony supports both Certificate pinning and Public Key pinning.
- **Which certificate/public key to pin against in the chain**: The certificate/public key that you choose for pinning impacts the level of security that can be achieved, and this security level decreases as you navigate up the certificate chain from leaf to root certificate. You must pin either the leaf in chain or the intermediate CA. Typically, you should choose the organizational CA as the intermediate CA for pinning.

Public Key Pinning

Kony has previously rolled out the Certificate Pinning feature. The drawback of Certificate Pinning is that when the server rotates its certificate on a regular basis, you would need to update the application regularly as well.

The Public Key Pinning feature addresses the downside of certificate pinning. By using key pinning, you can avoid frequent application updates as the pubic key can remain same for longer periods.

So if you want to minimize your maintenance efforts and still want a secure communication through your application, then leveraging the HTTP Public Key Pinning feature is the best solution.

> ***Note:*** For Android, you can alternatively achieve Public Key Pinning by using Android Network Security Configuration for API level 24 and later. To learn more about the differences between the Kony Public Key Pinning and Android Network Security Configuration features, click here.

Enable Public Key Pinning

To enable the Public Key Pinning feature for a Kony application, follow these steps:

1. In your Kony Visualizer project, from the Project Explorer, click **Project Setings**. The Project Settings window appears.
2. Click the **Native** tab. A horizontal list of sub-tabs appears under Native.
3. Follow these steps for the required platform:
    - For Android: Go to **Android** > **Mobile/Tablet**. From the **Network Trust Config** drop-down list, select **Allow Pinned**.
      If the Allow Pinned option is not available in Visualizer, you can manually specify the networktrustconfig property in the androidbuild.properties file.
    - For Windows Phone: Go to **Windows Phone** > **Common**. From the **Network Trust Config** drop-down list, select **Allow Pinned**.

- For Windows Tablet: Go to **Windows Tablet** > **Application UI**. From the **Network Trust Config** drop-down list, select **Allow Pinned**.
- For iOS: In **infoplist_configuration.json** file, add the entry **{ "KonyHTTPPublicKeyPinning" = true }**. For more information on how to configure custom key value pairs in iOS platform, click here.

In addition, you must provide the **<workspace>/<platform-specific-path-to-certs>/public_keys.json** file. This file contains all the information on the domain versus the pins configuration (the JSON format is explained in the next section).

Path of public_keys.json file in different Platforms and Channels

In iOS

- For iPhone: <workspace>/<app>/resources/mobile/native/iphone/certs/public_keys.json
- For iPad: <workspace>/<app>/resources/tablet/native/ipad/certs/public_keys.json

In Android

- For mobile: <workspace>/<app>/resources/mobile/native/android/assets/certs/
- For tablet: <workspace>/<app>/resources/tablet/native/androidtab/assets/certs/

In Windows

- For mobile: <workspace>/<app>/resources/mobile/native/winphone8/assets/certs/
- For tablet: <workspace>/<app>/resources/tablet/native/windows8/assets/certs/

> ***Note:*** Make sure the file name containing SSL Pinning Public Keys is in the exact public_keys.json case.

Format of the JSON File for the Public Keys (certs/public_keys.json)

```
{
    "domain-expression1": [
        "sha256-pin1",
        "backup-sha256-pin2",
        // ...
    ],

    "domain-expression2": [
        "sha256-pin3",
        "backup-sha256-pin4",
        // ...
    ],
    // ...
}
```

Domain Name (Expression) Rules

The rules for the domain names/expressions in the public_keys.json file are as follows:

- Wildcard patterns are permitted in hostnames (domain name expressions). The wildcard pattern rules are as follows:
    - * is only permitted in the left-most domain name label and must be the only character in the hostname label.
      For instance, ***.kony.com** is permitted. But, ***a.kony.com**, **a*.kony.com**, **a*b.kony.com**, and **a.*.kony.com** are not permitted.
    - * cannot match across domain name labels.
      For example, ***.kony.com** matches **manage.kony.com**, but it does not match **sub.manage.kony.com**.
- If the hostname is pinned directly as well as via wildcard pattern, both direct and wildcard pins are used for pin validation.

For example, if ***.kony.com** is pinned with pin1 and **manage.kony.com** is pinned with pin2 to check ***.manage.kony.com**: both pin1 and pin2 are used.

Example

```
{
    "*.kony.com": [
        "rSV28bZT885DlLB9/wTzyMuYG+VdA0OlRjjzC72rxno=",
        "JSMzqOOrtyOT1kmau6zKhgT676hGgczD5VMdRMyJZFA="
    ],

    "manage.kony.com": [
        "HA8d0iApa5nQhToDQIcwYQmDYi1rd07MLck8Px4+31B=",
        "JSMzqOOrtyOT1kmau6zKhgT676hGgczD5VMdRMyJZFA="
    ],

    "*.amazon.in": [
        "VkvE/TdvozXh8Frp01wrxI0nh63JIE7FKRt2EQ+Phew="
    ]
}
```

Remarks

- The Public Key Pinning feature is supported on iOS, Android, and Windows 10 tablet and Windows 10 mobile.
- The Public Key Pinning feature is applicable for kony.net APIs, and is not applicable for the kony.ui.Browser widget.
- It is recommended to specify at least one backup pin along with the valid leaf certificate pin. You must not choose the backup pin from the current certificate chain of the host, that is being pinned. You should always choose a pin that is not yet deployed on the server as a backup pin.
- Public Key Pinning does not work in case of self-signed certificates.

- For Android, while using Public Key Pinning, you must set the minimum SDK version as 17 or later.
- From V8 SP4 onwards, the **Allow Self Signed/ Untrusted Certs** option has been renamed as **Network Trust Config**.
- The public_keys.json file specifies a white list of domains and their pins. Consequently, the kony.net network calls made to any other domain that is not listed in the public_keys.json file will fail.

Generate SPKI Pin Hash

To generate the SPKI Pin Hash from the certificate, follow one of these procedures:

Generate by using the KonySSLPinningTool.jar Tool

Follow these steps to use the KonySSLPinningTool.jar file and generate SPKI hashes:

1. Download the zipped KonySSLPinningTool Jar file to your local system, and then unzip it.

   > ***Note:*** You must run the KonySSLPinningTool.jar file with Java 8 or later.

2. Run the following command to generate SPKI Pins Hashes for the entire certificate chain from leaf to root:
   java -jar KonySSLPinningTool.jar --cert_format DER --ssl_host HOST_NAME --ssl_port PORT
   here, HOST_NAME: The host name of your server.
   PORT: HTTPS port on which your server is listening. It defaults to 443, if --ssl_port option is not specified.
   This command prints the SPKI hashes on the console and saves them to a file.

You can get additional information on the supported command line arguments by using this command:
java -jar KonySSLPinningTool.jar -help

Retrieve Certificate and Generate SPKI Pin Hash from the Certificate

Follow these steps to retrieve the certificate by using either the openssl command or by using any web browser.

- Use the following openssl command to get the leaf certificate for a site.
  openssl s_client -servername <HOST_NAME> -showcerts -connect < HOST_NAME >:<PORT> </dev/null 2>/dev/null|openssl x509 -outform PEM >mycertfile.pem
  where, HOST_NAME: The host name of your server.
  PORT: HTTPS port on which your server is listening. The port number is usually 443, unless configured differently.

> ***Note:*** This command may return a different certificate when the server supports Server Name Indication (SNI).
> So while using with servers that support SNI, you must ensure that the openssl version is 1.1.1a or 1.0.2q or later, which has SNI extension enabled by default, or alternatively, use Browser to fetch certificates.

> ***Note:*** You can verify the Open SSL version by using the following command: **openssl version**

- Alternatively, you can retrieve the certificates by using browser by opening any HTTPS URL of the required domain in your web browser and export the certificate (leaf or any intermediate) by inspecting the certificates fetched from the HTTPS connection. Note that the certificate-exporting procedure may vary from browser to browser and from version to version of the same browser. You can consult the respective browser usage guides for the relevant procedure.

- Use the following command to generate the SPKI Pin hash from the certificate:
  openssl x509 -in mycertfile.pem -pubkey -noout | openssl pkey -pubin -outform der | openssl dgst -sha256 -binary | openssl enc -base64

Android Network Security Configuration

As a security improvement , apps that target Android 7.0 (API level 24 and later) and later will no longer trust user-installed certificates on devices with Android 7.0 and later.

From Android 7.0 devices and later, Android has offered a way to customize app behavior for all secure HTTPS communications that originate out of the application (including the WebView/Browser widget network calls) without touching the application code. This is achieved by using a Network Security Configuration that is defined as an xml resource (network_security_config.xml file) in the application.

This xml file helps you to customize the following features in the application:

- **Certificate Pinning**: Helps you to specify which certificates are to be trusted for all secure connections that originate out of the application. Typical examples of certificates pinned include self-signed certificates or a restricted set of organizational and public CAs.
- **Public Key Pinning**: Lets an application limit which server keys are to be trusted for secure connections.
- **Debug-only Overrides**: Helps you to do a different trust configuration for only debug builds.
- **Clear Text Traffic Opt-out**: Helps the application to restrict any HTTP communication or clear text traffic that originates out of the application.

> ***Note:*** For the apps that target Android 9 PIE or later (API level 28 or later), all HTTP communications are disabled by default.

For more information on the usage of the Android Network Security Configuration feature, refer the following links:

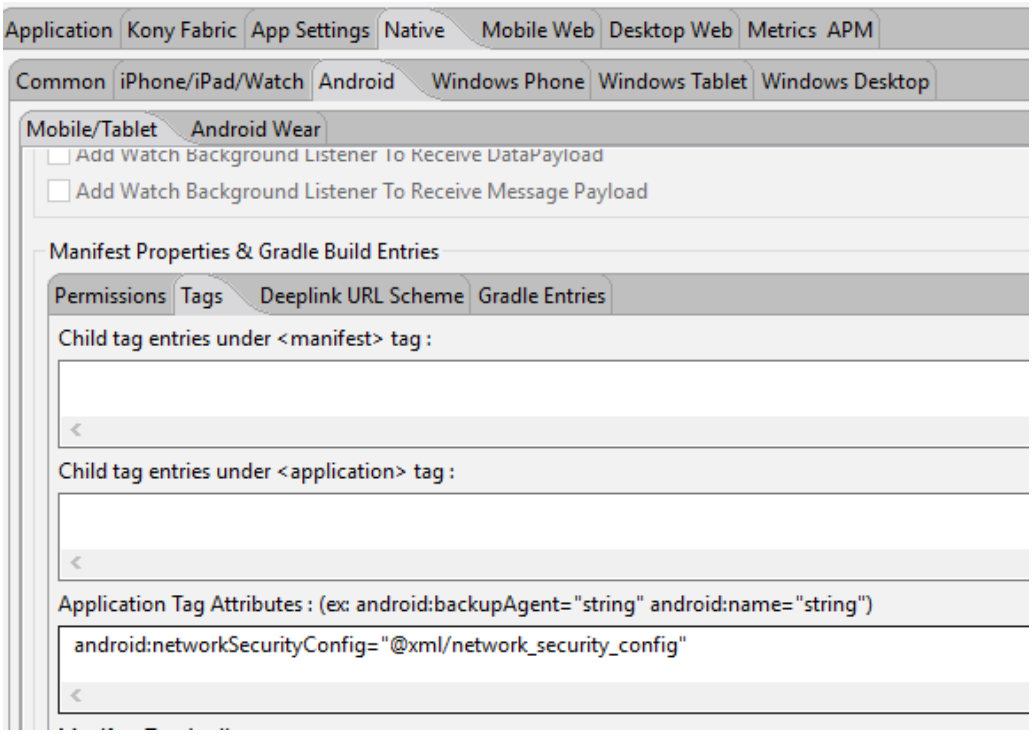
- https://developer.android.com/training/articles/security-config
- https://developer.android.com/about/versions/nougat/android-7.0#network_security_config

Add Android Network Security Configuration

If you want to add support for the Network Security Configuration feature to your existing Kony Andorid project, follow these steps:

> ***Note:*** This feature works only for devices with Android API level 24 and later.

1. In Kony Visualizer, go to **Project Settings > Native >Android >Tags >Application Tag Attributes**.
2. Add the following application tag entry in the app's Manifest file:
   android:networkSecurityConfig="@xml/network_security_config"



3. Add the **network_security_config.xml** file in the following path, as appropriate:
   - For mobile: <vizproject>/<app>/resources/mobile/native/android/xml/network_security_config.xml
   - For tablet: <vizproject>/<app>/resources/tablet/native/android/xml/network_security_config.xml

Kony Public Key Pinning vs. Android Network Security Configuration

The following table illustrates the differences between Kony HTTP Public Key Pinning and Android Network Security Config.

Difference Aspect	Kony Public Key Pinning	Android Network Security Configuration
Support Scope	kony.net.* JS APIs	Entire application including kony.net.* , FFI network calls, and WebViews (including Kony Browser widget)
Hosts Allowed	Network calls to non-pinned hosts fail by default, which is in line with the certificate pinning feature and the pinned hosts are allowed after the pin is successfully validated	Network calls to non-pinned hosts are also allowed
Can Certificate Pinning and Public Key Pinning be applied at the same time?	Mutually Exclusive	Both can be configured simultaneously
Precedence when both features are configured	Respected after Android Network Security Configuration (if already configured) rules are applied	Takes first precedence.
Pin Generation Mechanism	SPKI (Subject Public Key Info)	SPKI (Subject Public Key Info)
Supported Andorid Versions	17 and later (Android 4.2.x)	24 and later (Android 7)
Supported Kony Versions	V8 SP4 onwards	Applicable in any Kony version

Disable Screen Capture and Recording for Android

Applies to *Kony Visualizer Classic*.

With Kony Visualizer, you can provide a deterrent to the capturing and recording of an app's screens for the Android platform.

To disable screen capture and recording for Android, do the following:

1. On the **File** menu, click **Settings**.
2. Click the **Native** tab, and then the **Android** sub-tab.
3. Check the **Disable Application Screenshot** checkbox.
4. Click **Finish**.

Disable Screen Capture and Recording for Windows Phone

Applies to *Kony Visualizer Classic*.

With Kony Visualizer, you can provide a deterrent to the capturing and recording of an app's screens for the Windows Phone platform.

To disable screen capture and recording for Windows Phone, do the following:

1. On the **File** menu, click **Settings**.
2. Click the **Native** tab, and then the **Windows Phone** sub-tab.
3. Check the **Disable Application Screenshot** checkbox.
4. Click **Finish**.

Integrating Third-party Libraries Using FFI

Applies to *Kony Visualizer Classic*.

Foreign Function Interface (FFI) is a mechanism that provides support for the application written in one programming language to use the methods, functions, and services written in another programming language.

FFI leverages this mechanism to invoke native SDK functions from the JavaScript language runtime infrastructure while building the application in the IDE. FFI is also used to integrate third-party tools such as Web Trends, Omniture, and so on.

For a more hands-on approach on how to integrate third-party libraries by using FFI, import and understand the implementation of the Android Calendar Events FFI app by using Kony Visualizer.

DOWNLOAD THE APP

To view a video about FFI implementation, see Root Detector in the Kony Visualizer video series.

> ***Important:*** If you pick up an application that already exists, explicitly generate the code if it is not present.

Advantages/Limitations

FFI feature has the following advantages:

- Methods or functions available in the native SDK can be readily leveraged instead of re-creating the entire functionality in the application.
- Third party Mini application can also be integrated in the Kony application using FFI, For e.g. if a third party library provides a bar code scanning library (with its own UI and screens) such an app can be invoked from a Kony form. In summary once the control is given to the third party library it can show its own screens / forms as well. Using callbacks the user can also return back to the Kony form from a third party screen.

FFI has the following limitations:

- On JDK versions prior to 1.5, you have to explicitly typecast *boolean* to *Boolean*. From JDK 1.5 onwards, the typecasting is done automatically.
- Custom widgets cannot be consumed through FFI.
- Importing custom native types such as structures or classes is not supported currently.
- Importing the native C based functions for platforms like iPhone is not supported currently.

Prerequisites/Assumptions

Make sure the following software is installed:

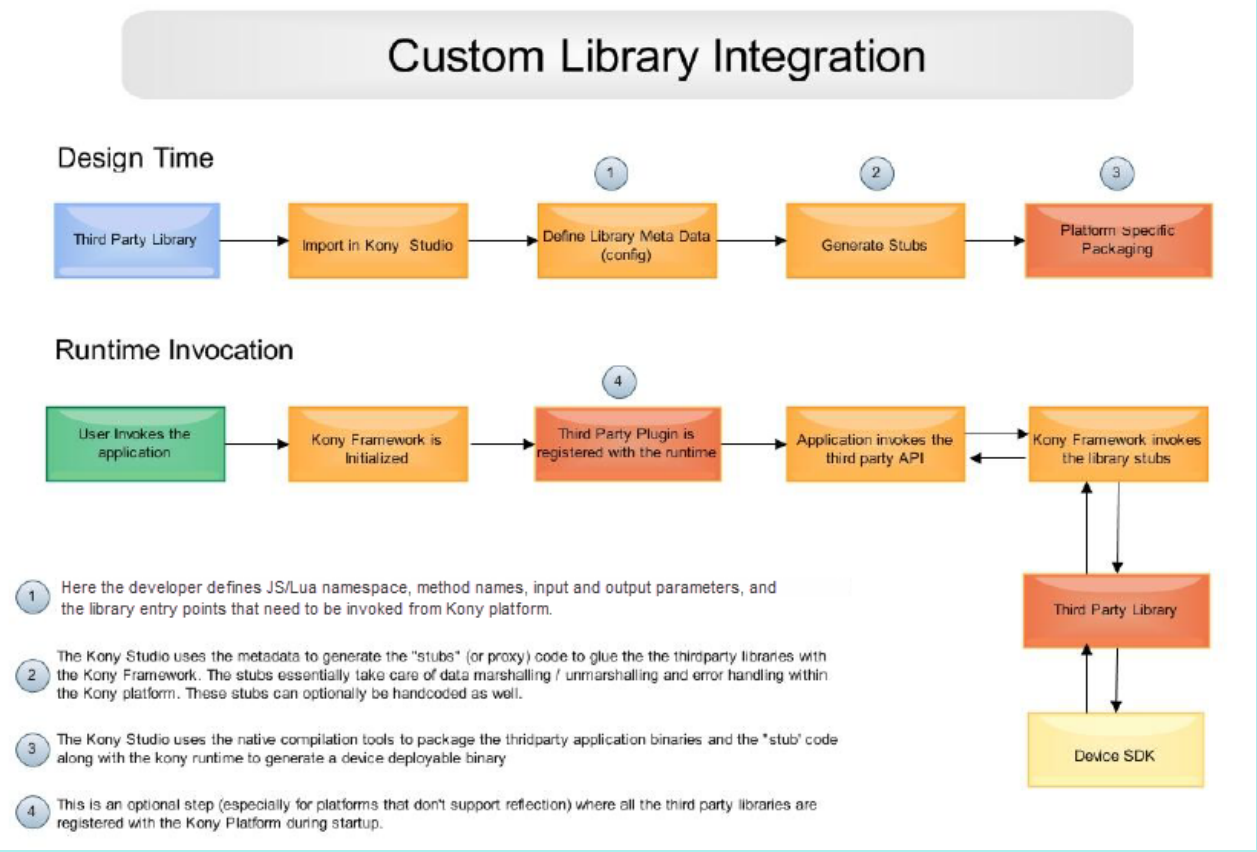
- `Java Development Kit (JDK)` on the local machine
- `.NET Framework 4.0` on the target machine on which you will run the application.

The following are the assumptions for Foreign Function Interface feature in the IDE:

- The metadata is applicable across all platforms. This is because the IDE assumes that for a given functionality, the binaries of different platforms adhere to similar signature.
- The native methods that are accessed are public methods.

FFI Approach

The following diagram illustrates the approach of FFI:



Using FFI for JavaScript Projects

This section lists the JavaScript datatypes mapped across platforms, procedures for integrating a third party library, adding a class, adding a function, and invoking JavaScript methods.

JavaScript Datatypes

This section provides information about the datatypes for the FFI feature:

- JavaScript data types are represented as Object counter parts in the Native SDK as against primitive types. For example, *java.lang.Double* is used instead of *double* and *java.lang.Boolean* instead of *boolean*.

- JavaScript numbers are represented as Objects in native SDK. For example, *java.lang.Double* in Java, *NSNumber* in iPhone and so on.
- The conversion between the Object and primitive types happens automatically on Android platform.

The following table explains the datatype mapping in different languages:

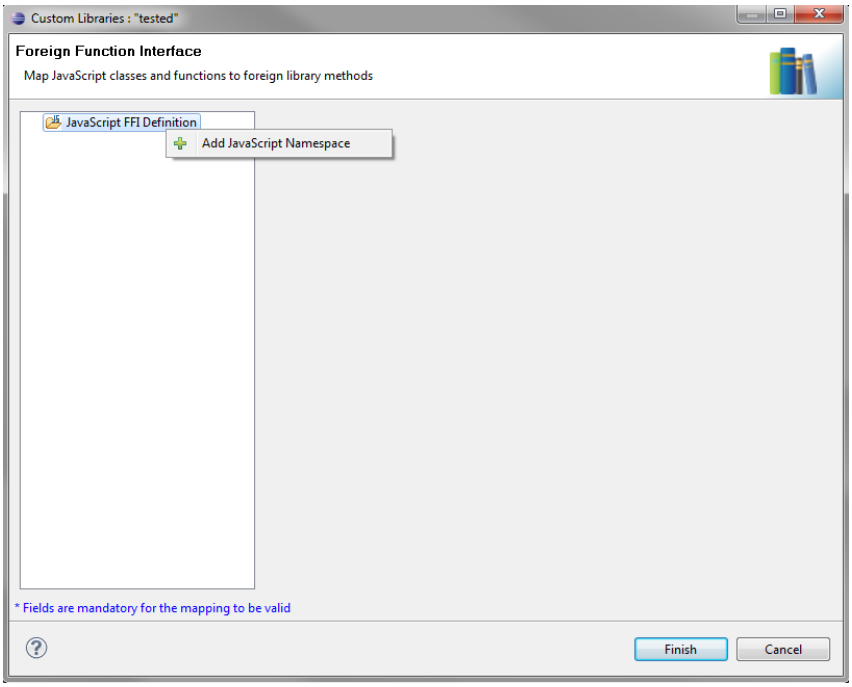
JavaScript Type	Objective -C Type	Java Type	C# Type
Number	int, long,float, double, NSNumber (Depending on the metadata provided)	int, long,float, double , java.lang.Integer, java.lang.Long, java.lang.Double, java.lang.Float (Depending on the metadata provided)	int, long, double (Depending on the metadata provided)
String	NSString	java.lang.String	String
Boolean	BOOL (primitive)	java.lang.Boolean, boolean	bool (primitive)
Array	NSArray	java.util.Vector	System.Collections.GenericList
Object	NSDictionary, id	java.lang.Object, java.lang.Hashtable	object
Function	Callback	com.kony.vm.Function	Closure
void	void	void	void

Adding Third-party Libraries

Integrate the third-party custom libraries with Kony Visualizer to generate XML files. The IDE uses these XML files as input and generates the platform specific Stub templates, which in turn are used by the JavaScript runtime infrastructure to invoke native SDK methods.

To integrate the third-party libraries with Kony Visualizer for a JS project, do the following:

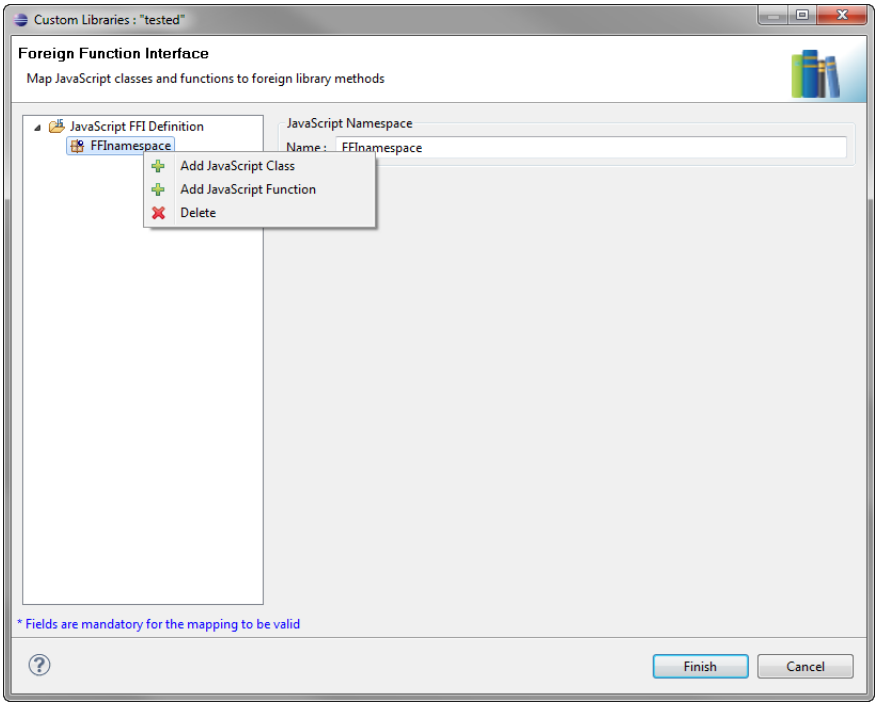
1. On the **Edit** menu, click **Integrate Third Party**, and then click **Manage Custom Libraries**. The Custom Libraries dialog box appears.



2. Right-click **JavaScript FFI Definition**, and then click **Add JavaScript Namespace**.
3. Select  .
4. Enter a name for the namespace.

> ***Note:*** The namespace name must be unique and should not be a part of the Kony system namespace. The namespace cannot start with the word *kony*.

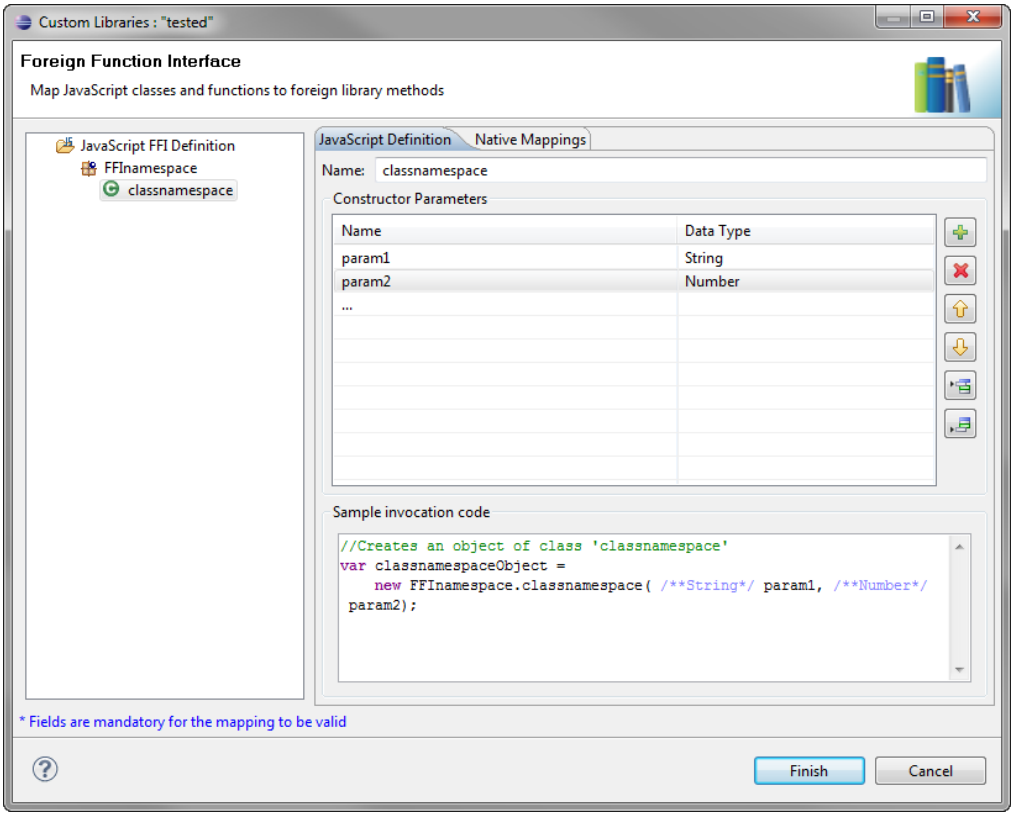
5. Right-click the namespace and do any of the following:
    - Select **Add JavaScript Class** to add a class. A class is the blueprint from which individual objects are created as represented in the external library. In an external library it has only instance methods.
    - Select **Add JavaScript Function** to add a function. A function represents a utility method exposed by the external library and can be accessed without creating individual objects.



After adding a class, follow these steps:

1. Expand the *namespace* and select .
2. In the **JavaScript Definition** tab, enter a name for the created class. The name of a class cannot begin with the name *kony*. When you create a class, you create an object of that class.
3. Add the constructor parameters for the class. The possible data types for the constructor parameters are *String*, *Boolean*, *Number*, *Array*, *Object* or *Function*. The **Sample invocation code** block provides a sample snippet for how the class is invoked.

> ***Note:*** This method does not have any application logic attached to it.



4. You can add parameters rows by clicking . You can remove parameters by selecting a parameter row and clicking . You can change the order in which parameters are passed by selecting the up or down arrows. You can insert a new parameter before another parameter by clicking . You can insert a new parameter after a parameter by clicking .

> ***Note:*** You must select the native datatype that corresponds to each input parameter.

5. In the **Native Mapping** tab, perform mapping of the parameter for every platform. The mapping is an essential step as it associates the JS object with the actual object specified for each individual platform. During runtime, this information is used to invoke the corresponding platform specific native method.

> ***Note:*** If you do not map the Javascript object at least on one platform, an empty `<mappings>` element is created in the corresponding XML file. For more information, see the sample XML file.

i. Click **Manage libraries** icon ( ) to load the third-party library. The **Import libraries** window appears as shown below:

   Click **Import** and browse the required jar file (s).

ii. Or select a **Library** from the drop-down list.

> ***Note:*** Only the libraries that are imported appear under the drop-down list.

iii. The following table shows the acceptable custom library formats that you can load for individual platforms:

Platform	Library File Format
iPhone	`.zip` file (the zip file should have at least one `.a` file and one `.h` file; else the IDE does not accept the zip file and displays an error)
Android	`.jar` file
Windows	`.dll` file
iPad	`.zip` file (the zip file should have at least one `.a` file and one `.h` file; else the IDE does not accept the zip file and displays an error)

iv. For Android and Windows, specify the required **Package** name that is part of the loaded library.

v. For iPhone, specify the required **Framework** name that comprises the libraries. You can specify multiple frameworks using a comma separator.

vi. Specify the **Class** name. This class should exist in the package / framework specified in the previous step. The class name corresponds to the native class on the platform.

vii. Select **Static** if you want to define the method as a static method exclusively.

> ***Note:*** By default, any method is a non-static method, i.e., an *instance method*. An *instance method* is associated with an *object*. You should use `<objectname>.<methodname>` to invoke an instance method.

> ***Note:*** A *static* method is associated with a class. You should use `<classname>.<methodname>` to invoke a static method.

viii. Specify the **Method** name. This method should be part of the class specified in step iii. The method should have an application logic defined.

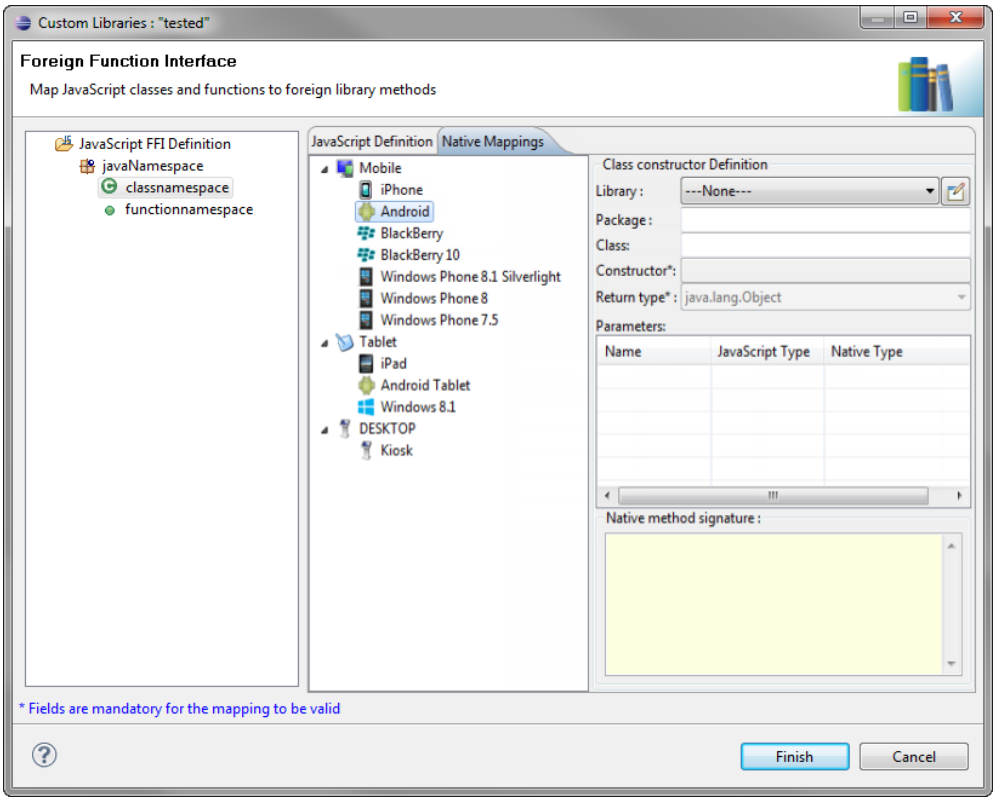
> ***Note:*** The method you specify here should have a one-one correspondence with the JaveScript method signature, i.e., the number of input parameters, the sequence in which they are passed, and the return value should exactly be the same for both the methods without any deviation.

> ***Important:*** If there is no one-one correspondence between the methods, Kony Visualizer throws an error at build time.

For example, the `add` JS method is mapped to the `testMethod` on Android platform. When you invoke the `add` method in any JS code module, the `testMethod` of native language is invoked internally on Android platform. i.e., the application logic associated with `testMethod` is executed.

ix. To copy the mapping created for the mobile platform to the tablet channel or vice versa, right-click the platform name and select ***Copy mapping to xxx***.

x. The method parameters defined in step 3 are automatically populated. Select the **Native type** for each of the input parameters. The **Native method signature** block, displays the native signature of the object on the native platform.



6. Define **Method Exceptions**.
    i. Add an **Error Code** and define specify the corresponding **Error Type**.
    ii. Repeat the above step for all the possible error code for this method.

7. Additionally you can add methods to a class. The methods within the class are non-static. The created method should belong in the same parent class.

> ***Note:*** By default, any method is a non-static method, i.e., an *instance method*. An *instance method* is associated with an *object*. You should use `<objectname>.<methodname>` to invoke an instance method.

i. Right-click the created class and select **Add JavaScript Method**.

ii. In the **JavaScript Definition** tab, select the **Return type** of the method.

iii. Specify the name of the created method and specify parameters within the Parameters area as mentioned in step 3.

iv. In the **Native Mappings** tab, specify the method name of the method on the native platforms.

v. Select the **Return type** per platform. The **Return type** list is populated based on the **Return Type** selection made in the **JavaScript Definition** tab.

vi. Select the **Native types** for each of the input parameters.

8. Repeat steps 2 to 4 and add other methods (if any).
9. Click **Finish**. A confirmation window appears.

   Select the appropriate option based on your requirement.

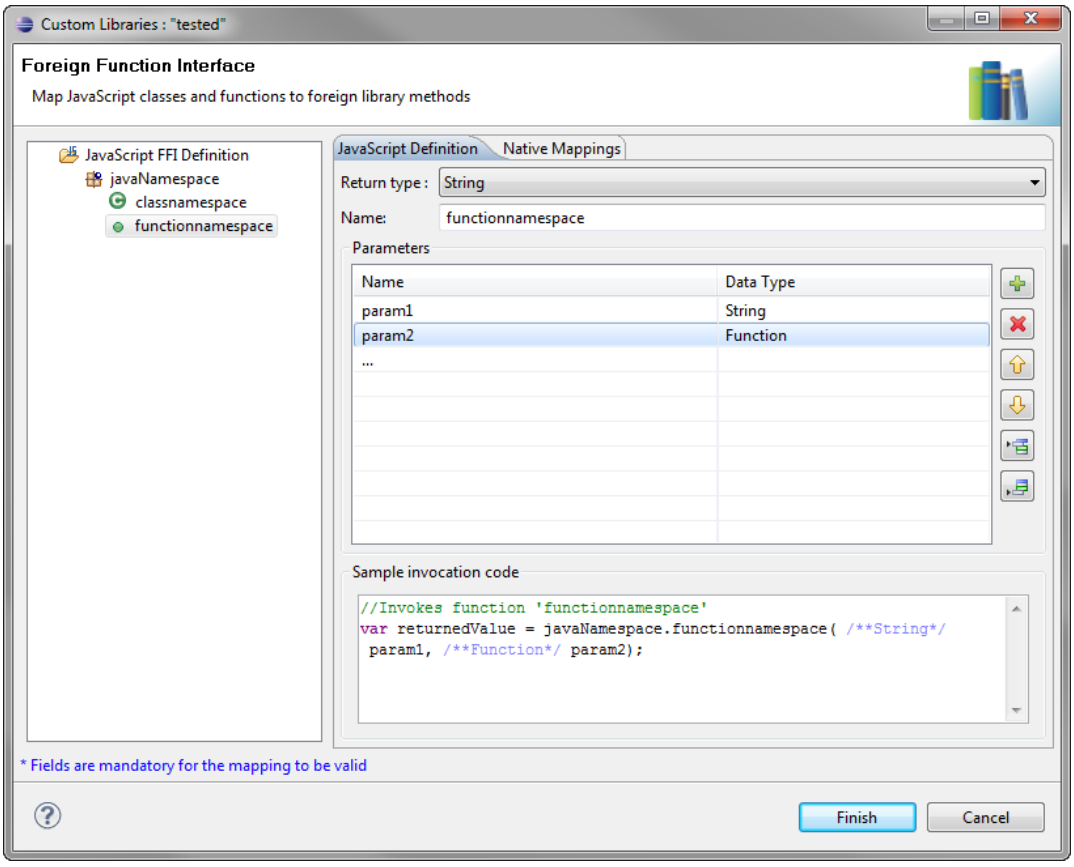
   When you click **Finish**, the FFI configuration is done in the background. As a part of the configuration, an XML file is generated for every namespace. For information about the location of the XML files and generated Stub templates, see FFI Directory Structure.

> ***Note:*** XML files are independent of platforms, whereas the generated Stub templates are platform dependent, i.e., there is one Stub template per platform. You can edit the generated Stub templates manually, if required.

Adding a JavaScript function, execute the following:

1. Expand the *namespace* and select .
2. In the **JavaScript Definition** tab, select the **Return type** of the JavaScript function.
3. In the **JavaScript Definition** tab, enter a name for the created function.
4. Add the input parameters for the method. The possible data types for the input parameters are *String*, *Boolean*, *Number*, *Array*, *Object* or *Function*. The **Sample invocation code** block provides a sample snippet for how the class is invoked.
5. You can add parameters rows by clicking . You can remove parameters by selecting a parameter row and clicking . You can change the order in which parameters are passed by selecting the up or down arrows. You can insert a new parameter before another parameter by clicking . You can insert a new parameter after a parameter by clicking .

> ***Note:*** You must select the native datatype that corresponds to each input parameter.



6. In the **Native Mapping** tab, perform mapping of the functions for every platform. The mapping is an essential step as it associates the JS method with the actual function specified for each individual platform. During runtime, this information is used to invoke the corresponding platform specific native method.

> ***Note:*** If you do not map the JavaScript method at least on one platform, an empty `<mappings>` element is created in the corresponding XML file. For more information, see the sample XML file.

   i. Click **Manage libraries** icon ( ) to load the third-party library. The **Import libraries** window appears as shown below:

   Click **Import** and browse the required jar file (s).

ii. Or select a **Library** from the drop-down list.

> ***Note:*** Only the libraries that are imported appear under the drop-down list.

iii. The following table shows the acceptable custom library formats that you can load for individual platforms:

Platform	Library File Format
iPhone	`.zip` file (the zip file should have at least one `.a` file and one `.h` file; else the IDE does not accept the zip file and displays an error)
Android	`.jar` file
Windows	`.dll` file
iPad	`.zip` file (the zip file should have at least one `.a` file and one `.h` file; else the IDE does not accept the zip file and displays an error)

iv. For Android and Windows, specify the required **Package** name that is part of the loaded library.

v. For iPhone, specify the required **Framework** name that comprises the libraries. You can specify multiple frameworks using a comma separator.

vi. Specify the **Class** name. This class should exist in the package / framework specified in the previous step. The class name corresponds to the native class on the platform.

vii. By default, any function is a static method, i.e., an *instance method*. An *instance method* is associated with an *object*. You should use `<objectname>.<methodname>` to invoke an instance method.

> ***Note:*** A *static* method is associated with a class. You should use `<classname>.<methodname>` to invoke a static method.

viii. Specify the **Method** name. This method should be part of the class specified in step iii. The method should have an application logic defined.

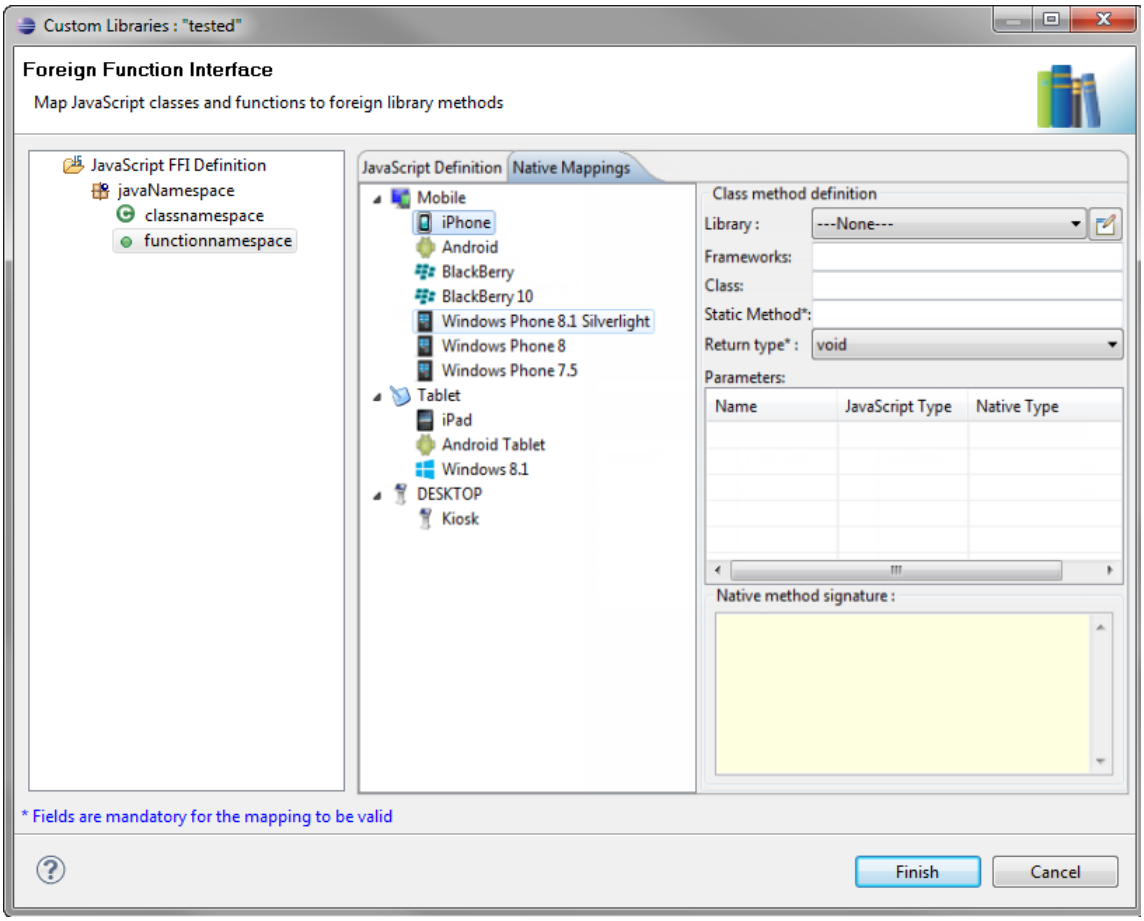
> ***Note:*** The method you specify here should have a one-to-one correspondence with the JaveScript method signature, that is, the number of input parameters, the sequence in which they are passed, and the return value should exactly be the same for both the methods without any deviation.

> ***Important:*** If there is no one-to-one correspondence between the methods, Kony Visualizer throws an error at build time.

For example, the `add` JS method is mapped to the `testMethod` on Android platform. When you invoke the `add` method in any JS code module, the `testMethod` of native language is invoked internally on Android platform. i.e., the application logic associated with `testMethod` is executed.

ix. Select **Pass Activity Context Object to method as first parameter** to pass Activity Context object as the first Parameter to third-party library methods. This step is applicable only for on Android mobile and tablet platforms.

x. The method parameters defined in step 3 are automatically populated. Select the **Native type** for each of the input parameters. The **Native method signature** block, displays the native signature of the function on the native platform.



7. Define **Method Exceptions**.
    i. Add an **Error Code** and define specify the corresponding **Error Type**.
    ii. Repeat the above step for all the possible error code for this method.
8. Repeat steps to add other functions (if any).
9. Click **Finish**. A confirmation window appears.

    Select the appropriate option based on your requirement.

When you click **Finish**, the FFI configuration is done in the background. As a part of the configuration, an XML file is generated for every namespace. For information about the location of the XML files and generated Stub templates, see FFI Directory Structure.

> ***Note:*** XML files are independent of platforms, whereas the generated Stub templates are platform dependent, i.e., there is one Stub template per platform. You can edit the generated Stub templates manually, if required.

Invoking JavaScript Classes

You can invoke the JavaScript Classes defined through FFI in the code modules. Use the following syntax to invoke the JavaScript class(integrated or imported):

```
<variable classobject> = <JavaScriptnamespace>.<class_namespace>
(<input parameters>)
```

The methods mapped through the FFI functionality return the following:

- result

Invoking JavaScript Functions

You can invoke the JavaScript Functions defined through FFI in the code modules. Use the following syntax to invoke the JavaScript function (integrated or imported):

```
<variable returnedvalue> = <JavaScriptnamespace>.<function_namespace>
(<input parameters>)
```

The methods mapped through the FFI functionality return the following:

- result

Building and Packaging the Final Executable

When you build the application, the third-party libraries (custom libraries you have loaded) and the generated Stub templates are packaged along with the application binary. If there are any errors during the Build process, the errors are displayed on the Console.

All the custom libraries are packaged along with the application as they may be inter-dependent on each other.

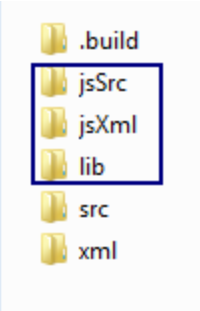
> ***Note:*** If you do not want a specific library to be included in the application binary, delete that library from the `<workspace>/<application name>/customlibs/lib/<platform>` folder.

> ***Important:*** If you delete a library that is in use, the IDE shows an error during Build generation.

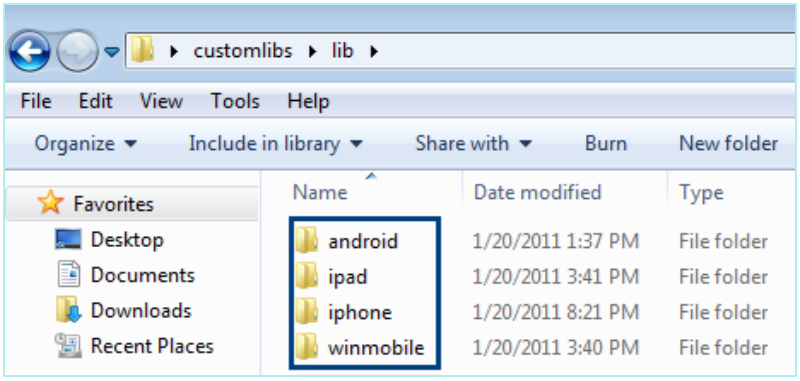
FFI Directory Structure for JavaScript

This section explains the directory structure involved in saving the FFI related files. FFI related files are stored at the following location on your local machine: `<workspace>/<application name>/resources/customlibs`. The `customlibs` directory contains the following resources:

- custom libraries
- generated XML files
- generated Stub templates.



- **/lib** (contains all the loaded custom libraries grouped by platform)
  - /android
  - /ipad
  - /iphone

- /winmobile and so on
- 

- **/jsSrc** (contains the generated Stub templates grouped by platform). For more information, see the sample Stub template.
  - /android
- **/jsXml** (contains the XML files generated by the IDE - one for each JS namespace).

Guidelines for Third-party Libraries

This section explains the guidelines that a third-party library must follow before it is integrated with Kony Visualizer.

iPhone

Prerequisite

- Ensure that the library uses the Object versions of primitive types while writing the Hybrids for FFI compatibility.
  For example, if a number is mentioned in the IDE, the Hybrid should accept NSNumber.
- Ensure that namespace for FFI does not clash with the names of the header files in the imported zip file.
  For example, if the namespace is `zxling` then there should not be any header file with the name `zxling.h`.

- Ensure that any dependent frameworks are added as part of the Xcode project configuration before building the Xcode project.

Invoking callback functions for iPhone

To execute a callback from FFI, follow these steps:

1. Import *Callback.h* file into the FFI class. For example: `#import "CallBack.h"`
2. Following are the two APIs to be called upon Callback object [passed from JavaScript], to execute the JavaScript method, as required.
    a. (NSArray*) executeWithArguments: (NSArray*) arguments;
        - **arguments**: Array containing the parameters to be passed for the JavaScript function.

    b. (NSArray*) executeWithArguments: (NSArray*) arguments spawn:(BOOL) spawn;
        - **arguments**: Array containing the parameters to be passed for the JavaScript function.
        - **spawn**:

          If set to *Yes*, then Callback is executed on the thread other than the main thread (which is responsible for executing all the JavaScript code). To avoid synchronization problems, it is advised to set as *Yes*.

          If set to *No*, then Callback is executed on the thread from which the FFI is called.

3. To execute any functionality asynchronously, FFI can create its own thread using Objective-C, NSThread APIs

Android

To develop a third-party Java library and to integrate it with Kony Application using FFI, you need the following .jar files:

- `android-support-v4.jar` - to access activity context. This file is located at `<workspace>/temp/<app>/build/luaandroid/dist/<app>/libs`.
- `konywidgets.jar` - to access some classes and methods to communicate back-and-forth between third-party libraries and Kony Platform. This file is located at `<workspace>/temp/<app>/build/luaandroid/dist/<app>/libs` when you build the application.
- `android.jar` - to access some Android specific APIs or refer Android classes. This file is located at `<android-sdk>/platforms/android-<api_level_number>`. For more information about API Levels, see http://developer.android.com/guide/appendix/api-levels.html.

You must link these jar files to your third-party Java library project while developing the application. After you link these files, you do not need to import these files while integrating Java library with FFI.

Invoking callback functions for Android

Callback functions that are mapped as parameters to third-party library methods are Function objects. The class Function is present in `konywidgets.jar` file. To invoke a callback function from third-party library, call `callback.execute` (`new Object[]{param1, param2, ….}`) where callback is of type `Function` and `param1, param2...` are arguments to callback function. The type of those arguments should match type of arguments of corresponding callback function. If you do not want to pass arguments, call `callback.execute` with null as the argument.

The `Context` object

From 5.0, IDE does not support passing implicit Android Context object ("Pass Context object as first parameter") to the FFI APIs.

Application developers (FFI Jar dev) would be able to get the Android context object using one of the following API's which are available in the *konywidgets.jar* file.

- KonyMain.getActivityContext();
- KonyMain.getAppContext();

Do not store the context object locally since the *Activity context* becomes invalid(null) based on the application state.

KonyMain.getActivityContext() can possibly return null, Application developers should check for not null before using it.

It is recommended to use "*KonyMain.getAppContext()*" unless Activity context is specifically required.

The `JavaScript` Function

The class `com.konylabs.vm.Function` represents the JavaScript datatype `Function`. If a callback function takes JavaScript Object as its parameter while invoking callback in Java, creates object of either Hashtable or Vector depending on the type of data. FFI Java should pass Hashtable if callback param Object is expected to be of type key-value pair JavaScript Object. FFI Java should pass Vector if callback param Object is expected to be of JavaScript Array object.

Example

```
Hashtable < String,
String > param1 = new Hashtable < String,
String > ();
param1.put("key1", "value1");
param1.put("key2", "value2");
Vector < String > param2 = new Vector < String > ();
param2.add("item1");
param2.add("item2");
callback.executeAsync(new Object[] {
    param1, param2
});
```

Consuming services offered by Android System or third-party or native applications installed on a Device

An application can leverage the services offered by another/same application or Android system itself in an asynchronous manner through Intent mechanism. These services can be in the form of UI or non-UI. In both the cases, some sort of data is returned back to calling application asynchronously as a result.

When you want to use services offered by another/same application, you must send an appropriate Intent to Android System. In case of UI, the system handles launching appropriate Activity of an application which can handle this Intent. The system will show a list of Activities if there are more than one which can handle this Intent. In case of non-UI, a listener is registered along with the Intent to the system so that system will broadcast the result to registered listeners.

The methods `android.content.Context.startActvity()` and `android.app.Activity.startActivityForResult()` can be used to send the intent to system for services involving UI. The former method launches the activity that handles Intent and latter method launches the activity. The launched activity returns the result back to the caller in an asynchronous manner. The method `android.content.Context.registerReceiver()` can be used to register a listener with the system along with intent for services involving non-UI.

For more information about intents, see Intents and Intent Filters on the Android developer site.

For information about Android API documentation, see Package Index on the Android developer site.

Capturing results of a service involving UI

For third-party libraries to capture the results of another/same application's activity, the Kony Android provides an interface named `com.konylabs.ffi.ActivityResultListener`. The third-party library class must implement and register this interface with the platform through `com.konylabs.android.KonyMain.registerActivityResultListener()`.

Interface Description

```
package com.konylabs.ffi;
import android.content.Intent;
```

```
/**
 * Callback interface which the client needs to be implemented and
registered with
 * ActivityResultPublisher (a.k.a com.konylabs.android.KonyMain) along
with
 * unique requestCode.
 */
public interface ActivityResultListener {
    /**
     * Callback that is called when  some other activity, launched via
                * Context.startActivityForResult(), exits.
     * Ensure that requestCode with which callback is being registered
should be        * the same as the one passed to startActivityForResult().
     * Same requestCode will be sent as one of the input parameters.
     *
     * @param requestCode - unique token with which this callback was
registered.      *      This should be the same as the one with which an
Activity was invoked                    via startActivityForResult().
     * @param resultCode - result code indicating status of the
result. Either of               Activity.RESULT_OK or Activity.RESULT_CANCELLED.
     * @param data - intent data that is passed by the called
Activity.
     */
    public void onActivityResult(int requestCode, int resultCode,
Intent data);
}
```

Interface Registration Method and Description

```
/**
 * Registers a callback that will be invoked when some other activity,
                * launched via Context.onActivityForResult() exits.
 * Ensure that requestCode with which callback is being registered
should be        * the same as the one passed to startActivityForResult().
 *
```

```
 * @param requestCode - unique token that is mapped to this callback.
This             *      should be the same as the one passed to
startActivityForResult().
 *
 * @param activityResultListener - the callback listener which the
client           *      needs to be implemented or null to deregister callback.
 */
public void
com.konylabs.android.KonyMain.registerActivityResultListener(
    int requestCode, ActivityResultListener activityResultListener);
```

Typical examples where you can use ActivityResultListener to capture result in third-party libraries are to:

- get the captured image by launching native camera application.
- pick an image or file from native gallery or file explorer application
- pick a contact from native phone book application.

Apart from these, you can use the services offered by other third-party applications installed in a particular device.

Capturing results of a service involving non-UI

For third-party libraries to fetch the result of a service involving non-UI, the client Class must register an instance of `android.content.BroadcastReceiver` with the system along with Intent for which it wants to listen. To register a listener, invoke `Context.registerReceiver()` and then implement its `onReceive()` method. For more information about BrodcastReceiver, see BroadcastReceiver on the Android developer site.

Typical examples where you use BroadcastReceiver to capture result in third-party libraries are:

- Knowing battery status like percentage of charge, battery connected or disconnected.
- Knowing system shutdown or booting finished

- Knowing alarm expiry or timezone changed.

Apart from these system-wide notifications sent by Android system, there can be application specific broadcast notifications. Those applications can be native or third-party applications installed on the device.

Extracting Result Data

The actual result data returned by the invoked Activity (in case of UI) or broadcast message sent (in case of non-UI) to the caller, will be present in Intent object parameter in `onActivityResult()` or `onReceive()` method respectively. The type of data that is returned is defined by the invoked Activity or broadcast message sender of a third-party/native application or Android system. Android defines a protocol on how to extract data. for more information, see Intent on the Android developer site.

Once the actual data is extracted, pass it back to the Kony Application as parameters to registered JavaScript callback function (optionally) by parsing that data into a type that can be interpreted by the JavaScript code.

Intent to select/pick an image or a file

```
//registering current class as an ActivityResultListener for the
specified request code. Current class implements
ActivityResultListener.
((KonyMain) context).registerActivityResultListener(SELECT_FILE_
REQUEST_CODE, this);
Intent intent = new Intent();
intent.setAction(Intent.ACTION_GET_CONTENT);
//set the mime-type of content to be selected. For only images specify
image/*
intent.setType("*/*");
((Activity) context).startActivityForResult(intent, SELECT_FILE_
REQUEST_CODE);
```

where SELECT_FILE_REQUEST_CODE can be any integer. Same integer is returned as a parameter to `onActivityResult()`. In `onActivityResult()`, get the URI of selected image/file from Intent data object as `Uri uri = data.getData()`.

Intent to get the image captured by Camera application

```
//registering current class as an ActivityResultListener for the
specified request code. Current class implements
ActivityResultListener.
((KonyMain) context).registerActivityResultListener(CAMERA_REQUEST_
CODE, this);
//create an empty image file at pre-defined location
File capturedImage = new File(FILE_LOCATION_PATH);
Uri capturedImageUri = Uri.fromFile(capturedImage);
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
//specify the uri at which camera should save the captured image
intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
((Activity) context).startActivityForResult(intent, CAMERA_REQUEST_
CODE);
```

where CAMERA_REQUEST_CODE can be any integer. Same integer is returned as a parameter to `onActivityResult()`. In `onActivityResult()`, get the URI of captured image from Intent data object as

```
Uri uri = data.getData()
```

For more information about the Camera API, including video capture, see Camera on the Android developer site.

Intent to send an Email

```
Intent emailIntent = new Intent();
emailIntent.setAction(Intent.ACTION_SEND_MULTIPLE);
emailIntent.setType("text/plain");
//specify email addresses of primary recipients
```

```
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[] {
    to
});
//specify email addresses of secondary recipients
emailIntent.putExtra(Intent.EXTRA_CC, new String[] {
    cc
});
//specify email addresses of tertiary recipients
emailIntent.putExtra(Intent.EXTRA_BCC, new String[] {
    bcc
});
//specify subject
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
//specify email body
emailIntent.putExtra(Intent.EXTRA_TEXT, message);
//specify attachments if any
ArrayList < Uri > attachmentUris = new ArrayList < Uri > ();
for (String attachment: attachments) {
    Log.d(TAG, "Attachment: " + attachment);
    attachmentUris.add(Uri.parse(attachment));
}
emailIntent.putParcelableArrayListExtra(Intent.EXTRA_STREAM,
attachmentUris);
context.startActivity(emailIntent);
```

This intent opens an email client installed on the device capable of handling this intent. This intent does not result any data back to the caller.

Intent to pick a contact

```
//registering current class as an ActivityResultListener for the
specified request code. Current class implements
ActivityResultListener.
((KonyMain) context).registerActivityResultListener(PICK_CONTACT_
```

```
REQUEST_CODE, this);
Intent intent = new Intent(Intent.ACTION_PICK);
//specify URI of the Contacts Table
intent.setData(ContactsContract.Contacts.CONTENT_URI);
context.startActivityForResult(intent, PICK_CONTACT_REQUEST_CODE);
```

where PICK_CONTACT_REQUEST_CODE can be any integer. Same integer is returned as a parameter to onActivityResult(). In onActivityResult(), get the URI of selected contact from Intent data object as

```
Uri contactUri = data.getData()
```

Using ContentResolver and this contactUri get the Cursor object to selected contact as

```
Cursor cursor = context.getContentResolver().query(contactUri, null,
null, null, null, null)
```

For more information about retrieving information from the cursor and contacts table, see the following on the Android developer site:

Content Providers

Cursor

ContactsContract.Contacts

Intent to get the battery status

```
//register a broadcast receiver to receive updates regarding battery
status
mContext.registerReceiver(mBatInfoReceiver,
    new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
```

where `mBatInfoReceiver` is an instance of class that extends `BroadcastReceiver`. System will broadcast the information about battery such as charging level, status, and other information about battery to the registered broadcast receivers as intent data to its `onReceive()` callback. In `onRecieve()`, retrieve information about battery like charging status, level etc from the intent object passed to it.

```
//get the level of battery charge
int level = intent.getIntExtra(BatteryManger.EXTRA_LEVEL, 0);
//get the status of battery such as charging, discharging, full, not
charging
int status = intent.getIntExtra(BatteryManger.EXTRA_STATUS, 0);
```

For more information, see BatteryManager on the Android developer site.

Guidelines

- The Java Class on which the static method is invoked should have a *public* default constructor:

```
public class Foo
{
......public Foo()
......{
......}
}
```

- A few Android native APIs require access to the *Activity* class present in the `konywidgets.jar` file. Ensure that this JAR file is located at `<workspace>\temp\<appName>\build\luaandroid\dist\<appName>\libs\` folder. Use the static method `com.konylabs.KonyMain.getActContext()` to access the current *Activity* class.

- You can reference any of the APIs that are already present on the device (including `konywidgets.jar`) in the Java projects. But ensure that you do not import these JAR files into Kony Visualizer.

Windows

The following are the operating systems supported for FFI.

> *Important:* To use the information in this section, you must be an expert at Visual Studio and in creating class libraries. If you are not sure how to use the information below, please contact your administrator.

- **Windows 10**: Windows store class libraries are allowed to deploy onto the Windows 10. Class libraries developed with other .Net framework libraries are not allowed to deploy. Download and install Visual Studio from MSDN.

Prerequisites

- Targeted system must have .Net framework 4.5
- Appropriate SDKs have to be installed for native library development. App developers can download SDKs from MSDN locations.
    - Windows10

Exposed Syntax Restrictions

All functions defined in the library contract of Kony studio must be publicly accessible. Parameter passing type can be “ref” (?), “out” (?) or normal “in”.

See the following sample FFI Calculator Class for FFI Windows 10.

```
using Framework;
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
```

```
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
namespace Calc {
    public class Calculator {
        public int Add(int a, int b) {
            Util.Trace("The Addition is" + a + b);
            return a + b;
        }
        public int Subtract(int a, int b) {
            return a - b;
        }
        public int Divide(int a, int b) {
            return a / b;
        }
        public int Multiply(int a, int b) {
            return a * b;
        }
    }
}
```

Implementing FFI in Windows

If the native library has libraries specific to architecture (X86 or ARM) or foundation, dll is referenced to call closure or write debug statements. Library hierarchy should be in a zip file. format. For example, if **CustomLibrary.dll** is used for Windows Phone 8, for each processor, the architecture should include **ARM\CustomLibrary.dll** folder structure and **X86\CustomLibrary.dll** folder structure.

1. If you want to call a closure by referencing the *foundation.dll*,
    i. Create a zip file, change the build configuration to ARM, add reference to foundation.dll for ARM architecture, and build the class library.
    ii. Change the build configuration to X86 and add a reference to *foundation.dll* for X86

architecture, and build the class library.

> ***Note:*** Since foundation.dll is already present in the build folder, you do not need to add it to the zip file.

iii. Take the class library from the build folder of the visual studio for each architecture types and create two folders ARM and X86. Place the respective architecture built class library and dependent component in the folders and zip it. Keep the name of the zip file the same as the name of class library, in current example the zip file name will be Classlibrary1.zip.

iv. Now in kony studio, manage custom libraries , Select the zip file for the Library.

2. FFI is commonly executed on a background thread. However user interface components can be invoked by wrapping the UI component execution within the code specific to windows phone. To call user interface methods in the FFI Library, see the example below.

```
AutoResetEvent ar = new AutoResetEvent(false);
Deployment.Current.Dispatcher.BeginInvoke(() = > { //your code
goes herear.Set();});ar.WaitOne();
```

> ***Note:*** It is necessary to include dependent libraries for the class library used for FFI, else the FFI may not work at all.

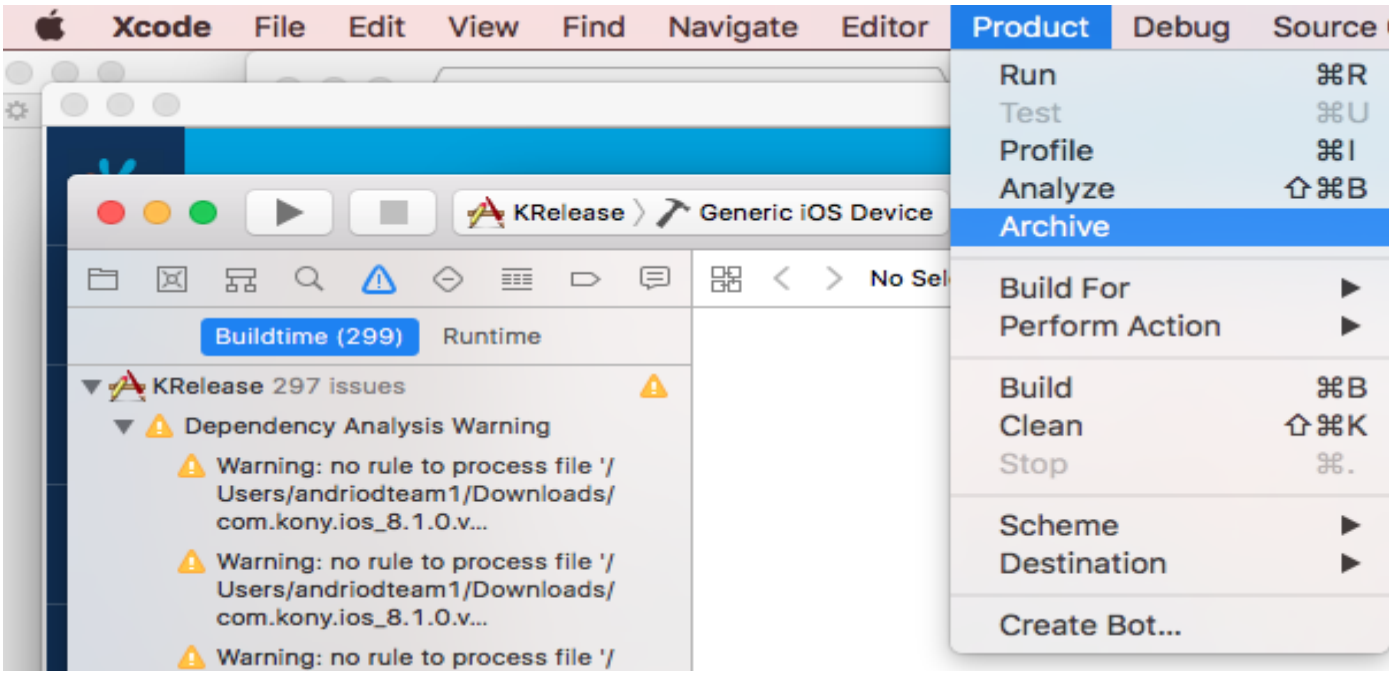
3. The method **Util.Trace** will log your statements to **Logger** output you can see in debug mode. To use **Util.Trace** specified in Add method (in the example), add a reference to **Foundation.dll**.

Generating IPA from KAR files

To generate an .ipa file from the .kar file in Xcode, follow these steps:

1. Once you extract the kar file, open the Xcode project.
2. Select a device under **KRelease**.

3. Select the option **Product** > **Archive**.



4. Click next in all other windows that follow.

5. Click **Finish**.

Developing Offline Applications

Applies to *Kony Visualizer Classic*.

Offline Applications enable you to access services on your device if it is not connected to a network (data). This basically provides you with the flexibility to define a service for your application which can be accessed by the user when the device is not connected to a network. You can access the following Data sources offline by using the Kony Sync tool chain View.

- Sky Data Objects
- Databases
- Web Services
- SAP Services
- Siebel Services

For more information, see Synchronization and Object Services.

Building and Viewing an Application

Kony Visualizer gives you different ways of building and viewing your app, depending on your needs. In Kony Visualizer Classic, you can build your app either from the **Product** menu, or from a command line. And you can view your built app either by previewing it on a mobile device, or by using device emulators and simulators that you configure.

For a smooth build in Kony Visualizer Classic, you'll want to ensure that the following issues have been addressed:

- Ensure that Kony Fabric is set up in Kony Visualizer and that you have selected a default Kony Fabric environment to publish to. For more information, see Connect to the Kony Fabric Console.
- Be prepared to provide your Kony Visualizer licensing information the first time you build an application.
- You can choose to not generate print statements during either or both Debug Mode builds and Release Mode builds. For more information, see Disable Print Statements in Builds.
- For Android, ensure you have set the minimum SDK version to between 4.0 and 4.4. To change this setting, on the **File** menu, click **Settings**. Click the **Native** tab, and then click the **Android** sub-tab. The Minimum setting is located in the SDK Versions section of the dialog box. After changing the setting, click **Finish**.
- For Android, ensure you have set the location of the Android SDK. To do so, on the **Window** menu, click **Preferences**. In the left pane, double-click **Kony Visualizer**, and then click **Build**. If it hasn't done so already, at this point Kony Visualizer auto-detects the Android SDK and asks if you would like to use the path that it has discovered as the Android Home. If you wish to, click **OK**. If Kony Visualizer did not auto-detect the Android SDK, in the **Android Home** text box, enter the path to the Android SDK packages. To ensure you don't introduce errors into the path that you type, you may want to click the accompanying **Browse** button, navigate to the Android SDK's location, and then click **OK**.

- For Android, ensure you have set the SDK home environment variable for your computer. To do so, see Set the Android SDK Home Environment Variable.

> ***Important:*** If your app contains deprecated widgets, it's possible that their skins may make reference to the Helvetica font. If you are not explicitly using Helvetica in your app, you should verify your app's configuration and manually remove references to Helvetica before submitting it to the store. The following topics are covered with regard to building an app using Kony Visualizer.

Build an iOS Application

Build an Android Application

Build a Universal Application

Build a Windows 10 Application

Build an SPA Application

Build a Native App on a Local machine

Disable Print Statements in Builds

Build an App Offline

Perform a Headless Build

Continuous Integration

Publish a Project to KonyFabric

Preview an App on a Device

Monitor an App's Performance

View an Application on an Emulator

iOS Build Automation

Open Webapp and Build Folders

Incremental Build

Build in the Background

When a user builds an application in Kony Visualizer, till Kony Visualizer V8, the user was not allowed to perform any other activities on Visualizer till the build is complete. The user is effectively blocked from performing any design activities when the build is in progress. From Kony Visualizer V 8.1 release onwards, some parts of the build process happens in the background thus enabling a user to continue with their design activities while the build is in progress.

When you click on Build, the Initiating Build message appears. Then, the dialog box minimizes to the console panel, and the build continues in the background. You can see the Build status in the Build tab next to the Console and Search tabs in the console panel.

The tab displays information on different types of builds you chose and their status. A blue progress bar displays the progress of the build for each of the channels you chose. Once the build process is complete, an alert pop up appears on the screen. Once you close the alert popup, the build panel displays the build status for each one of the channels.

Incremental Build

While developing any application on Visualizer, you must undertake certain steps such as testing the application and building the binaries multiple times. For every change that you make in the app, the whole project needs to be built each time; thereby, increasing the build time.

To ensure that there is a faster turnaround, a new feature, Incremental Build is introduced in Kony Visualizer V8 SP4 release.

When an app is built for the first time, a full build is performed. The subsequent builds are incremental. In an incremental build, the previously built state of the project is used and only those resources that are changed since the last build are regenerated.

The following are the scenarios where the build is always a full build.

- If controllers that interact with the NFI are modified.
- If controllers that interact with third-party libraries are modified.
- If any changes are made to the property window. For example, SDK/iOS deployment.
- If a marketplace component is added into the Visualizer project.
- If a new app module(group) is added into the Visualizer project.
- If any changes are made to plugins like Sketch and Adobe Photoshop.
- If the following project settings are modified:
    - Accessibility Config
    - Enable Cordova
    - Enable Action Bar (for Android native platform)
    - Enable Responsive Web (for Desktop Web platform)
    - Enable i18n Layout Config
    - Enable designer/ developer actions

You do not have to perform an incremental build and a full build separately. Visualizer will automatically decide whether to perform a full build or an incremental build based on the changes made.

If you add an external asset or library to the Visualizer project after a full build, you must clean up the project using the **Clean** menu. The **Clean** menu erases files that were generated during the previous build and makes the project ready for the subsequent build. Once the project is cleaned, Visualizer will perform a full build the next time you trigger a build.

If you do not clean the project and perform a build, Visualizer will not take into consideration the assets and libraries added while performing the build.

> ***Note:*** The Incremental build is supported only by the regular build.

Build an iOS Application

Applies to *Kony Visualizer Classic*.

Once you have created and configured an app's assets, resources, and services, you compile and link them by building your app.

Prerequisites

- Configure Xcode on your Mac
- Connect your Mac with Visualizer

This topic covers the following topics:

About App Transport Security (ATS)

Automatically Modify info.plist with Custom Key Value Pairs

Automatically Add Required System Frameworks to the XCode Project

Build an App

About App Transport Security (ATS)

As of January 2017, Apple Corp. requires that all iOS apps make use of App Transport Security (ATS), which implements the https protocol when communicating with internet resources. Introduced in iOS 9, ATS enforces secure connections between internet resources (such as the backend server) and an iOS app, ensuring that internet communications via the iOS platform conform secure connection best practices. Doing so lessens the likelihood of accidentally disclosing sensitive information either directly through an app or through a library consuming data.

Kony Visualizer supports ATS and assumes that 1) your iOS applications make use of https, and 2) your communication through higher-level APIs is encrypted using TLS version 1.2 with forward secrecy. As a workaround to this requirement, if your app needs to make a request to an insecure domain, you have to specify this domain in your app's `info.plist` file. For more information, see the Apple Developer site.

Automatically Modify info.plist with Custom Key Value Pairs

When you build an app for the iOS platform in Kony Visualizer, Xcode generates a build properties file called `info.plist`. By modifying this file, you can customize the properties of the build, which are reflected in the app the next time you build it. Modifications to `info.plist` take priority over whatever properties are set by Kony Visualizer.

If your app requires that the `info.plist` file be configured with sets of key value pairs of native SDK or custom-defined entries, you can automate this customization using a json file. The json file contains the key value pairs that you want added to `info.plist`, and during the building of the app, this json file is incorporated into the kar file. When the kar file is extracted, iOS incorporates the key value pairs in the json fie into `info.plist`.

To automatically modify info.plist with custom key value pairs, do the following:

1. Create a json file with the following file name:
   `infoplist_configuration.json`
2. Place the json file in the following Kony Visualizer project folder:
   <WorkspaceName>\<ProjectName>\resources\common
3. Edit the json file, adding the key value pairs that you want automatically incorporated into the `info.plist` file. Save the file and close it.
4. In Kony Visualizer, build the app.

   The json file is incorporated into the kar file. When the kar file is extracted, iOS incorporates the key value pairs in the json fie into `info.plist`.

Automatically Add Required System Frameworks to the XCode Project

When you use FFIs with Kony Visualizer, if the FFI has dependencies on any system frameworks, you can add those system frameworks to your XCode project. You can do so by configuring a kony_ frameworks.json file.

To add system frameworks to the XCode project, do the following:

1. Navigate to the common folder in your Kony Visualizer project. For example, *<Workspace Name>/<Project Name>/resources/common*
2. In the folder, create a JSON file and name it **kony_frameworks.json**.
3. Open the JSON file in any text editor.
4. Add a JSON object with the key as systemframeworks and value as an array containing all the system frameworks that are dependencies for FFI's. For example,

```
{
"systemframeworks" : ["AVFoundation", ”Security”,
”LocalAuthentication”]
}
```

5. In Kony Visualizer, build the app.
   The JSON file is incorporated into the KAR file. When the KAR file is extracted, system frameworks provided in the JSON is added to the XCode project.

Build an App

To save time and effort, you can rebuild your app with your most recent build settings, or you can build with new or different settings.

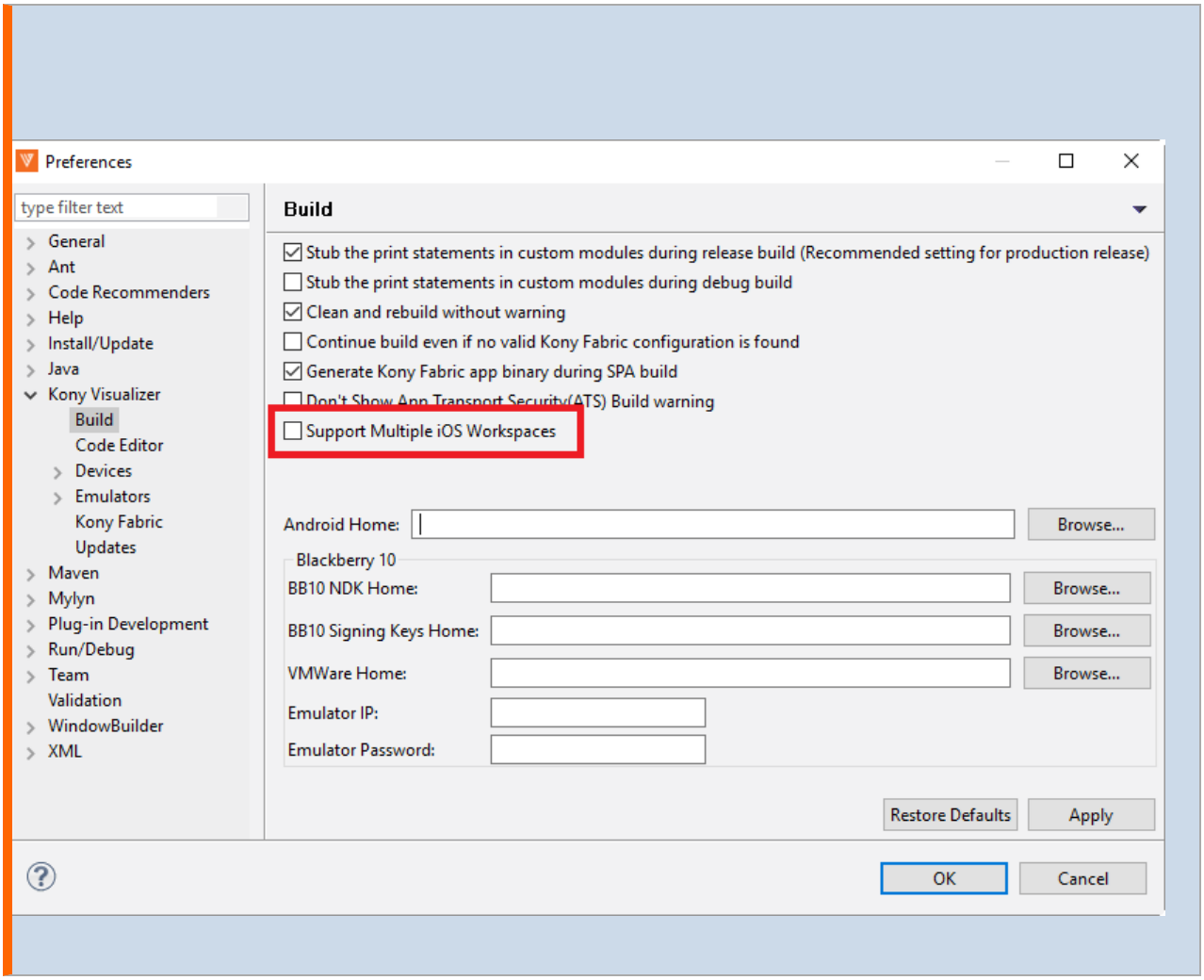
To build an app with your most recent build settings, do the following:

- On the **Product** menu, click **Build Last**.

To build an app with new or different settings, do the following:

1. On the **Product** menu, click **Build**.
2. In the Build Generation dialog box, select the channels and platforms for which you want to build your app. For example, you may want to build a native type of app for Mobile (phone) devices and Tablet devices for the iOS and Android platforms. For more information about native and SPA apps, see Types of Applications.
3. Select the build mode, either debug or release.
    - **Debug mode.** To help you identify and fix errors, Kony Visualizer emits the complete symbolic debug information . To lessen the amount of time necessary to complete the build, the build is not optimized for code execution, so it may tend to execute slower than a build optimized for release. Also, the inclusion of the symbolic debug information causes the final executable to be larger than a release build.
    - **Release mode.** Kony Visualizer optimizes the build for execution, requiring more time to generate the build. It also does not emit the complete symbolic debug information, making the final executable smaller than a debug build.
4. Click **Build**.

Important: If you want to build your iOS app with multiple workspaces, you must select the **Support Multiple iOS Workspaces** from the Build section (**Windows** > **Preferences** > **Kony Visualizer** > **Build**). If this option is not checked, the workspace is replaced for every build. When this option is checked, a separate workspace is created for each application. It will reduce the time and effort required for building applications every time you switch from one application to another.



Build an Android Application

Applies to *Kony Visualizer Classic*.

Once you have created and configured an app's assets, resources, and services, you compile and link them by building your app.

To save time and effort you can rebuild your app with your most recent build settings, or you can build with new or different settings.

To build an app with your most recent build settings, do the following:

- On the **Product** menu, click **Build Last**.

To build an app with new or different settings, do the following:

1. On the **Product** menu, click **Build**.
2. In the Build Generation dialog box, select the channels and platforms for which you want to build your app. For example, you may want to build a native type of app for Mobile (phone) devices and Tablet devices for the iOS and Android platforms. For more information about native and SPA apps, see Types of Applications.
3. Select the build mode, either debug or release.
    - **Debug mode.** To help you identify and fix errors, Kony Visualizer emits the complete symbolic debug information . To lessen the amount of time necessary to complete the build, the build is not optimized for code execution, so it may tend to execute slower than a build optimized for release. Also, the inclusion of the symbolic debug information causes the final executable to be larger than a release build.
    - **Release mode.**Kony Visualizer optimizes the build for execution, requiring more time to generate the build. It also does not emit the complete symbolic debug information, making the final executable smaller than a debug build.
      If you want to sign your android application with your keystore certificate, see Build and Sign an Android application in Release Mode.
4. Click **Build**.

Build and Sign an Android Application in Release Mode

Kony Visualizer V 8.1 introduced a new feature for Android application build where you can generate and sign an Android application in the release mode with a keystore certificate.

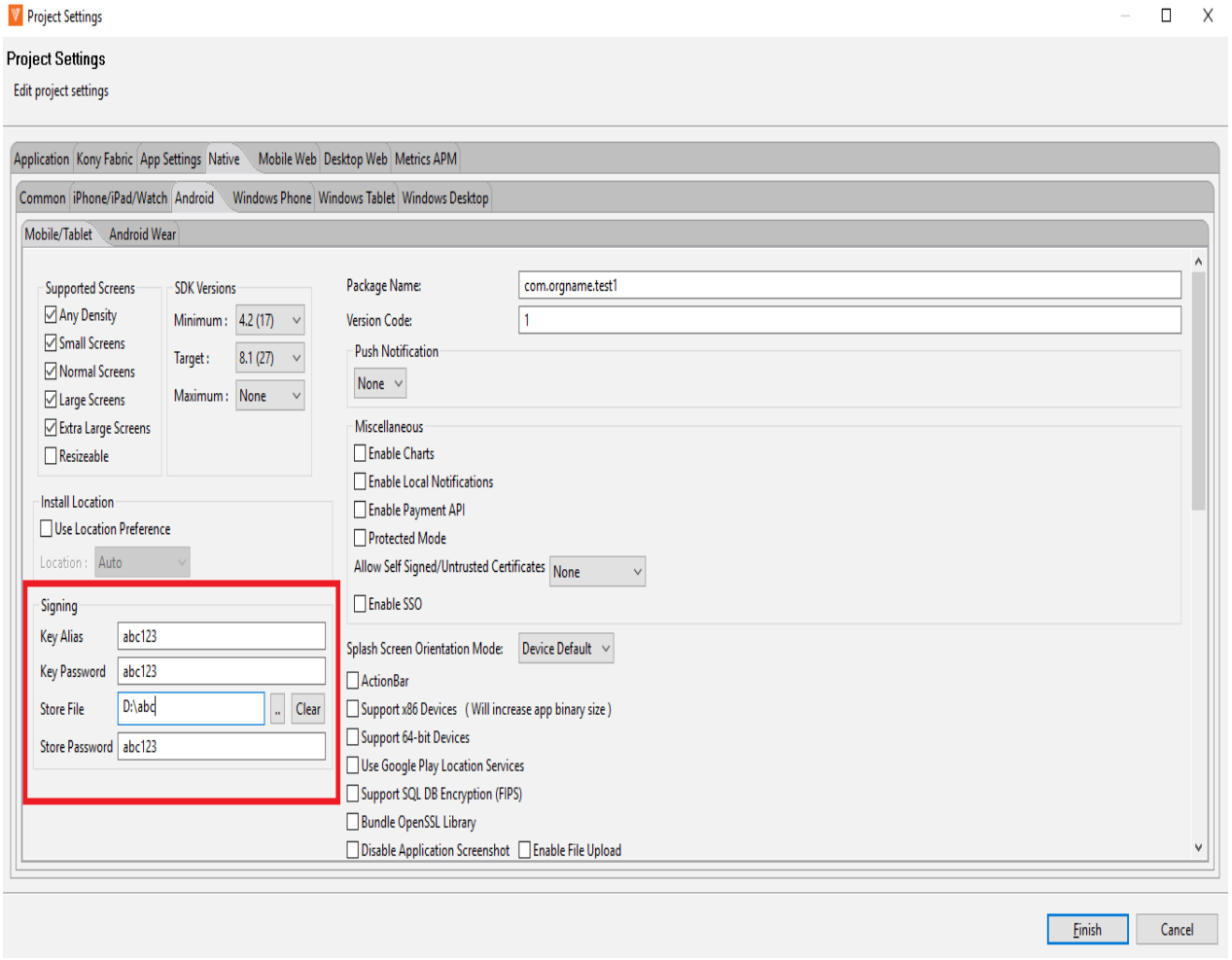
You must configure the settings for the keystore certificate in the Project Settings section.

During the build process in the release mode, keystore configuration from the Project Settings is used to sign the Android binary. Once the Android binary is built with the signing certificate, you can upload the binary to Google Playstore.

To configure the keystore certificate, do the following:

1. On the File menu, click **Settings**.
2. Click the **Native** tab.

3. On the Android tab, in the Signing section, set the following properties:



4. **Key Alias**: Enter the alias for your key.
5. **Key Password**: Enter the password to access the key.
6. **Store File**: Browse to the location of your store and select the store certificate.
7. **Store Password**: Enter the password to access the store.
8. Click **Finish**.

Android SDK and Gradle Dependencies

From Kony Visualizer V8 release, Visualizer downloads the required dependencies before the Android build. If you have started the build for Android for the first time, all required SDK, support packages, and dependent libraries download in the background. Your build is blocked till the time the downloads are complete.

Known Issues:

- When a download dependency is in progress, and the network disconnects in the middle, Android build may hang indefinitely. This is a known technical issue with Gradle. Refer https://github.com/gradle/gradle/issues/868 for more info. You may have to restart visualizer to build again.
- When an SDK component is partially downloaded or corrupted, the build fails. Delete the corrupted SDK component and then proceed with the build.

Updating an Android App

If your Android app is published even once, when you modify the app, you must ensure to change the version of the app from project settings. Modifying the version number is a mandatory step for the changes to reflect in the app.

- From the **File** menu, navigate to **Project Settings**.
- In the **Application** tab, update the version number to the next version. For example, if the **Version** is 1.0.0, modify it to 1.0.1.

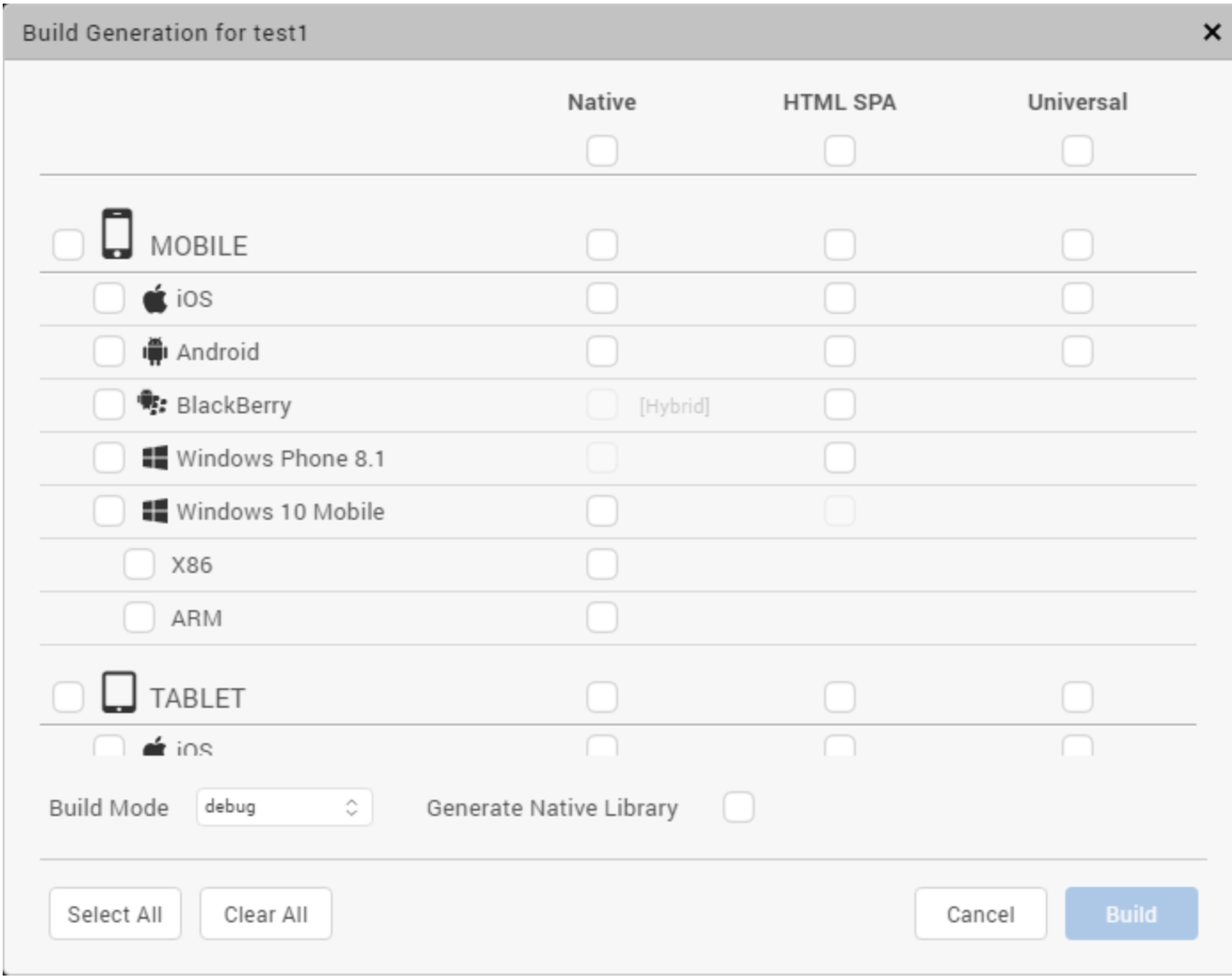
Build a Universal Application

Applies to *Kony Visualizer Classic*.

For iOS and Android platforms, you can build an application in universal binary format, allowing the application to run on both mobile and tablet channels.

To build a universal application:

1. On the **Product** menu, click **Build**.
2. In the Build Generation dialog box, select the channels and platforms in the Universal column for which you want to build your app. For example, to build a universal application for the iOS platform only, select the iOS box in the Universal column for either mobile or table channels:



Kony Visualizer automatically selects the iOS box for both mobile and tablet channels. Similarly, Kony Visualizer automatically selects the Android box for both mobile and tablet channels if you select the Android box in the Universal column. To select mobile and table channels for both iOS and Android platforms, select the top box in the Universal column, or the boxes on the MOBILE or TABLET row.

3. Select the build mode, either debug or release.

    - **Debug mode.** To help you identify and fix errors, Kony Visualizer emits the complete symbolic debug information . To lessen the amount of time necessary to complete the build, the build is not optimized for code execution, so it may tend to execute slower than a build optimized for release. Also, the inclusion of the symbolic debug information causes the final executable to be larger than a release build.

    - **Release mode.**Kony Visualizer optimizes the build for execution, requiring more time to generate the build. It also does not emit the complete symbolic debug information, making the final executable smaller than a debug build.

4. Click **Build**.

Build a Windows 10 Application

Applies to *Kony Visualizer Classic*.

The process of developing and deploying a Kony Visualizer app for the Windows 10 platform consists of three phases:

1. Develop and build the application in Kony Visualizer.
2. Create an APPX file on Windows 10.
3. Deploy the APPX file on a Windows 10 target system.

In the pursuit of this process, this section covers the following topics:

Prerequisites to building applications for Windows 10

Setting up the environment for development

Secure application code

APPX Creation

Create APPX files from zip files (x86/x64/ARM)

Deploying APPX on a computer

- Deploying an x86 zip on a x86/x64 computer
- Deploying an x64 APPX on ax64 computer
- Deploying an ARM APPX on an ARM tablet
- Deploying an ARM Zip on an ARM tablet

Understanding Prerequisites and Processor Architecture

Before building an application for Windows10, ensure you have knowledge about the prerequisites, compatibility matrix, and processor architecture of the machine.

Prerequisites

Following are the prerequisites and compatibility matrix to build applications:

Windows 10

- Kony Visualizer
- Development environment setup (to generate APPX files and test applications on the Windows 10).

> ***Important:*** While developing a Windows app, do not create a variable with the name `uid`.

Compatibility Matrix (APPX File vs. Machine architecture)

The following compatibility matrix shows APPX install possibility in a machine. For example, if you want to install a 64-bit APPX file, you need a 64-bit system.

	x86 machine	x64 machine	ARM machine
x86 APPX	Yes	Yes	No
x64 APPX	No	Yes	No
ARM APPX	No	No	Yes

Processor Architecture

Before deploying an APPX file, you must know the processor architecture of the machine.

To know the processor architecture, follow these steps:

1. Open command prompt.
2. Type **echo %PROCESSOR_ARCHITECTURE%**. A list of results depending on the processor architecture appears.

Processor Architecture	Result
x86 based	x86
x64	AMD64
ARM based	ARM

Alternatively, you can find the architecture type from the system properties.

1. Open **Control Panel > All Control Panel Items > System**.
2. Observe **System Type** under **View basic information about your computer > System**.

The following chart shows the system type and result.

System Type	Result
X86	x86
X64	x64
ARM	ARM

Development Environment Setup

To run the Windows 10 simulator and test applications, you have to download and install **Visual Studio Community 2015 for Windows 10** from Microsoft's website.

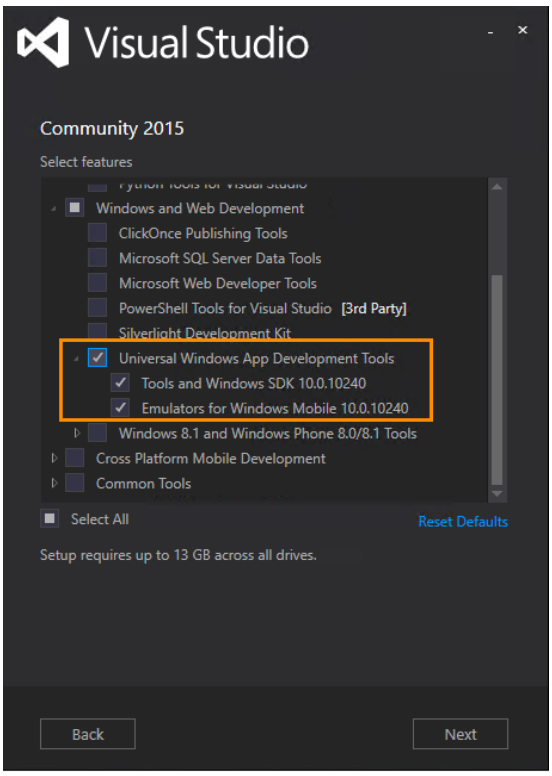
In this section, you will learn about:

1. Preliminary Steps Required to Set Up Development Environment
2. Enable Developer Mode
3. Secure Application Code
4. Restrict the Application to a Device Family
5. Issue in Universal Windows Applications

Preliminary Steps Required to Set Up Development Environment

To setup development environment, follow these steps:

1. Ensure you have a x86 or x64 Windows 10 machine.
2. Ensure that the Universal Windows App Development Tools are selected from the optional features list.



3. After installing this software, you need to enable your Windows 10 device for development.
4. Download and install Visual Studio community 2015 for `Windows 10 vs_community.exe` file from https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx
5. Launch Visual Studio.
6. Complete the trial activation using Visual Studio onscreen instructions.
7. Get a valid developer license.
8. Install Visual C++ Redistributable Windows 8/8.1 (Optional) - Visual C++ Redistributable (for Visual Studio 2012 and Visual Studio 2013) packages install runtime components of visual C++ libraries that are required to run Windows 10.
   Installing the package speeds up the application deployment time.
   a. Visual C++ Redistributable for Visual Studio 2012 (x86) - English : http://go.microsoft.com/?linkid=9815734

b. Visual C++ Redistributable for Visual Studio 2012 (x64) - English : http://go.microsoft.com/?linkid=9815744

c. Visual C++ Redistributable for Visual Studio 2012 (ARM) - English : http://go.microsoft.com/?linkid=9815754

d. Visual C++ Redistributable for Visual Studio 2013 (x86) - English : http://www.microsoft.com/en-us/download/details.aspx?id=40784

e. Visual C++ Redistributable for Visual Studio 2013 (x64) - English : http://www.microsoft.com/en-us/download/details.aspx?id=40784

f. Visual C++ Redistributable for Visual Studio 2013 (ARM) - English : http://www.microsoft.com/en-us/download/details.aspx?id=40784

> ***Important:*** To build Windows 10 applications, either Windows 7, Windows 8/8.1, or Windows 10 can be used. But to run Windows 10 applications, you need a Windows 10 machine.

Enable Developer Mode

To enable developer mode, follow these steps:

1. In Visual Studio, open **Settings** and navigate to **Updates and Security > For Developer**.
2. Click **Enable Developer Mode**.

Secure Application Code

The Windows 10 binary contains platform code, application code (JavaScript) and other resources. Platform code is secured from disassembly when the application is built in Release mode. Windows internally uses native .Net Compilation in release mode and classic .net compilation in debug mode.

To hide the JavaScript code, Kony Visualizer needs to pass *hideSourceCode* property with True as value to the build process. You have to set the flag so that the build process ensures the application is hidden from dis-assembly.

To set hideSourceCode property, follow these steps:

1. Open the Kony Visualizer installation location.
2. Search for **Win10FfiGenerator.exe** in Windows Explorer.
3. Open the folder location.
4. Navigate a level up in the folder structure.
5. Open `build.xml` in an editor.
6. Create a property **hideSourceCode** and set its value to **True** or **False**. For example, `<property name="hideSourceCode" value="true"/>`.
7. Save and close the file.
8. Build the application.

Restrict the Application to a Device Family

Windows 10 allows a single application binary to be deployed and run on multiple device families such as mobile, desktop/tablet, Xbox and IoT. You can choose the binary to be run on any Windows 10 device (of any device family) or you can restrict the binary to be allowed to run on particular device family/families only. Currently, the allowed device families are:

1. **Universal** - an app can be deployed and run on any Windows 10 device.
2. **Mobile** - an app is allowed to be deployed and run on Windows 10 mobiles.
3. **Desktop** - an app is allowed to deployed and run on Windows 10 desktops/tablets.

Kony Visualizer needs to pass this option to platform build scripts. Currently, in 7.0 there is no way to pass this flag. By default, all apps are allowed to run on any Windows 10 device.

If you want to restrict the application to a device family, follow these steps:

1. Open the Kony Visualizer installation folder.
2. Search for Windows 10 plug-in extracted location. Search for **Win10FfiGenerator.exe** in Windows Explorer.
3. Open the folder location.
4. Navigate one level up in the folder structure.
5. Open `build.xml` in any editor.
6. Create a property `win10targetdevicefamily`, and set its value to Universal/Mobile/Desktop. For example, `<property name="win10targetdevicefamily" value="Desktop"/>`.
7. Save and close the file.
8. Build the application.

> ***Note:*** You can also open the manifest file in Visual Studio and reuse the manifest.

Issue in Universal Windows Applications

If you build the application for Windows 10, the build fails for the first time because of extension SDKs.

To resolve this issue, follow these steps:

1. Open **konyApp.sln** once from the Visual Studio so that it installs the required packages for the universal Windows application.

   Location of `KonyApp.sln` file:

   ```
   %Workspace% \ temp \ %Application% \ build \ windows10 \ windows10 \ KonyApp \ KonyApp.sln
   ```

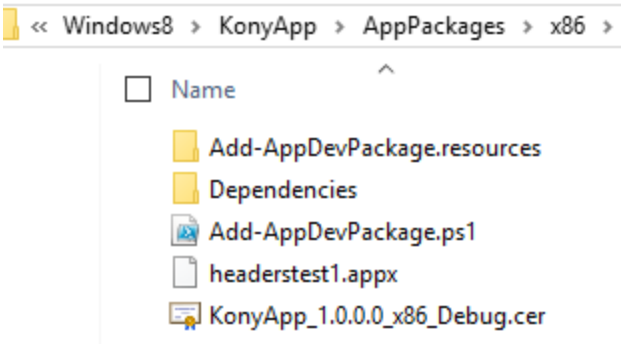
2. Run the application in debug/release for x86/x64 in the local machine. Ensure you have internet

connection at this time to download the required packages.

3. Install the certificate.

APPX Creation

The deployable component for Windows 10 tablet/desktop is an APPX file. If an application is built using Kony Visualizer on Windows 10, the application creates an APPX file with the name `<applicationname>.appx` in target build output folders `<workspace>\temp\<eclipseproject>\build\windows10\windows10\KonyApp\x86|x64|ARM\<applicationname>.appx`.



If an application is built using Kony Visualizer on Windows 7 or Windows 8/8.1 system, a zip file is created with all the required information with name `konywindows10.zip` in the build output folder. The file can help you to create an APPX file.

In this section, you will learn about:

1. Create APPX Files From Zip Files (x86/x64/ARM)
2. Deploying APPX on a Computer
    a. Deploying x86 Zip on x86/x64 Computer
    b. Deploying x64 APPX on x64 System
    c. Deploying ARM APPX on ARM Tablet

d. Deploying ARM Zip on ARM Tablet - Manual

e. Use Different Developer Certificates

3. Build the Windows Application
4. Debug Kony Application in Visual Studio
5. Update Manifest File and Code Signing Certificate
6. Application Submission to the Windows Store

Create APPX Files From Zip Files (x86/x64/ARM)

If an application is built on Windows 7, Windows 8/8.1, or Windows 10, a zip file named `konywindows10.zip` is created with all the required information at:

- <EclipseWorkspace>\temp\<Eclipse project name>\build\windows10\Windows10
- Kony Visualizer copies the zip file to Jetty server.

On a separate Windows 10 system with development configuration, copy the above output folder (through a USB or pen drive). Ensure the extracted folder is not a system folder such as Program Files or Windows. The folder has the following batch file:

- `PackageFromZipFile.bat` - for x86, x64 and ARM.

You can use `PackageFromZipFile.bat` with either of the following options to deploy the application on to a x86/x64/ARM-based Windows 10 machine. Ensure APPX file is created in the same folder. The batch file deploys the application when the APPX file is created.

1. **PackageFromZipFile.bat <jetty url>**
    - URL must be a fully qualified name of the zip file on the Jetty server. Example - `http://10.10.4.56:8888/<appid>r?type=windows10`.

- **Windows 8** - http://localhost:8888/<appid>r?type=windows8
- **Windows 10** - http://localhost:8888/<appid>r?type=windows10

2. **PackageFromZipFile.bat <zipfile name>**
   - App developers must copy the zip to the extracted folder before using the command.
   - Example - PackageFromZipFile `konywindows10.zip debug|release x86|x64|arm`.

Deploying APPX on a Computer

APPX files can be deployed to the Windows 10 machine (x86/x64/ARM) as mentioned in the compatibility matrix.

1. Open the target output folder, and ensure the APPX file is present in the folder.
2. Run `Add-AppDevPackage.ps1` with PowerShell on device.
3. Open PowerShell with elevated permissions, navigate to `Windows10\KonyApp\x86|x64|ARM` folder, and execute `Add-AppDevPackage.ps1`.
4. After you see a successful installation in the PowerShell, launch the application from the Start menu.

Deploying an x86 Zip on x86/x64 Computer

If the application is built on Windows 7 or Windows 8/8.1, `konywindows10.zip` file is created at `<EclipseWorkspace>\temp\<Eclipse project name>\build\windows10\Windows10`.

1. Follow the APPX creation process in APPX creation for x86 systems.
2. When the APPX file is created, continue from step 3 in section Deploying an APPX on respective system.

Deploying an x64 APPX on x64 System

To deploy an x64 APPX on an x64 machine, follow these steps:

1. The application must be built on a Windows 10 system, and x64 must be selected in the Build option.
2. Open the build folder and navigate to the **Windows10** folder. Ensure the APPX file is in the folder.
3. Copy the **Windows10** folder to the x64-based system.
4. If the APPX file is present in the target folder, continue from step 3 in section Deploying an appx on respective system.

Deploying an ARM APPX on ARM Tablet

To deploy an ARM APPX on an ARM tablet, follow these steps:

1. The application must be built on a Windows 10 system for ARM configuration.
2. Open the build folder and navigate to the Windows10 folder.
3. Ensure the APPX file is present.
4. Copy the **Windows10** folder to surface/tablet with a USB/use network.
5. If an APPX file is present in the target folder, continue from step 3 in Deploying an APPX on respective system.

Deploying an ARM Zip on ARM Tablet - Manual

If an application is built on Windows 7 or Windows 8/8.1 the `konyWindows10.zip` file is created at `<EclipseWorkspace>\temp\<Eclipse project name>\build\windows10\Windows10`.

To deploy an ARM zip file on an ARM tablet, follow these steps:

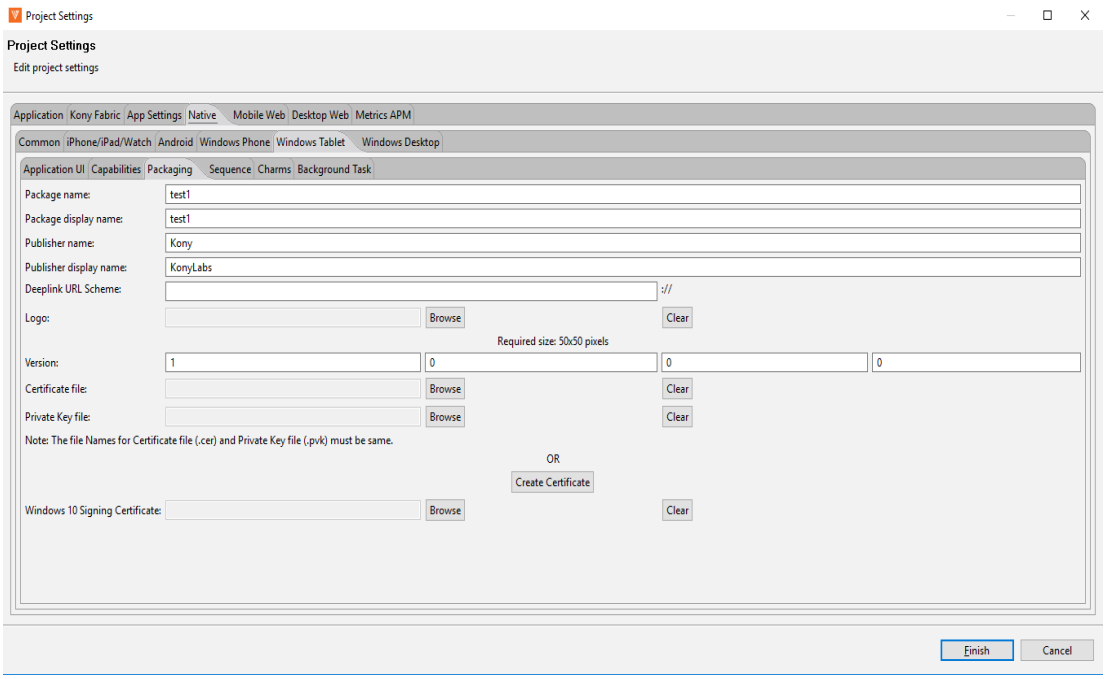
1. Follow the APPX creation process in APPX creation for ARM systems.
2. When the APPX file is created, continue from step 3 in section Deploying an APPX on a machine.

Use Different Developer Certificates

Do not use Kony developer certificate to sign the customer application. The Kony developer certificate should be used only during the application development. When submitting the application to the store, you can use your own certificate to sign the APPX package.

To use different developer certificates, follow these steps:

1. Right-click on an application in Eclipse, and open the properties page.
2. Navigate to **Native App > Windows Tablet > Packaging**.



3. To use the existing certificate file, use the **Certificate file** section, and select the required PFX certificate file. Ensure the publisher name of the existing certificate matches the publisher name

provided previously. The build fails if the publisher names does not match.

a. To create the certificate on your own see these articles:
   i. Article 1
   ii. Article 2

b. To create a certificate on your own, the following command can be used to create the certificate file. The .cer and .pvk file need to be used to generate the .pfx file, which is provided as input to Certificate file section in Kony Visualizer.

c. Execute the below command line with makecert.exe

```
makecert -cy end -eku 1.3.6.1.5.5.7.3.3 -sv yourcompany.pvk -n "CN=yourcompany" yourcompany.cer -b mm/dd/yyyy -e mm/dd/yyyy -r
```

d. Execute the below command line with pvk2pfx.exe to get the certificate pfx file

```
pvk2pfx -pvk yourcompany.pvk -spc yourcompany.cer -pfx yourcompany.pfx
```

4. To create a new certificate, see Create Your Own Test Certificate.
5. Build the application.

Build the Windows Application

Once you have created and configured an app's assets, resources, and services, you compile and link them by building your app.

To save time and effort you can rebuild your app with your most recent build settings, or you can build with new or different settings.

To build an app with your most recent build settings, do the following:

- On the **Product** menu, click **Build Last**.

To build an app with new or different settings, do the following:

1. On the **Product** menu, click **Build**.
2. In the Build Generation dialog box, select under each applicable **channel**[1] the Windows versions you are configured to build for, along with the hardware architecture, and environment (e.g. Native, HTML SPA). For more information about native and SPA apps, see Types of Applications.
3. Select the build mode, either debug or release.
    - **Debug mode.** To help you identify and fix errors, Kony Visualizer emits the complete symbolic debug information . To lessen the amount of time necessary to complete the build, the build is not optimized for code execution, so it may tend to execute slower than a build optimized for release. Also, the inclusion of the symbolic debug information causes the final executable to be larger than a release build.
    - **Release mode.**Kony Visualizer optimizes the build for execution, requiring more time to generate the build. It also does not emit the complete symbolic debug information, making the final executable smaller than a debug build.
4. Click **Build**.

> ***Note:*** While building a Windows application, you cannot build the app for other platforms. This is a limitation of Windows emulator.

Debug Kony Application in Visual Studio

Kony helps you launch and debug applications in Visual Studio.

Debugging an application involves two steps:

1. Open the Kony Application in Visual Studio
2. Debug the Kony Application

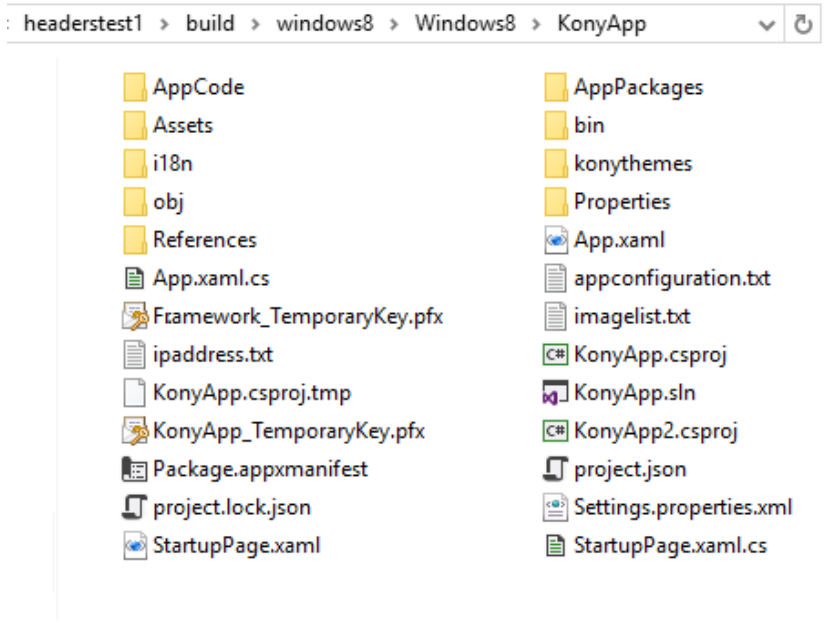
---

[1]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

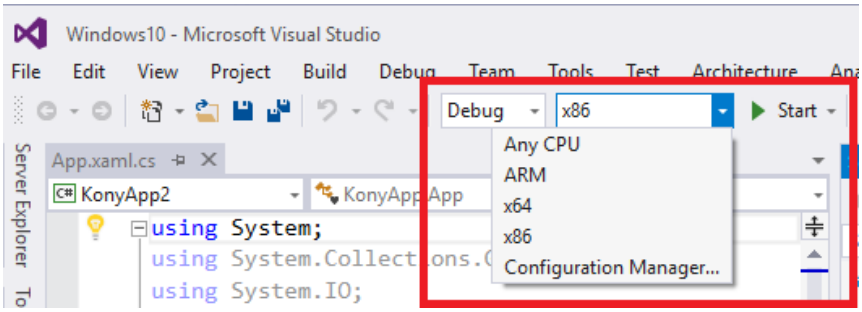
Open the Kony Application in Visual Studio

To open a Kony application in Visual Studio, follow these steps:

1. Open Visual Studio Community 2015. You can use any Visual Studio version that can open universal projects.
2. Navigate to the build folder (<workspace>\temp\<eclipseproject>\build\windows10\Windows10) and find the **KonyApp** folder.

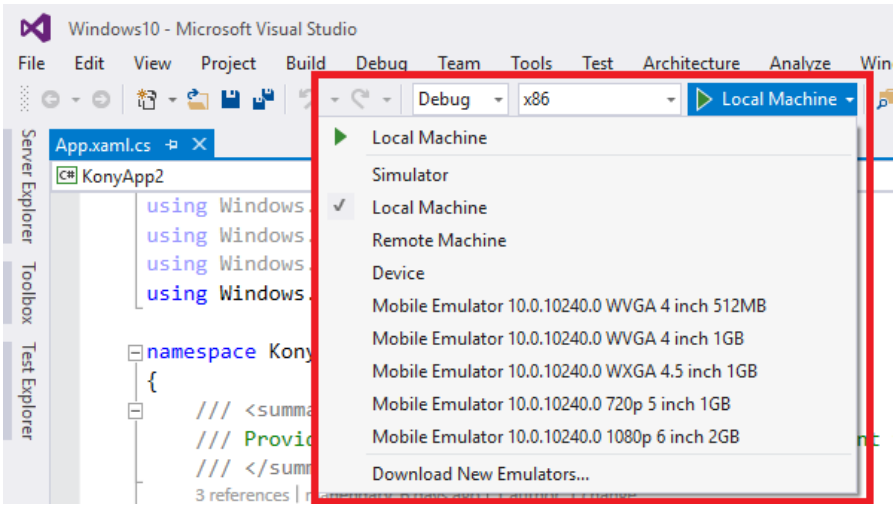


3. Open **KonyApp.sln** in Visual Studio.
4. Change the configuration to x86, x64, or ARM depending on the requirement. The configuration must not be **Any CPU**. App developers can select the application to be deployed in Debug or Release mode by choosing the required value in the drop-down list.



5. Right-click **KonyApp (Universal Windows)** project and select **Set as Startup Project**.

6. Choose the right simulator/emulator/local machine to deploy the application.

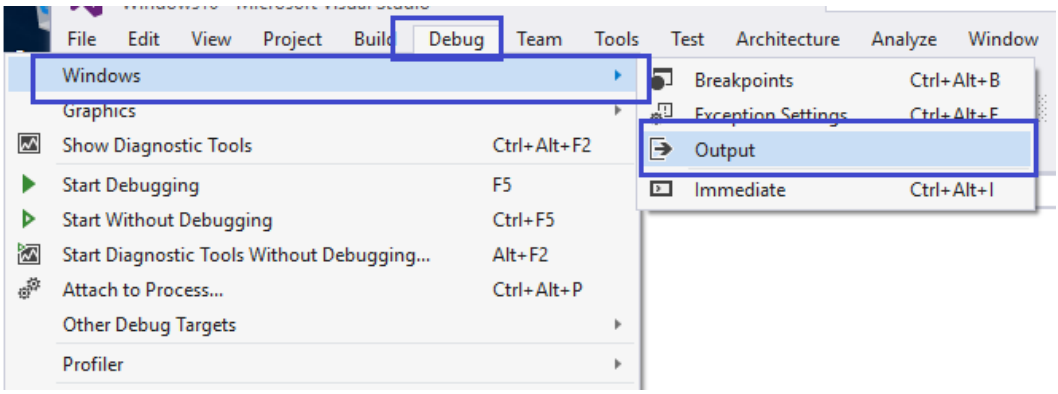


Debug the Application

To debug the application, follow these steps:

1. Press F5 on the keyboard, or select **Start > Visual Studio > Debug > Start debugging**.

2. To see the application logs, open the output window under **Debug > Windows > Output**.



3. You can use other debugging tools available in Visual Studio to debug the application like any other native Universal Windows application. The output window will not show logs when the release option is selected.

Update Manifest File and Code Signing Certificate

You can open the solution folder, and make changes to the app manifest file. You must place the updated manifest file in the **Resources** folder so that the build process picks and uses for application build.

> ***Note:*** After you place the new manifest file in resources, any change done in Kony Application properties page will not be considered as the files provided by user replace all Kony Application Properties page.

To update the manifest file and code signing certificate, follow these steps:

1. Launch Visual Studio.
2. In Solution Explorer View, open the **Package.appxmanifest** file in design or XML view.
3. For more information, see the following links:
    a. https://msdn.microsoft.com/en-us/library/windows/desktop/br211474.aspx.
    b. https://channel9.msdn.com/Series/Windows-Phone-8-1-Development-for-Absolute-

Beginners/Part-8-Working-with-the-package-appxmanifest

c. https://msdn.microsoft.com/en-us/library/br230260.aspx

Application Submission to the Windows Store

To create the app package, follow the Microsoft MSDN guidelines at https://msdn.microsoft.com/en-us/library/hh454036.aspx. You can also refer, https://msdn.microsoft.com/en-us/library/windows/hardware/dn265142%28v=vs.85%29.aspx.

Build an SPA Application

The process of building an app differs between Kony Visualizer and Kony Visualizer Classic. Select the procedure for the edition you're using.

Build an SPA Application for Kony Visualizer

Build an SPA Application for Kony Visualizer Classic

Build an SPA Application for Kony Visualizer

To build an SPA application for Kony Visualizer, do the following:

1. Indicate that you want to build the app for the SPA environment. To do so, on the Edit menu, click Preferences.
2. On the left pane of the dialog box, click Functional Preview, and then in the right pane, click the HTML 5 SPA checkboxes for the platforms and channels that you want to generate an SPA build of. Click **Apply**.
3. On the **Run** menu, click **Run**.
4. After the build is completed, open your browser. Ensure that it is in Developer mode.
5. Open the app. The URL for doing so should be as follows:
   `localhost:9989/<ProjectName>/p`

For Desktopweb, the URL should be: `localhost:9989/<ProjectName>/kdw`

Build an SPA Application for Kony Visualizer Classic

To save time and effort you can rebuild your app with your most recent build settings, or you can build with new or different settings.

Click here to watch a video on building an SPA application for Kony Visualizer.

To build an app with your most recent build settings, do the following:

- On the **Product** menu, click **Build Last**.

To build an app with new or different settings, do the following:

1. On the **Product** menu, click **Build**.
2. In the Build Generation dialog box, select the channels and platforms for which you want to build your app. For example, you may want to build a native type of app for Mobile (phone) devices and Tablet devices for the iOS and Android platforms. Also, select the HTML SPA checkboxes among the relevant platforms and channels that you are building for. For more information about native and SPA apps, see Types of Applications.
3. Select the build mode, either debug or release.
    - **Debug mode.** To help you identify and fix errors, Kony Visualizer emits the complete symbolic debug information . To lessen the amount of time necessary to complete the build, the build is not optimized for code execution, so it may tend to execute slower than a build optimized for release. Also, the inclusion of the symbolic debug information causes the final executable to be larger than a release build.
    - **Release mode.** Kony Visualizer optimizes the build for execution, requiring more time to generate the build. It also does not emit the complete symbolic debug information, making the final executable smaller than a debug build.

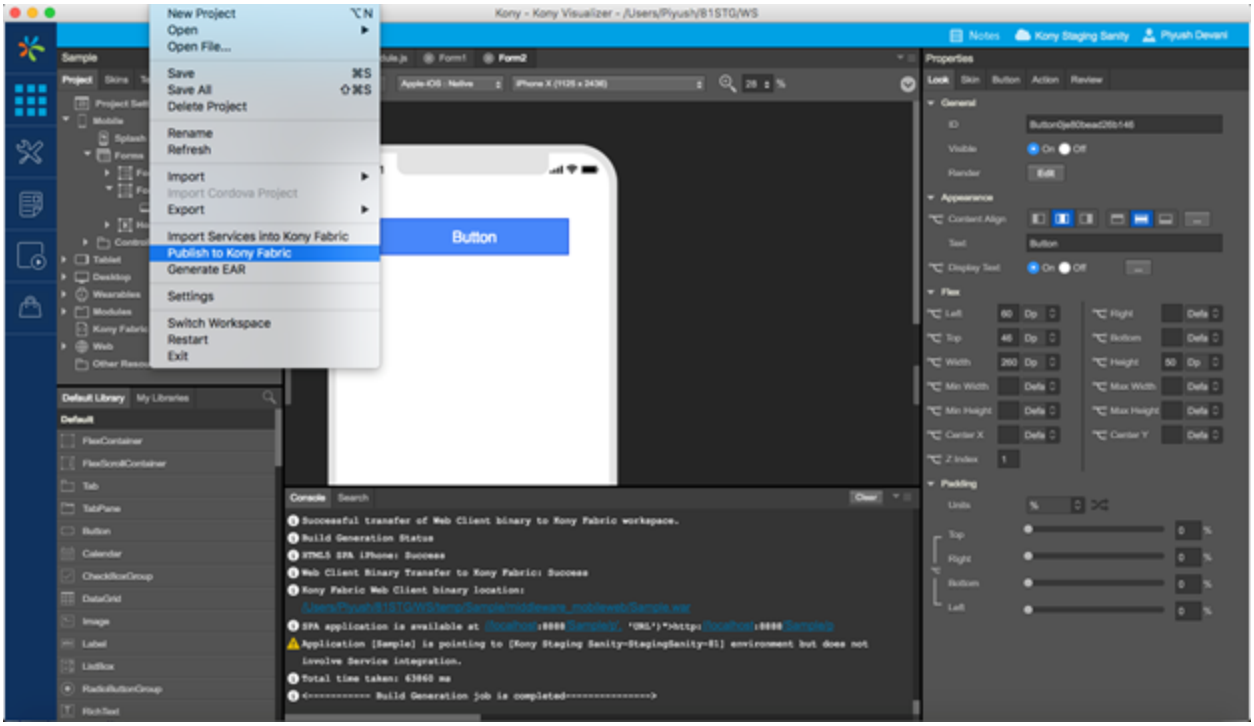
4. Click **Build**, and address any dialog boxes that appear. When the build finishes, a dialog displays indicating the URL to use to view the app.

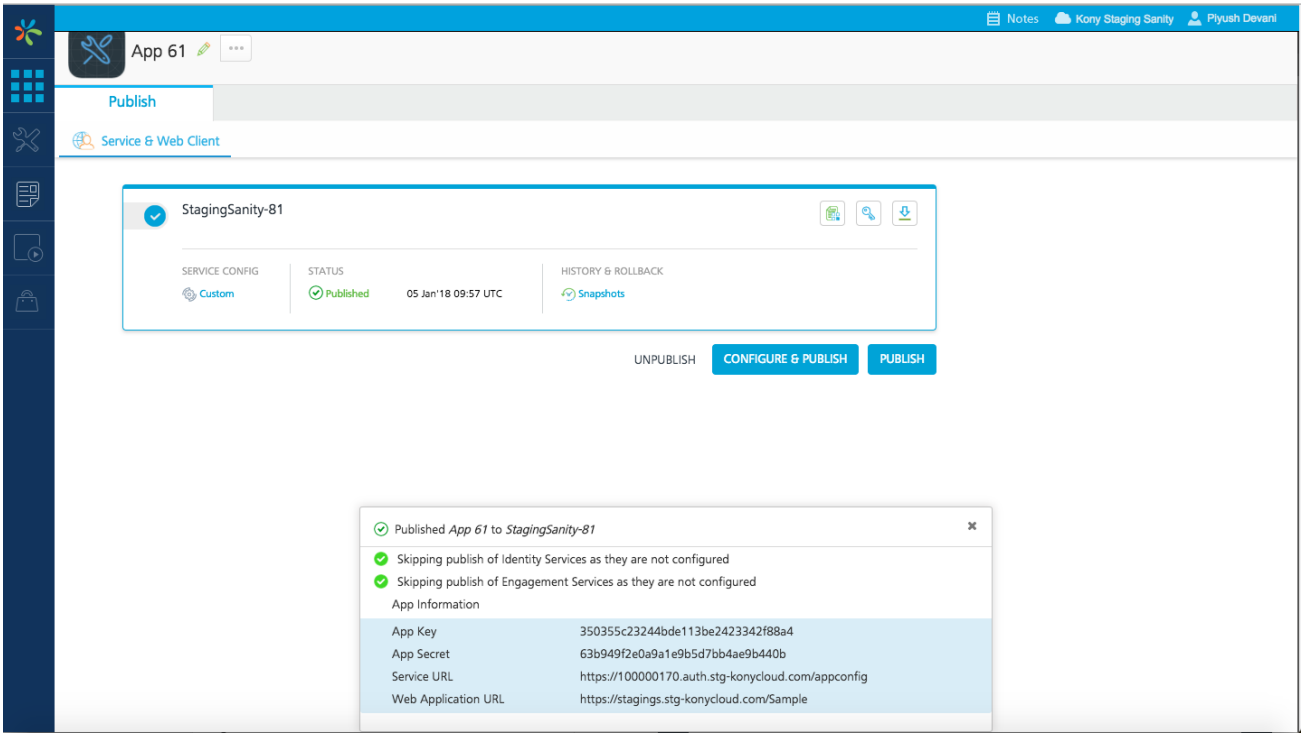
   > ***Note:*** You can also locate the URL in the Console log.

5. After the build is completed, open your browser. Ensure that it is in Developer mode.

6. Using the URL provided by Kony Visualizer, open the app. The URL for doing so should be as follows:
   `localhost:8888/<ProjectName>/p`
   For Desktopweb, the URL should be: `localhost:8888/<ProjectName>/kdw`

7. If your SPA app is published to Kony Fabric runtime server, the URLs of the SPA app are printed on the Kony Fabric Publish section. To access the publish page, navigate to File > Publish to KonyFabric. For more details, see the images below:



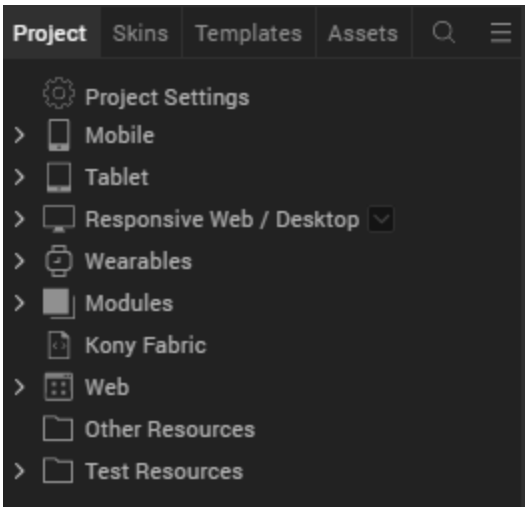


Web App Compatibility Mode

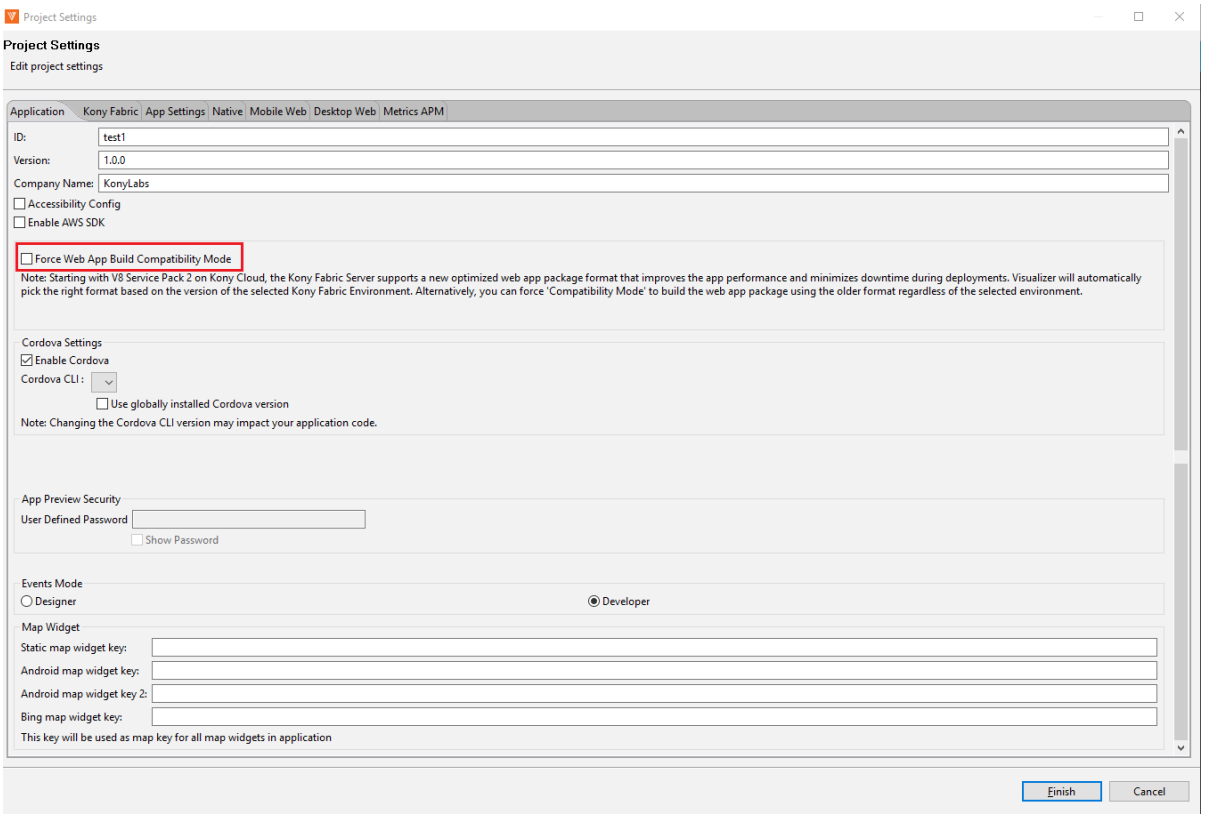
Starting with Kony Visualizer V8 Service Pack 2 on Kony Cloud, the Kony Fabric Server supports a new optimized web app package format that improves the app performance and minimizes downtime during deployments. Visualizer automatically picks the right format based on the version of the selected Kony Fabric Environment. Alternatively, you can force the Compatibility Mode to build the web app package using the older format irrespective of the selected environment. You can choose Compatibility mode from the Project Settings pane.

To select compatibility mode, do the following:

1. In Kony Visualizer, from Project Explorer, select **Project Settings**.



2. In the Project Settings pane, select **Force Web App Build Compatibility Mode**.



3. Click **Finish**.

Securing your Web Applications

Client- side attacks leave your web applications vulnerable, allowing attackers to steal data. Obfuscators protect your apps from reverse engineering and malware attacks. Using Kony Visualizer V8 SP4, you can create a post-build hook for your SPA or Desktop Web applications.

> ***Note:*** Ensure that the Kony application is working before you implement obfuscation. Also, make sure that the obfuscation of the web artifact works before importing it into Kony Visualizer.

To implement obfuscation in your web apps, do the following:

1. Navigate to the location of your project. For example, `<your workspace folder>/<appid>/.`
2. Create a New Folder with the name **custombuild**.
3. In the **custombuild** folder, create a new file, **build.xml**.
4. In the build.xml file, create an ANT task, **postbuild**.
    a. In the postbuild task, write a code that implements an obfuscation of your choice.
    b. After you implement an obfuscation, write a code to replace the existing artifact in the web artifact folder with the protected artifact.
    The protected artifact is generated as the output from the obfuscator tool.

Here is an example of the contents of the **build.xml** file with the **postbuild** task:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="CustomBuildTask" basedir=".">
    <target name="postbuild"  description="post build for
spa/desktopweb">
        <echo message="Post Build Started for Project ::
```

```
${projname}" />
        <!-- Code to generate protected artifact from chosen
obfucator and replacing the existing webartifact -->
        <!-- App Developer Code start -->
        <exec executable="cmd" failonerror="true">
            <arg line="${project.loc}\custombuild\somebatch.bat
--app ${webartifactpath} -- ANY OTHER INPUTS FOR YOUR
OBFUSCATOR" />
            <redirector output="${basedir}\protected_ob.log"
alwayslog="true"></redirector>
        </exec>
        <!-- start error message code - to halt the system when
any error occurs -->
        <loadfile srcfile="${basedir}\protected_ob.log"
property="errorline">
            <filterchain>
                <linecontains>
                    <contains value="ERROR, UNEXPECTED
EXCEPTION"></contains>
                </linecontains>
            </filterchain>
        </loadfile>
        <fail message="Unable to obfuscate - ${errorline}">
            <condition>
                <contains string="${errorline}"
substring="ERROR, UNEXPECTED EXCEPTION"/>
            </condition>
        </fail>
        <!-- error message Code end -->
        <!-- replacing old artifact with protected artifact
start-->
        <move  file="${webartifactfolder}
```

```
/${projname}.${webartifacttype}" tofile="${webartifactfolder}
/${projname}-old.${webartifacttype}"/>
        <move  file="${basedir}/protected_
${projname}.${webartifacttype}" tofile="${webartifactfolder}
/${projname}.${webartifacttype}"/>
        <!-- replacing old artifact with protected artifact
end-->
        <!-- App Developer Code end -->
    </target>
</project>
```

5. Once you create the ANT task, save and close the file.

> ***Note:*** Ensure that you exclude any variables with global scope from obfuscation.

6. Build the project. After the project is built, the protected binary is uploaded to Kony Fabric. Once the build is completed, you must publish the app to your Fabric environment. Once you publish the app, you will get the Web Application URL.

Build a Progressive Web App

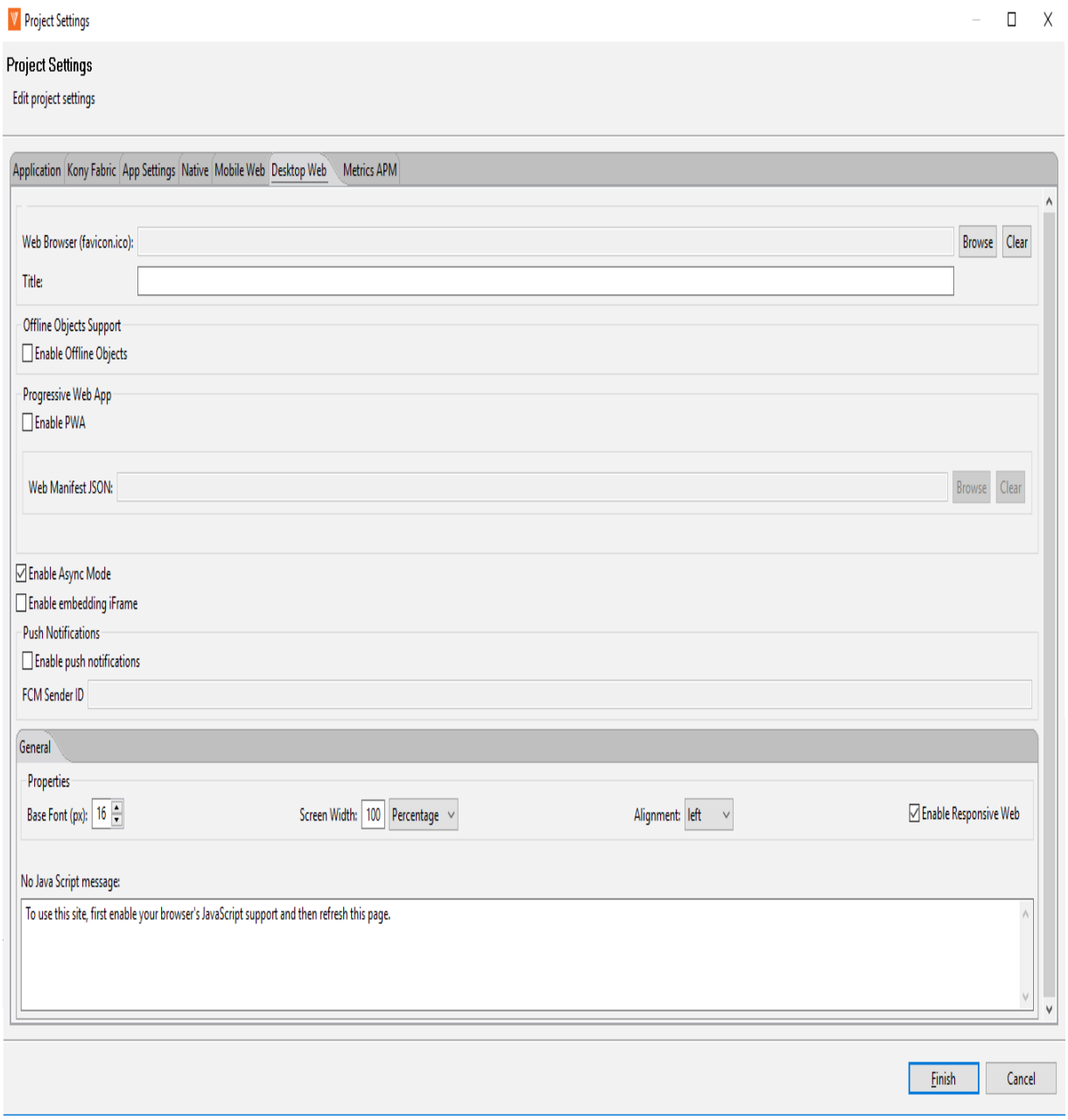
A Progressive Web App (PWA) is the next step of a Responsive Web app. A Progressive Web App is an application that feels like a Native app, but is available over web browsers to a user. A user can access a Progressive Web App on any web browser on a mobile device. The app is responsive, functions even when the device is offline or has limited network speed, and does not require any updates.

For more information about Progressive Web Apps, click here.

> ***Important:*** Before you start to build a Progressive Web App, you must enable the PWA option for a Responsive Web application. When you do so and build the application, a new output is created in the Progressive Web App format. This new output does not replace any Responsive Web output, but instead a new output is created in the same folder. For more information about Responsive Web apps, click here.

To build a Progressive Web App, follow these steps:

1. In Kony Visualizer, click **Project Settings**. The **Project Settings** dialog box appears.



2. Click the **Desktop Web** tab. The Desktop Web contents appear.
3. Select **Enable PWA**. The **Web Manifest JSON** field is enabled.
4. If you want to add information about the Progressive Web app with a manifest file, select **Browse**. The file explorer opens.

5. Go to the folder where the JSON file is located, and then select your Desktop Web manifest file. The Web Manifest JSON file contains information on the resources that the Progressive Web App requires. The information can be details such as name of the app and app icons. Click here for a sample.
6. Click **Finish**.
7. From the Kony Visualizer **File** menu, go to **Product** > **Build**.
   The Build page appears.
8. Under the **Desktop** section, select **Responsive Web**, and then click **Build**. The build process starts.

Once the build is completed in Release mode, you must publish the app to your Fabric environment with HTTPS. Once you publish the app, the Web Application URL is your Progressive Web App.

For a more hands-on approach on the Progressive Web Apps feature provided by Kony AppPlatform, import and preview the Events, Employee Directory, and Resort Feature sample apps by using Kony Visualizer.

- **Events app**: 

- **Employee Directory app**: 

- **Resort Feature app**: 

Live Preview

Overview

Live Preview, a feature introduced in Kony Visualizer V8 SP4, provides a seamless in-app preview experience within Kony Visualizer. Live Preview reduces the build and preview time of an adaptive web app. It enables you to view your app as it appears on various devices without having to view the app on those devices.

Faster Previews:

Before V8 SP4, if one had to view a web app, they had to build the app, publish the app, get the URL, and open the URL in a web browser to test it.

From V8 SP4, this multistep process has been simplified. Now, when a developer selects the Live Preview option, Visualizer builds the app. Once the build is complete, a Visualizer Preview window appears which displays the web app.

Preview a Web app on Different Devices on a Single window:

Using Live Preview, you can switch between various Devices to experience their different form factors. From the Visualizer Preview window, you can select the device from a drop-down list.

When you select a specific device, the previewer instantly displays the web app mimicking the view on the device.

In-app Debugging:

Provides you with a debugger to detect and diagnose errors in your applications. You can set breakpoints and step through the application's code using the in-app debugger.

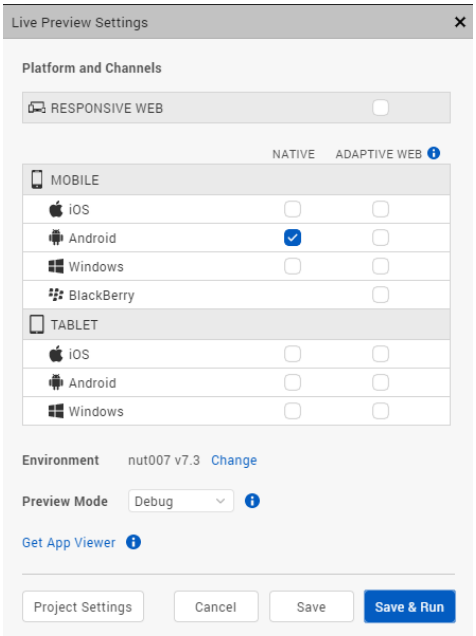
Preview your Web App with Visualizer

> *Important:* Before you run a Live Preview, ensure that you have a project in which you have a Web app designed.

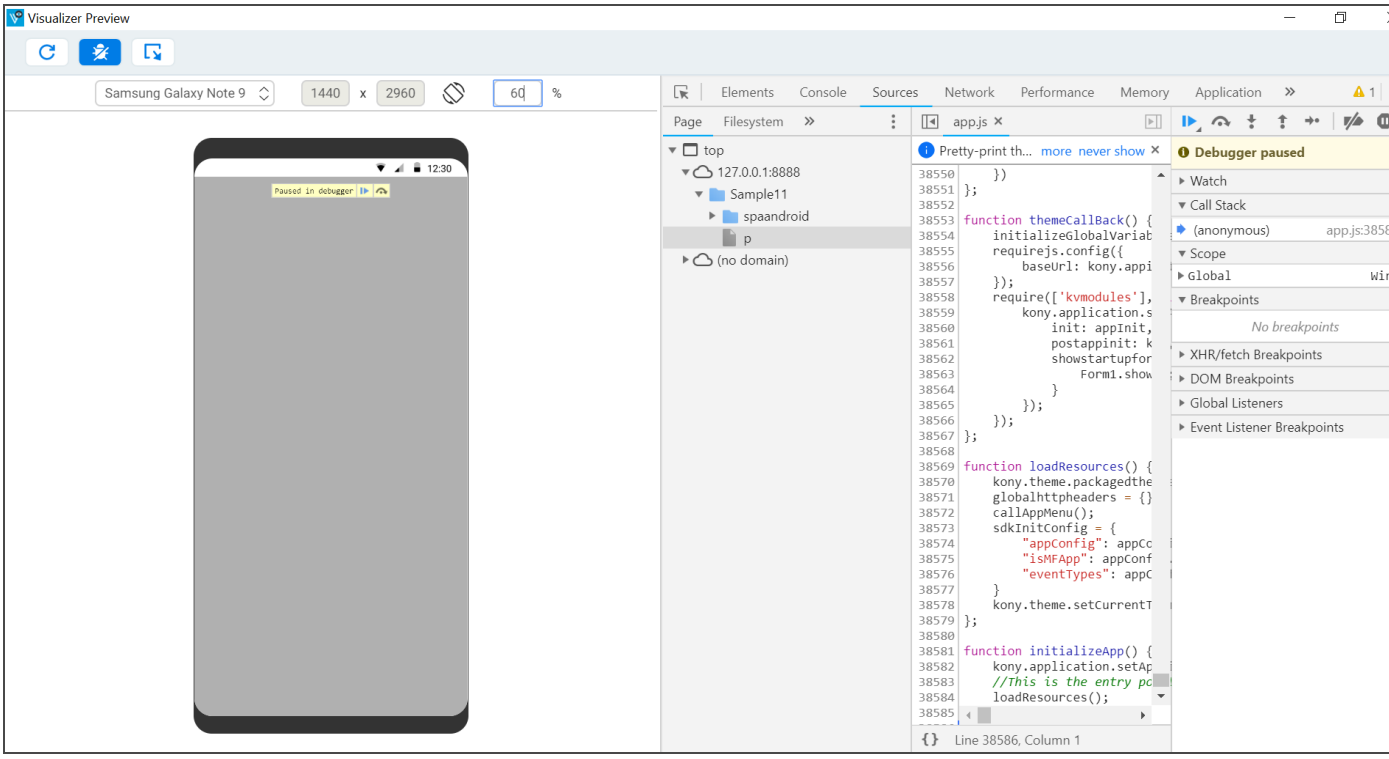
To preview an Adaptive Web App, do the following:

1. Open the Kony Visualizer project in which you have your web app designed.
2. You can perform any of the following actions:
    - For Kony Visualizer Classic: From the main menu, go to **Preview** > **Configure Channels**.
    - For Kony Visualizer: From the main menu, go to **Build** > **Live Preview Settings**.

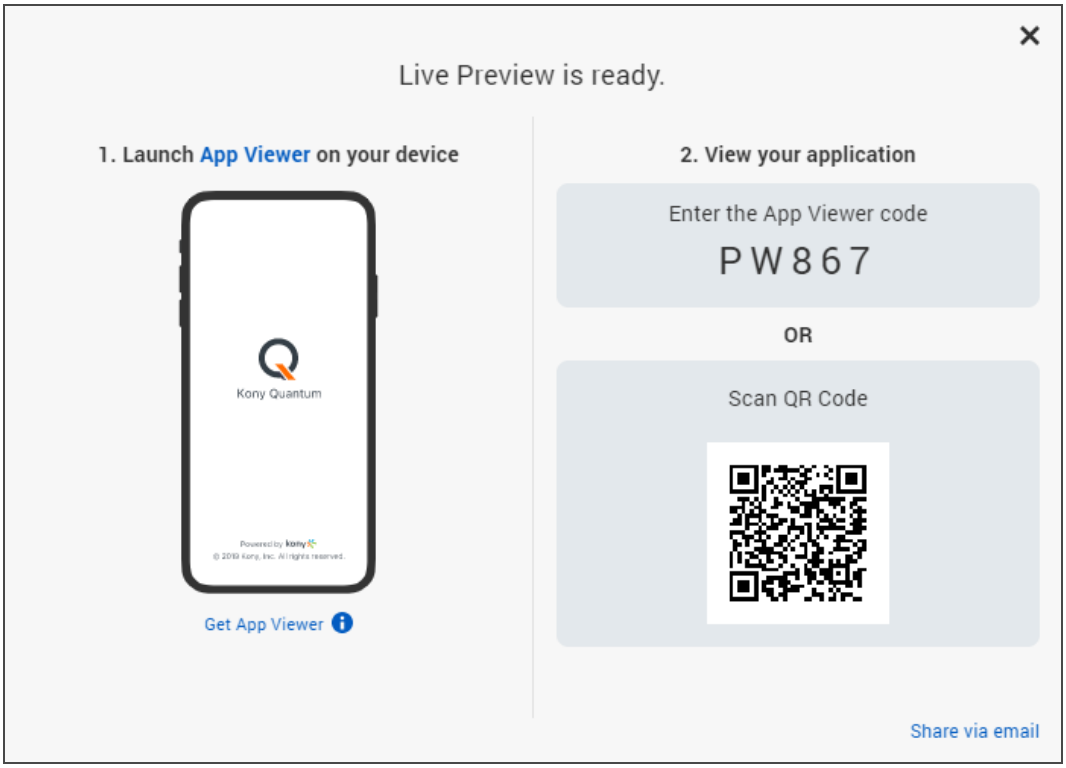
The **Live Preview Settings** window appears.



3. Select the **Adaptive Web** channel for all the required platforms.
4. Click **Save & Run** or click **Save** and press **Ctrl+R** (**CMD+R** on Mac) on your keyboard to launch the Live Preview.
   The build process begins in the Visualizer Build tab.
   Once the build for the selected platforms is completed, a new **Visualizer Preview** window appears.
5. From the Visualizer Preview window, you can select specific platform models from the drop-down menu and adjust their dimensions as required.



In Visualizer, you can see a **Live Preview is ready** window. The window contains App Details and QR code for the web app. Using the details, you can preview the web app on your device. For more details on how to view the app on your device, go to Preview an App on a Device.

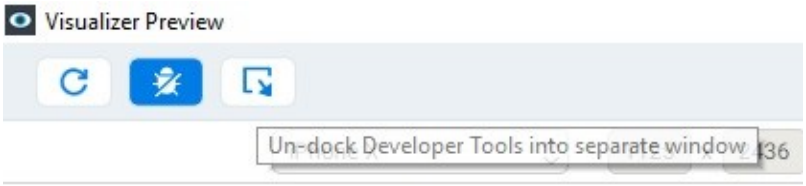


Debugging in Visualizer Live Preview

Kony Visualizer V8 SP4 comes with an in-app debugger that can detect and diagnose errors in applications. Adaptive Web apps can be debugged within the Visualizer Preview window using the latest Visualizer debugger.

After launching the Live Preview, a debugger window automatically appears within the Visualizer Preview window.

You can also open the debugger in a separate window by clicking on the **Un-dock Developer Tools into separate window** button on the upper-right corner of the Visualizer Preview window.



The debugger allows you to control the execution of your application by:

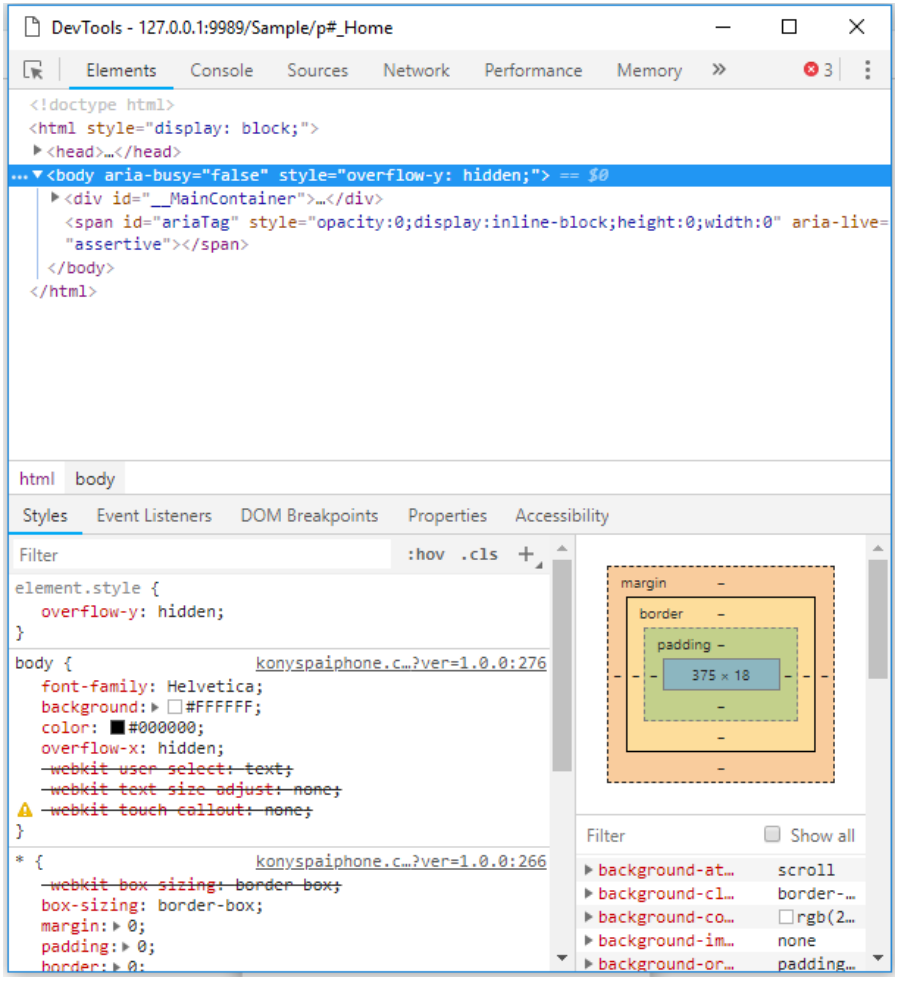
- Setting breakpoints.
- Suspending launched applications.
- Stepping through your code.
- Examining the contents of the variables.

Debugging an application involves launching the application in the Live Preview mode on your desktop.

You can then use the in-app Google Chrome debugger to debug the application. For information on using the Chrome debugger, see Get Started with Debugging JavaScript in Chrome DevTools on the Google Chrome website.

To understand how to Debug an application for iOS and Android platforms you can refer to:

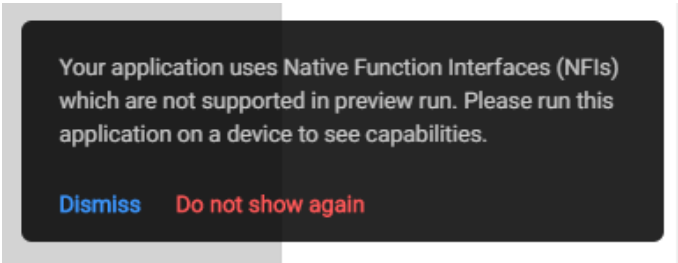
- Debug JavaScript for iOS in Kony Visualizer
- Debug JavaScript for Android in Kony Visualizer



Limitations

The Live Preview feature has the following limitations:

- It does not support viewing the functionality of the NFIs/FFIs in your app. If your app contains an NFI or a FFI or any third-party library dependency an error message is displayed.



- It does not support the execution of APIs that require a handheld device, for example: Vibration API.
- You must still view your Native applications on a handheld device.

Generate Native Library for an App

Applies to *Kony Visualizer Classic*.

You can build a native library for an app that is created in Kony Visualizer. Once you generate the native library, you can embed the native library in your platform specific native applications. For example, if you generated a native library for Android, you can use the library in Android native application.

The Native Library feature of Kony Visualizer uses some Kony application APIs and a few native APIs that establish the communication between Kony library and the native application.

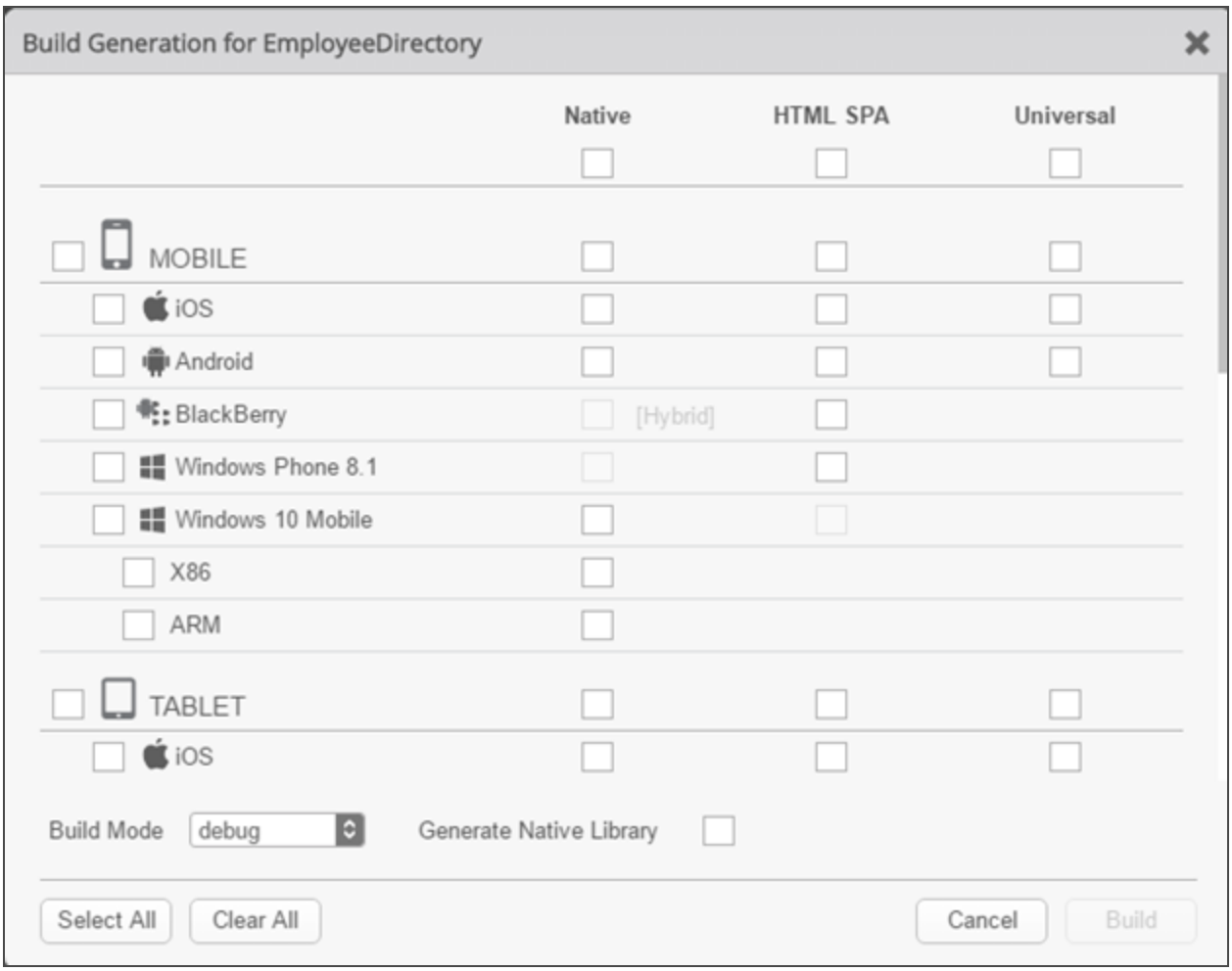
- Communication API for Native Library

Click here to watch a video on generating a native library for an app in Kony Visualizer.

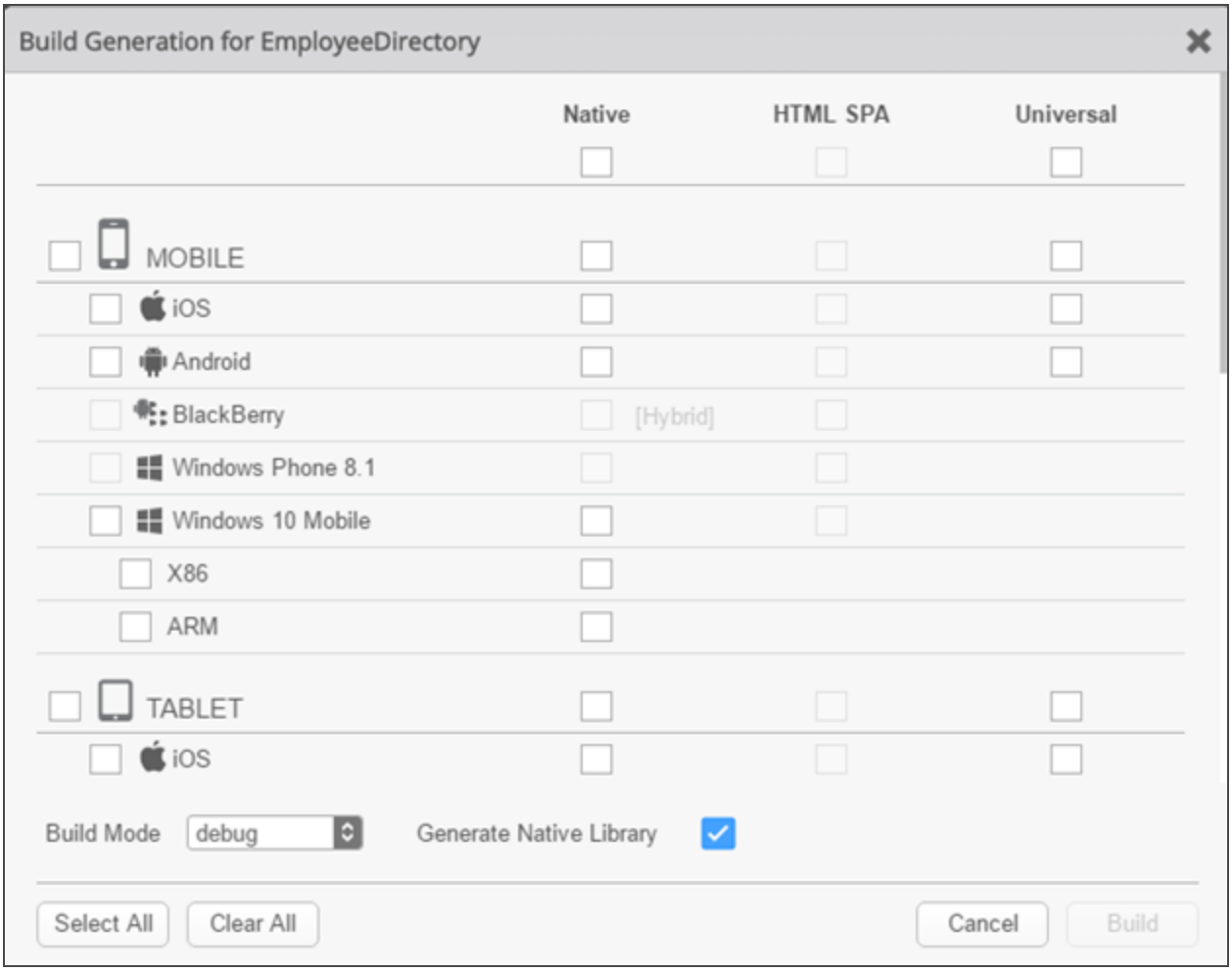
The iOS library is created in the .zip format while the Android library is created in the .aar format. If there are any errors during the process of library creation, the Console section of Kony Visualizer logs the issues.

To generate native library for your app, do the following:

1. In Kony Visualizer, from the File menu, click Product, and then Build.
   The Build generation page displays.



2. Select **Generate Native Library**.
   Those builds which do not support generating Native library are disabled.



3. Select the channel you want to generate the native library and then click Build.

4. The build process starts.
   Once the build is complete, a success message is displayed. Hyperlink for location of generated library artifacts appears in the console.

5. Click the hyperlink to navigate to the path to view the build files.

Integrating the Native Library into a Native App - Android

To integrate native library into a native app, do the following:

1. Once you build the native library successfully for the application in Kony Visualizer, you must copy the generated aars of the application to the libs folder of app module of the Android native project. If the libs folder is not present in the app module of the native project, create the libs folder and paste the aar files in it.

   Location of aars in Kony Visualizer project:
   Mobile channel: <`workspace>/temp/<AppID>/build/luaandroid/dist/<AppID>_aars`
   Tablet Channel:
   `<workspace>/temp/<AppID>/build/luatabrcandroid/dist/<AppID>_aars`

   > ***Note:*** Hyperlink of the generated library artifacts location will appear in the build console of Kony Visualizer.

2. In the native app's main project build.gradle, add the following snippet under allprojects > repositories scope.

   ```
   flatDir {
       dirs 'libs'
   }
   ```

   Here is how the code looks before adding flatDir

   ```
   allprojects{
          repositories{
                  google()
                  jcenter()
          }
                                  }
   ```

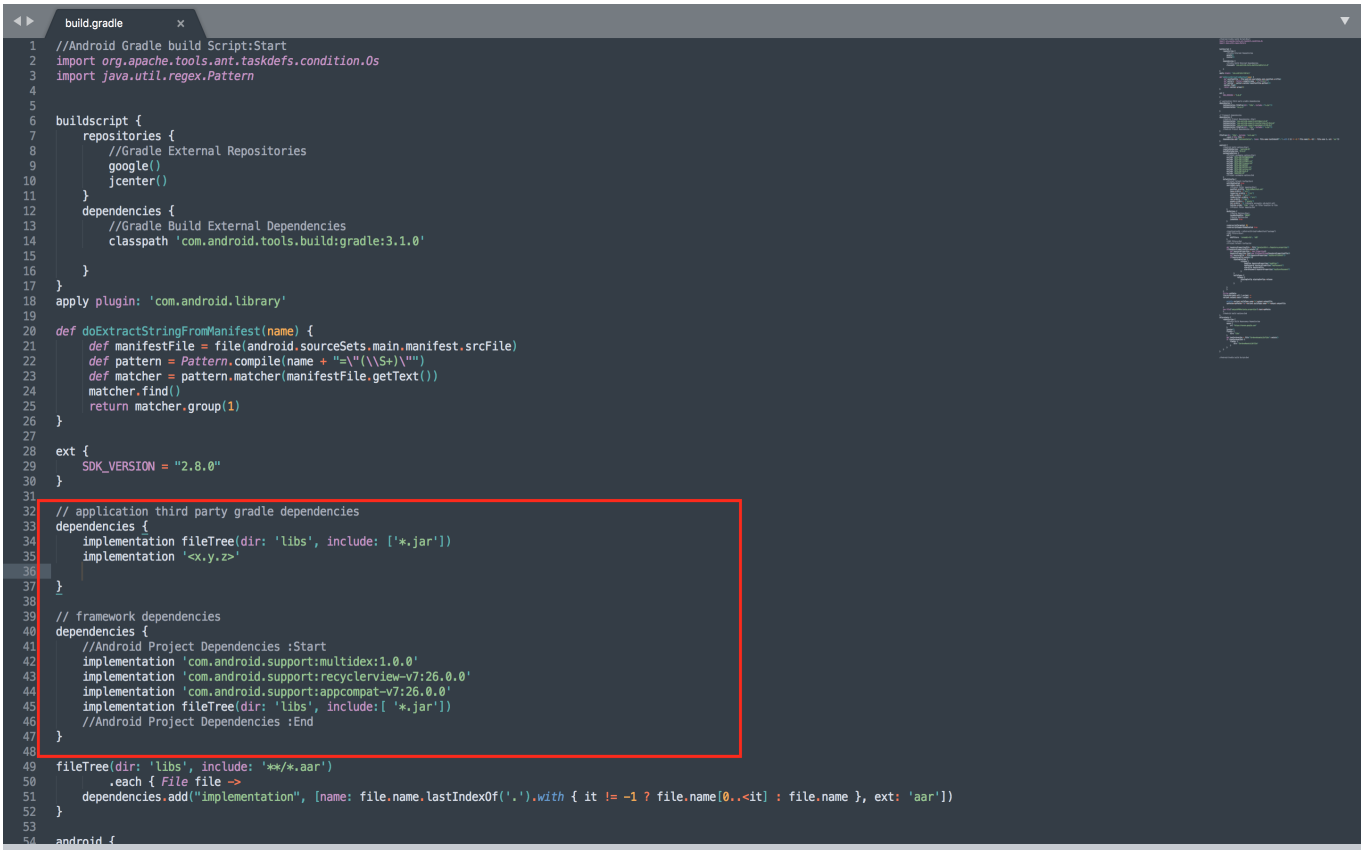
   Once the code is added, here is the view.

```
allprojects{
        repositories{
                google()
                jcenter()
                flatDir{
                        dirs 'libs'
                }
        }
}
```

3. In the build.gradle file of app module of the native project, to sync Kony application aars into the app module of native project, add the following code.

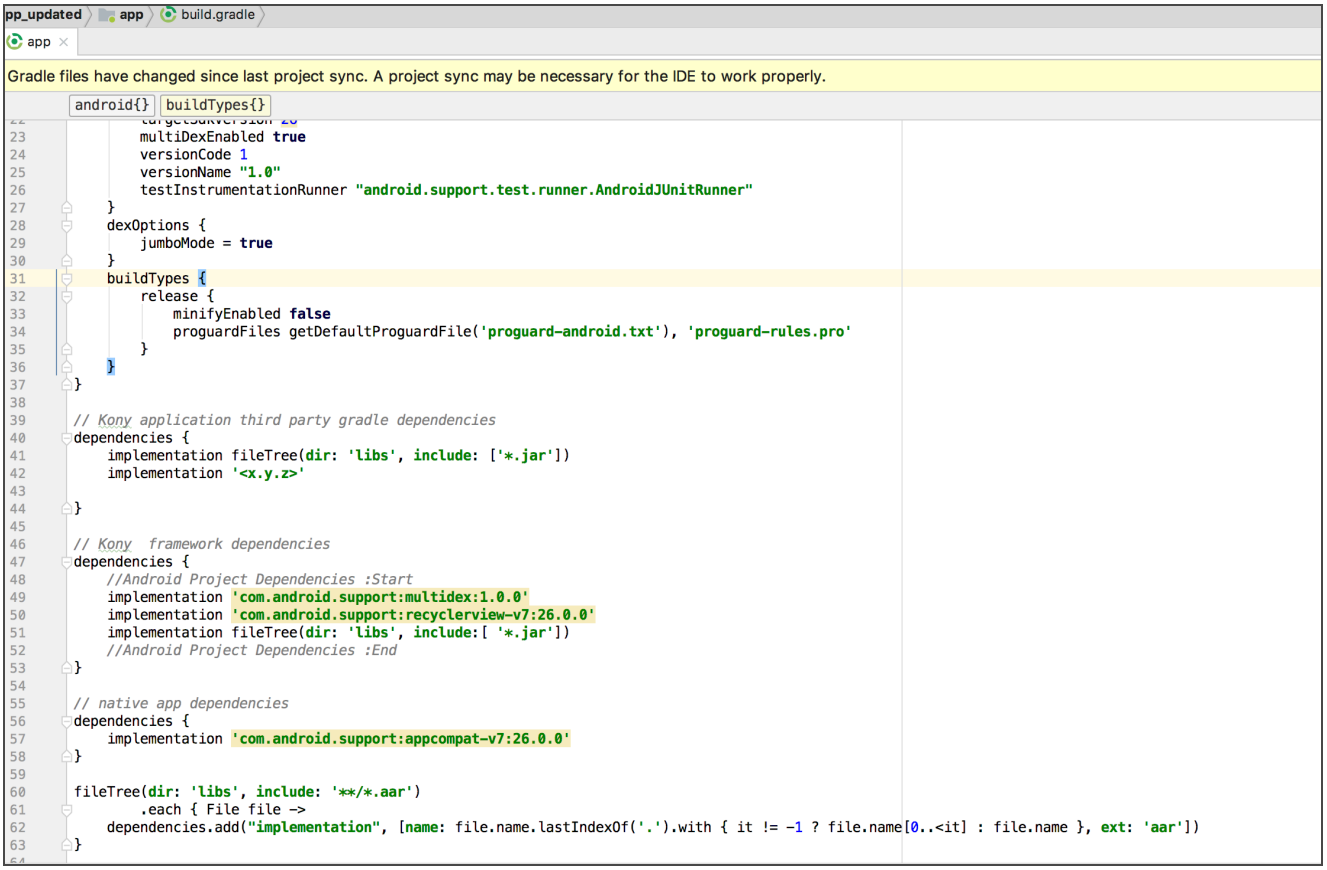
```
fileTree(dir: 'libs', include: '**/*.aar')
    .each {
        File file - & gt;
        dependencies.add("implementation", [name:
file.name.lastIndexOf('.').with {
                it != -1 ? file.name[0.. & lt; it] : file.name
            },
            ext: 'aar'
        ])
}
```

4. Add gradle dependencies of Kony library application (mentioned under dependencies scopes) of build.gradle file to the native project app module build.gradle file dependencies scope.
   Mobile channel: `<workspace>/temp/<AppID>/build/luaandroid/dist/<AppID>`
   Tablet channel : `<workspace>/temp/<AppID>/build/luatabrcandroid/dist/<AppID>`



> ***Note:*** Library application build.gradle file may contain multiple dependencies scope, so you must add all gradle dependencies under multiple dependencies scope of library application . If there are any external(remote) dependencies, then you should mention corresponding repository and url information in the native project main build.gradle file under allprojects repositories scope .

Once you incorporate the changes mentioned in the steps above, the gradle file looks like the image below.



5. Sync gradle changes of app module to add library artifacts.

Reduce Android App Binary Size

When you build a Kony app as Android library, native framework libraries (.so files)are bundled for supported 32 and 64-bit architectures in the library . For example, 32-bit architectures **armeabi-v7a**, **x86**, 64-bit architectures **arm64-v8a**, **x86_64**, and so on.

You can reduce the binary size of the native app by specifying the specific architecture of the library you want to support in your native app. You can do that by adding appropriate ndk filters to the app module build.gradle file of the native app.

```
android {
    defaultConfig {
```

```
        ndk {
            abiFilters "armeabi-v7a"
        }
    }
}
```

In this example, only **armeabi-v7a** architecture is bundled resulting in reduced binary size.

You can add more architectures you need by separating them with a comma. For example,

```
android {
    defaultConfig {
        ndk {
            abiFilters "armeabi-v7a, x86"
        }
    }
```

Integrating the Native Library into a Native App using integrateiOSLibrary Tool

You can integrate the native library in iOS using the integrateiOSLibrary tool.

1. Download the integrateiOSLibrary tool from here.
2. Open the tool and run the following arguments.
   `./integrateiOSLibrary -proj <path to xcodeproj> -lz <path to Generated iOS Native Library>`

   > ***Note:*** In case of xcworkspace, pass one of the projects in the workspace as an argument to the executable.

   Once you run the executable with arguments, the integrateiOSLibary tool adds all the frameworks packed in the generated iOS native library to App targets of the provided native XCode project.

> ***Note:*** All frameworks of Kony are dynamic frameworks. Hence, all the frameworks should get added to both Embed Frameworks and Link Binary with Libraries sections of App Targets of the provided native XCode project.

integrateiOSLibrary Executable Usage

The following table provides information on options that you can execute using the integrateiOSLibrary tool.

Options	Requirement	Details
--help	Optional	Executable Information about usage and options.
-g | --groupname	Optional	Group Name in Xcode Project to which frameworks need to be added. If a new group name is given, the tool will create that group in the Xcode project and adds all the frameworks to that group. If the group name is not provided, a group with generated library zip name will be created. `./integrateiOSLibrary -proj <path to xcodeproj> -lz <path to Generated iOS Native Library> -g <group name>`

Options	Requirement	Details
-lf | --libraryframeworks	Optional	If frameworks generated with wrappers around library APIs need to be added to any framework target, pass respective framework targets using --frameworkTargets and respective frameworks(containing wrappers) using `--libraryframeworks`. You can provide multiple Frameworks separated by a comma. `./integrateiOSLibrary -proj <path to xcodeproj> -lz <path to Generated iOS Native Library> -f <frameworktarget1,frameworktarget2> -lf <libraryframework1,libraryframework2>`
-f | --frameworkTargets	Optional	If library APIs must be used in any framework target, provide that framework target as an argument. Tool will add the required frameworks which has library APIs to the provided framework targets. Multiple Framework Targets can be provided separated by comma `./integrateiOSLibrary -proj <path to xcodeproj> -lz <path to Generated iOS Native Library> -f <frameworktarget1,frameworktarget2>`
--debug	Optional	Prints debug statements
-clean	Optional	Removes all the Kony Frameworks added to the project. Argument : Pass xcodeproj to which kony frameworks are added and group name in the xcodeproj to which frameworks are added using -g option. `./integrateiOSLibrary --clean <path to Xcode project> -g <group name>`

Options	Requirement	Details
-proj	Mandatory	Xcode Project

Example

```
. / integrateiOSLibrary - proj / Users / abcd / iOSNativeProject /
iOSNativeProject.xcodeproj - lz / Users / abcd / iOSNativeLibrary.zip
```

> ***Note:*** If any of Kony frameworks are added manually to the XCode project, make sure to delete the frameworks manually before running the executable.

Integrating the Native Library into a Native App Manually - iOS

Once you build the native library for the application in Kony Visualizer, you must copy the generated .zip files of the application to the iOS native project. The generated library artifacts are present in a .zip file with the AppID (<AppID>.zip).

The location of the file is

```
<Workspace>/temp/<AppName>/build/server/(iphonekbf (or) ipadkbf (or)
universalkbf)
```

> ***Note:*** Path of the generated library artifacts location will appear in the build console of Kony Visualizer.

When you unzip the .zip file, it contains the following files.

- Framework with name of <AppID>.framework - Contains Kony iOS platform frameworks
- PlatformDependencies.zip.
- LibraryDependencies.zip. - Contains all dynamic frameworks.

To add frameworks to the iOS native project, do the following:

1. Create a folder in the native project and copy all the frameworks in the .zip file that includes AppID.framework, frameworks in the Library Dependencies, and frameworks in Platform Dependencies.
2. Open the XCode Project and add the folder as a group to the project. Initially uncheck all the targets in the **Add to targets** section.
3. Navigate to **Project** > **App Target** >**General** and add all the dynamic Frameworks in the folder to the embedded binaries section.

> ***Note:*** All the frameworks will be available in both **Embedded Binaries** and **Link Binary with Libraries** sections in Build Phases of the App target.

Integrating the Native Library into a Native App - Windows

Once you build the native library for the application in Kony Visualizer, you must copy the generated library artifacts of the application to the Windows native project.

The generated library artifacts are available in the following locations:

- **Mobile channel**:
  `<workspace>/temp/<AppID>/build/windows10/Windows10Mobile/<AppID>`
- **Tablet Channel**:
  `<workspace>/temp/<AppID>/build/windows10/Windows10/<AppID>`

To link Kony App to the Windows native library, do the following:

1. Download the KonyLibraryLinker tool from here.
2. Open KonyLibraryLinker tool.
3. Configure Target Path to the library artifacts for specific channels.
   Mobile channel:
   `<workspace>/temp/<AppID>/build/windows10/Windows10Mobile/<AppID>`

Tablet Channel:
`<workspace>/temp/<AppID>/build/windows10/Windows10/<AppID>`

Limitations

Android:

- Kony Android library does not support the **supportsRtl** feature . Native apps integrating with Kony android library must use **tools:remove ="android:supportsRtl** line under the application tag of the app module in the **AndroidManifest.xml** file to disable the **supportsRtl** feature .
- SQLCipher and BundleOpenSSL options are not supported for x86_64 bit architecture .

iOS:

The following features are not supported for the iOS platform:

- Watch App Channel
- Extensions
- Universal Links
- Deeplink
- Native Support for App life cycle events through **ApplicationNativeCallbacks.m**
- You must manually add usage Descriptions such as **NSCameraUsageDescription** to the Native App to which Micro App is integrated. Even if Usage Descriptions are provided in the **infoplist_configuration.json** file, the descriptions are not respected.
- If any resources such as png (or) xib are accessed through the code written as a part of FFI, the resources should be accessed from Framework Bundle instead of Main Bundle.

  NSString *konyLibraryPath = [[[[NSBundle mainBundle] bundlePath] stringByAppendingPathComponent:@"Frameworks"] stringByAppendingPathComponent:@"AppID.framework”];

> ***Note:*** AppID refers to the AppID in **Project settings** > **Application** > **ID of Viz Project**.

NSBundle *konyLibraryBundle = [NSBundle bundleWithPath:libraryPath];

Bundle for accessing resources should be konyLibraryBundle instead of [NSBundle mainBundle]

Build Native Local on Kony Visualizer

Overview

Build Native Local, a feature introduced in Kony Visualizer V9, is used to build native app binaries in Kony Visualizer and store the binaries on a local machine.
For example, if you have an Android device connected to your system, you can generate the .apk file locally and run it on your device by using adb.

Through the *build* process, application components are collected and repeatedly compiled for testing purposes, to ensure a reliable final product. The build process creates new resources, updates the existing resources, or does both.

After you develop an application, you must build the application to do the following:

- Test the application for its performance and appearance on a device or on emulators.
- Install the application on devices.

> ***Note:*** The plugins for the native builds will be downloaded when you build the app for the first time. Hence, it may take longer to build the app for the first time. Any subsequent builds will be incremental and faster.

If you add an external asset or library to the Visualizer project after a full build, you must clean up the project using the **Clean Build** option. The **Clean Build** option erases files that were generated during the previous build and makes the project ready for the subsequent build. Once the project is cleaned, Visualizer will perform a full build the next time you trigger a build.

If you do not clean the project and perform a build, Visualizer will not take into consideration the assets and libraries added while performing the build.

Incremental Build

When an app is built for the first time, a full build is performed. The subsequent builds are incremental. In an incremental build, the previously built state of the project is used and only those resources that are changed since the last build are regenerated.

You do not have to perform an incremental build and a full build separately. Visualizer will automatically decide whether to perform a full build or an incremental build based on the changes made.

The following are the scenarios where the build is always a full build.

- If controllers that interact with the NFI are modified.
- If the Project Settings for Android Mobile/Tablet are modified.

Prerequisites

Following are the prerequisites to build a native app on a local machine within Kony Visualizer:

- Access to a Kony Fabric Environment. You must sign in to Kony Visualizer using the login credentials of either your Kony Fabric Cloud or on-premise environments. If you want to use the Kony Fabric on-premise environment, you must configure Kony Visualizer to connect to the Kony Fabric URL.
- Kony Visualizer V9 or later.
- For publishing to the Enterprise App Store, you must have Kony Fabric V8 SP4 or later.

- 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 Development method, Development Team ID and Keychain password. To do so, go to **Project Settings** > **Native** > **iPhone/iPad**. For more details on the iOS configurations, click here.

    > ***Note:*** If you do not configure the project settings, the Build process generates a .kar file.

  - If you choose to build an application for the **Android** platform, you must provide the paths of the Android Home and Java Home. To do so, go to **Edit** > **Preferences** > **Build**. Under the Android section, provide the location of the **Android Home** and **Java Home**.
  - 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 signing details, click here.
  - If you choose to build an application for the **Windows** platform, you must provide the windows application settings and set the capabilities. To do so, go to **Project Settings > Native > Windows**.

    > ***Note:*** You can generate a binary for a Windows app only on a Windows machine.

- 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.

Post Build Actions

The Build Native Local option in Kony Visualizer builds the application for the selected native platforms and performs the selected Post Build Action. You must choose the Post Build Action in the **Build Native Local** window, before the build process begins. There are three types of Post Build Actions:

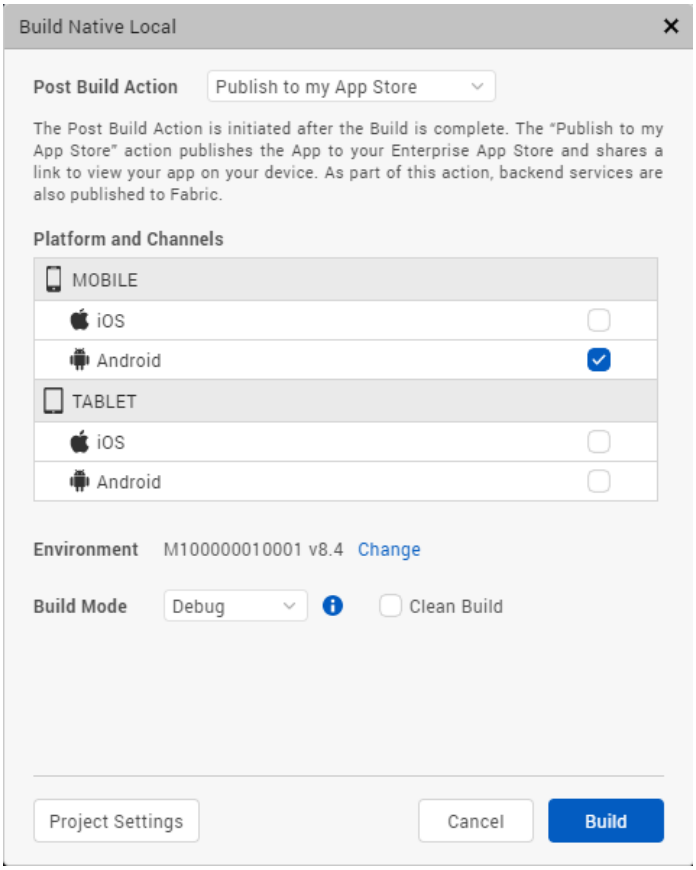
- Publish to my App Store – This action publishes the application to your Enterprise App Store
- Run on my Device – This action installs the application to your connected device and enables you to view your app on your device
- Generate Native App – This action generates the binaries and build logs for your Native application and saves it on your file system

The Post Build Action is initiated after the Build is complete.

Publish to my App Store

The Publish to my App Store action generates native app binaries and publishes the application to your Enterprise App Store. After a successful publish, a confirmation window appears, which shares a link to view the Enterprise app store on your device.

To publish an app to the Enterprise App Store, logging in to your Kony Account is mandatory.



> ***Note:*** You cannot build apps for the universal channel using this option.

For more information on Publish to my App Store, click here.

Run on my device

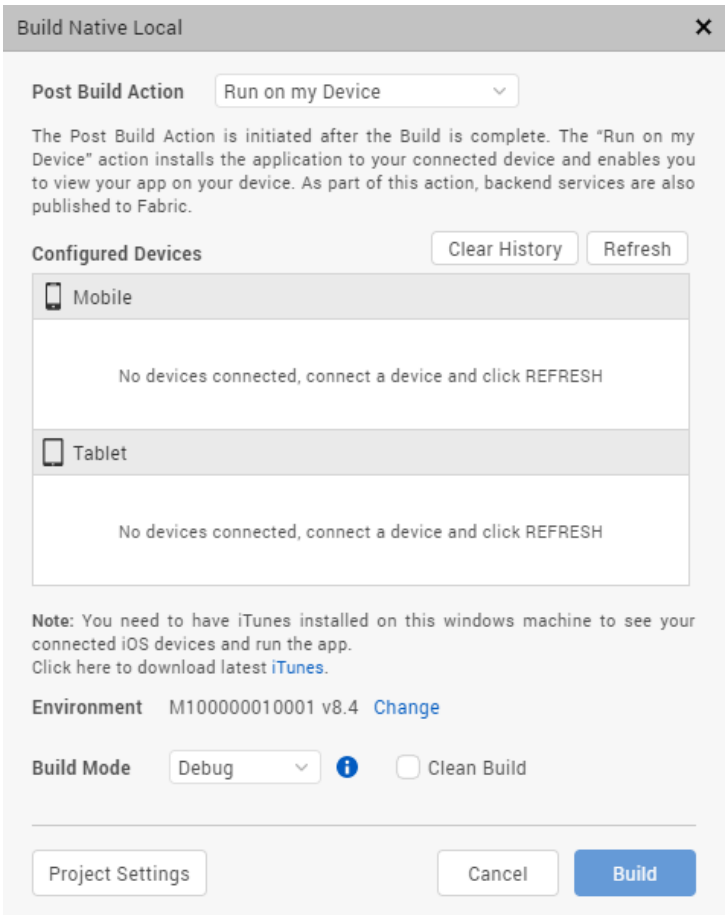
The Run on my Device action installs the application to your connected device and enables you to view your app on your device.

Establish a USB connection between the computer that built the app, and your device.

> ***Important:*** **USB Tethering for iOS devices on Windows Machine:**
> **Prerequisites** - Ensure that the latest version of iTunes is installed on the Windows machine. Before you start viewing the app on your iOS device by using the USB feature on Kony Quantum App, open iTunes on your Windows machine.

If you connect your device to the system after selecting the post build action, use the **Refresh** option to refresh the list of available devices that are connected to the system.

Use the **Clear History** option to clear out old entries of devices that are not connected to the system.



Once this action is completed, by default Android devices launch the app. Whereas, for iOS devices you need to explicitly launch the app by tapping on the app icon.

Generate Native App

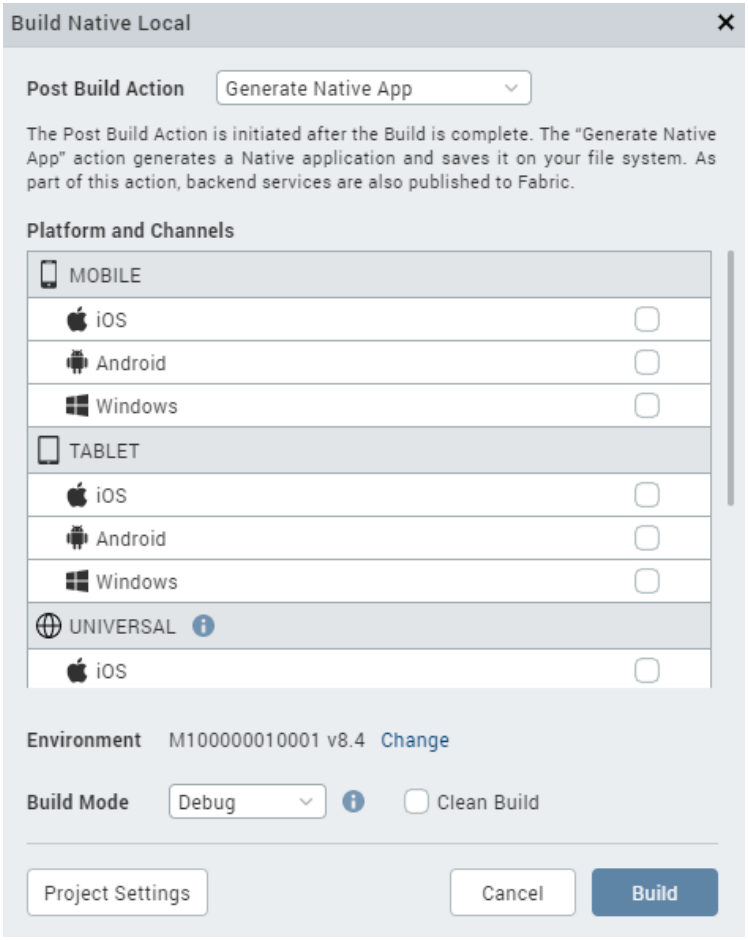
The Generate Native App action generates the binaries and build logs for your Native application and saves it on your file system. The Visualizer project does not have to be linked to Kony Fabric to complete this action.

Once the build is completed,

- If you have successfully built your Visualizer project for the Android channel, you will get Android mobile and/or tablet native APKs in your project's **Kony Visualizer workspace > binaries > local** folder.
- If you have successfully built your Visualizer project for the iOS channel, you will get iOS mobile and/or tablet native IPAs in your project's **Kony Visualizer workspace > binaries > local** folder.
- If you have successfully built your Visualizer project for the Windows channel, you will get windows mobile and/or tablet native APPXs in your project's **Kony Visualizer workspace > binaries > local** folder.

You can generate native apps even for the Universal channel by selecting a platform from the Universal section. This generates the APK, IPA or APPX for each of the platforms and channels selected.

To understand any build failures, you can go through the log file. To understand Run and Publish actions related to this type of Build, refer Post Successful Build.



Build a Native App Locally

To build an application, follow these steps:

1. On your Kony Visualizer, from the main menu, select **Build**.
2. From the context menu, select **Build Native Local**.
3. Select the platforms and channels for which you want to build the application.
4. From the **Post Build Action** drop-down menu, select the desired **Post Build Action**. For more details about the Post Build Action, click here.

5. You can choose to change the cloud environment on which your app is to be published. To do so, click **Change** beside the **Environment** option. By default, the Environment displayed is the one that was last selected.
6. From the **Build Mode** drop-down list, select your desired build mode.
7. Click **Build**. The build generation begins.

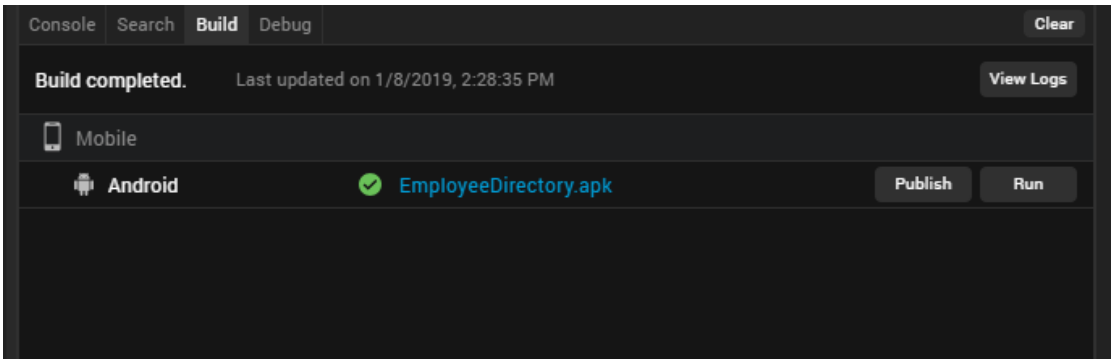
   You can check the status of your build in the Build tab. It undergoes various actions, like Project compression, uploading the compressed project to the cloud, and then the actual build begins. This process may take some time.

   If there are any errors, they appear in the Build tab.
8. From the Build tab, click **View logs** to open the build logs for the build service on your system.

> ***Note:*** The app generated using the Build Native Local option cannot be viewed on the Kony Quantum app.

Post Successful Build

Once the build is completed, the details of the generated binaries are provided in the Build tab.



- Click on the binaries to open the path of the binaries in your local machine.

- Click **Run** to run the app on your local device. If your post build action is already selected as Run on my Device, then the Run option is not enabled and there is no need to explicitly run the app again.
- Click **Publish** to publish the App on Enterprise App Store. If your post build action is already selected as Publish to my App Store, then the Publish option is not enabled and there is no need to explicitly publish the app again.
- Click **View Logs** to view the build related logs.

> ***Note:*** When an application is built for the Windows platform, the Publish and Run options are inactive.

Customize Quantum App

Overview

The **Customize Quantum App** feature is used to install the Kony Quantum App onto your device. The Kony Quantum App enables you to utilize and manage a set of capabilities related to Kony Visualizer and Kony Fabric services offered as part of Kony Quantum.

To preview an app that uses NFIs, you must provide certificates and signing keys used to generate the binary of the app. The NFIs are packaged with the app binary and built on the Kony cloud. You can use the generated app binary to install and run your app on a device.

If your app does not use NFIs, a QR code to download the Kony Quantum App appears on the canvas.

Kony Visualizer supports cloud builds of the application for the selected platforms and performs the selected Post Build Action. There are two types of Post Build Actions:

- Run on my Device - This action installs the application to your connected device and enables you to view your app on your device
- Generate Native App - This action generates the binaries and build logs for your Native application and saves it on your file system

Prerequisites

Following are the prerequisites to generate a customized Kony Quantum App within Kony Visualizer:

- 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. By default, new users get access to the Cloud build environment. Existing users need to request for access.
- Kony Visualizer V9 or later.
- 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 choose to build an application for the **Android** platform, then the Android signing details are mandatory. To do so, go to Project Settings > Native > Android Mobile/Tablet. For more details on Android signing details, click here.

Post Build Actions

The Post Build Action is initiated after the Build is complete. You must choose the Post Build Action in the Customize Quantum App window, before the build process begins. There are two types of Post Build Actions, they include:

1. Run on my device
2. Generate Native App

Run on my device

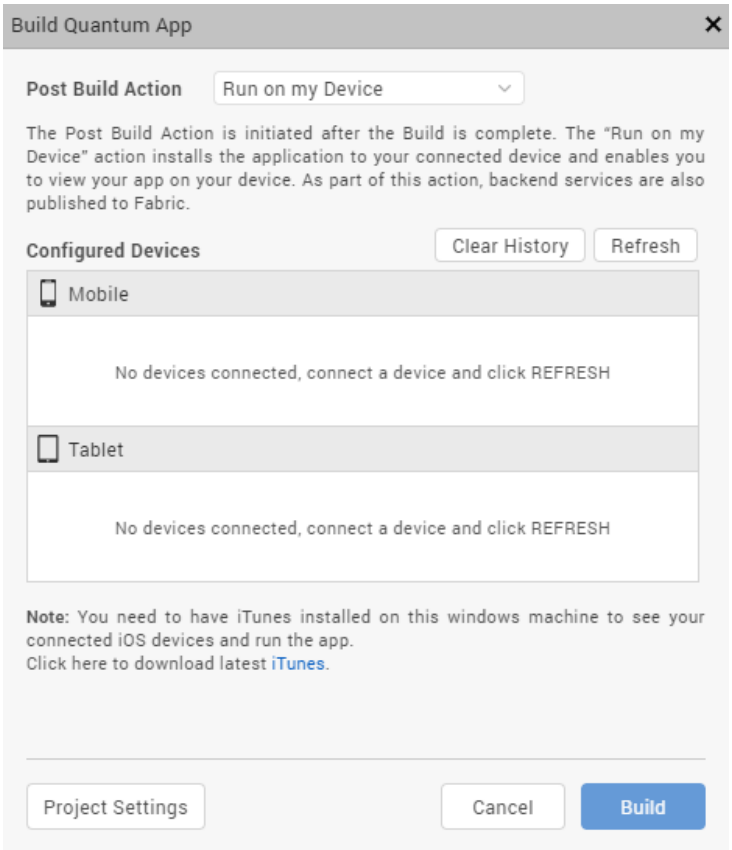
The Run on my Device action installs the application to your connected device and enables you to view your app on your device.

Establish a USB connection between the computer that built the app, and your device.

> *Important:* **USB Tethering for iOS devices on Windows Machine:**
> **Prerequisites** - Ensure the latest version of iTunes is installed on the Windows machine. Before you start viewing the app on your iOS device by using the USB feature on Kony Quantum App, open iTunes on your Windows machine.

If you connect your device to the system after selecting the post build action, use the **Refresh** option to refresh the list of available devices that are connected to the system.

Use the **Clear History** option to clear out old entries of devices that are not connected to the system.



Once the post build action is completed, by default Android devices launch the app. Whereas, for iOS devices you need to explicitly launch the app by tapping on the app icon.

Generate Native App

The Generate Native App action generates the binaries and build logs for your Native application and saves it on your file system. The Visualizer project does not have to be linked to Kony Fabric to complete this action.

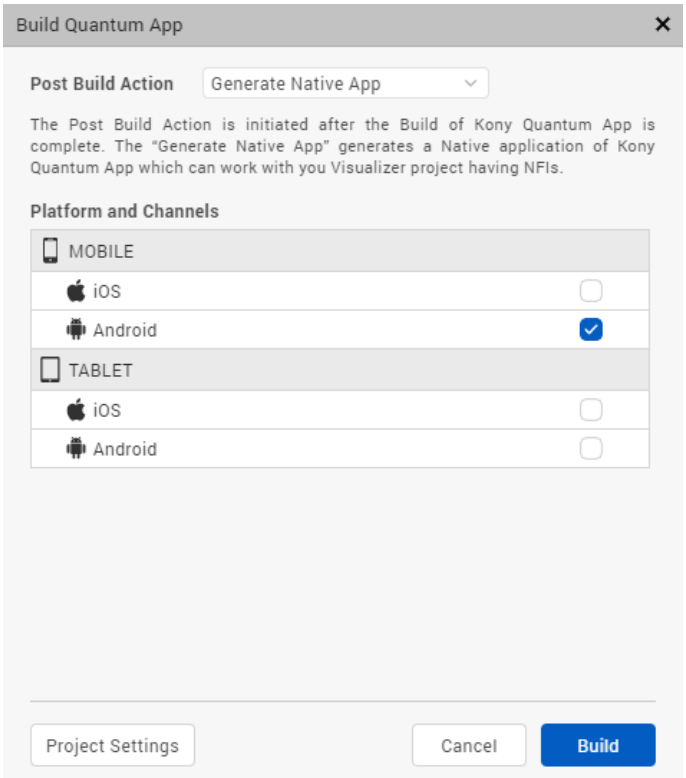
Once the build is completed,

- If you have successfully built your Visualizer project for the Android channel, you will get Android mobile and/or tablet native APKs in your project's Kony Visualizer workspace > binaries folder.
- If you have successfully built your Visualizer project for the iOS channel, you will get iOS mobile and/or tablet native IPAs in your project's Kony Visualizer workspace > binaries folder.
- You will also get the build logs in your project's Kony Visualizer workspace > binaries folder. You can refer to the logs to analyze the build for failures or success.

Alternatively, once the build is complete, you will get notified by an email from **Kony - Build Service**, with download links for all the generated binaries.

You can generate native apps even for the Universal channel by selecting a platform from the Universal section. This generates the APK or IPA for each of the platforms and channels selected.

To understand any build failures, you can go through the log file. To understand Run and Publish actions related to this type of Build, go to Post Successful Build.

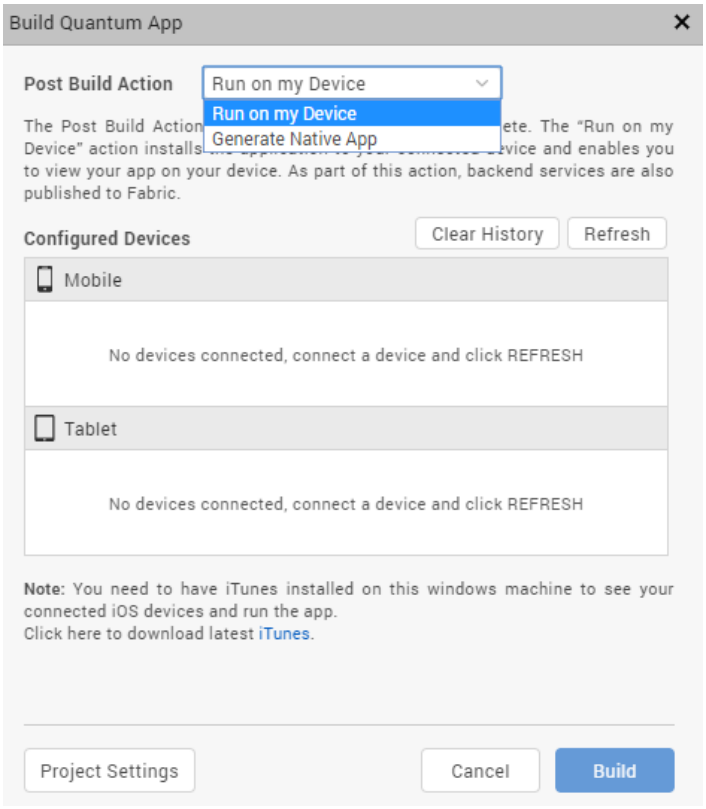


Customize Quantum App

To build an application, follow these steps:

1. On your Kony Visualizer, from the main menu, select **Build**.
2. From the context menu, select **Customize Quantum App**.
   The **Build Quantum App** dialog box appears.
3. Select the platforms and channels for which you want to build the application.
   Ensure you have unselected the check boxes for all the other platforms and channels.

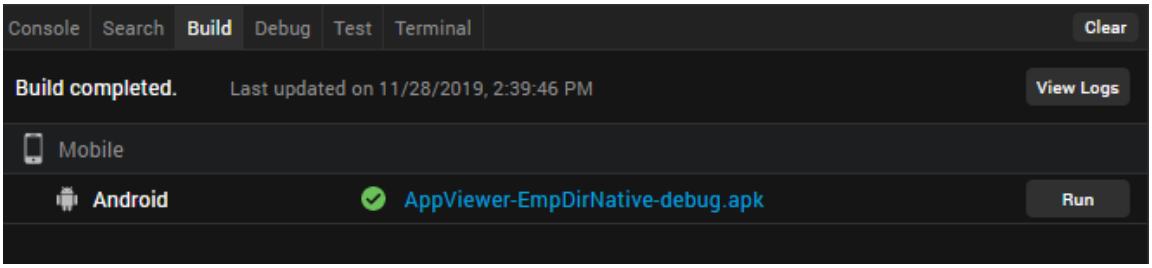
4. From the **Post Build Action** drop-down list, select the desired **Post Build Action**. For more details about the Post Build Action, click here.



5. Click **Build**.
   - If your app does not use NFIs, a QR code to download the Kony Quantum App appears on the canvas. Scan the QR code to download and install the Kony Quantum App to preview your app.
   - If you app uses NFIs, the build generation begins. You can check the status of your build in the Build tab. If there are any errors, they appear in the Build tab. Switch to the Console tab to view a detailed log of the errors.
     You can check the status of your build in the Build tab. It undergoes various actions, like Project compression, uploading the compressed project to the cloud, and then the actual build begins. This process may take some time. If there are any errors, they appear in the Build tab.
6. From the Build tab, click **View logs** to open the build logs for the build service on your system.

Post Successful Build

Once the build is completed, the generated binaries with the download link are provided in the Build tab.



1. Click on the binaries to open them in your default web browser.
2. Click **Run** to run the app on your local device. If your post build action is already selected as Run on my Device, then the Run option is not enabled and there is no need to explicitly run the app again.
3. Click **View Logs** to view the build related logs.

Alternatively, once the build is complete, you will get notified by an email from **Kony - Build Service**, with download links for all the successfully built app binaries. The mail contains details related to the Build. It contains the Project name, Build Action Triggered by, Date of Build, Build duration details. The Build Information section contains details about the Channels for which the App has been built.

- If you have successfully built your Visualizer project for the Android channel, you will get Android mobile and/or tablet native APKs link in the mail.
- If you have successfully built your Visualizer project for the iOS channel, you will get iOS mobile and/or tablet native IPAs link in the mail. The mail will also contain the OTA (plist) link, through which you can directly install the app on your device.
- You will also get the build logs. You can refer to the logs to analyze the build for failures or success.

> ***Note:*** The artifact links will be available only for 24 hours.

Limitations

- Apps that use Cordova plugins (or have Cordova enabled) are not supported.
- On the Android platform, apps that use push notifications are not supported.
- On the iOS platform, apps that use the universal links feature are not supported.

Integrate a React Native App to a Kony App

Applies to *Kony Visualizer Classic*.

Using this feature, you can integrate a React Native app (which is developed in React Native framework) to a Kony app (which is developed in Kony framework). The React Native app is embedded into the Kony form through a container, called as the React Native Container. Furthermore, some APIs are used to communicate between the React Native app's JavaScript framework and the Kony app's Kony framework. This feature is available from Kony Visualizer V8 SP4 onwards.

React Native is a framework developed by Facebook. React Native is used to create cross-platform mobile applications.

To gain a more hands-on experience on the React Native Integration process, download and use the TaskListSample app from GitHub.

For more information on the container and communication APIs that are essential to integrate a React Native app to a Kony app, refer these sections:

- React Native Container
- Communication APIs for React Native App

Prerequisites

Before you try to integrate the React Native app into your Kony project, ensure that you compile the reactNative project and run it at least once (by which you can confirm that the project is error-free).

Requirements for the React Native Project

For integrating the reactNative app into your Kony project, the requirements for the reactNative Project are as follows:

- You must create the reactNative project by using the **react-native init <AppName>** command.
- If multiple React Native apps are used in the Kony app, ensure that all React Native apps have the same dependency versions (for example, react-native": "0.57.0). Verify the **package.json** file of the React Native app to identify the dependencies.
- Ensure that the node dependencies of the reactNative project are installed. If they are not installed , install it by using the **npm install** command.

> ***Note:*** For iOS, the application code inside the React Native AppDelegate class is ignored. Multiple apps must not have the same class files or classes with the same name.

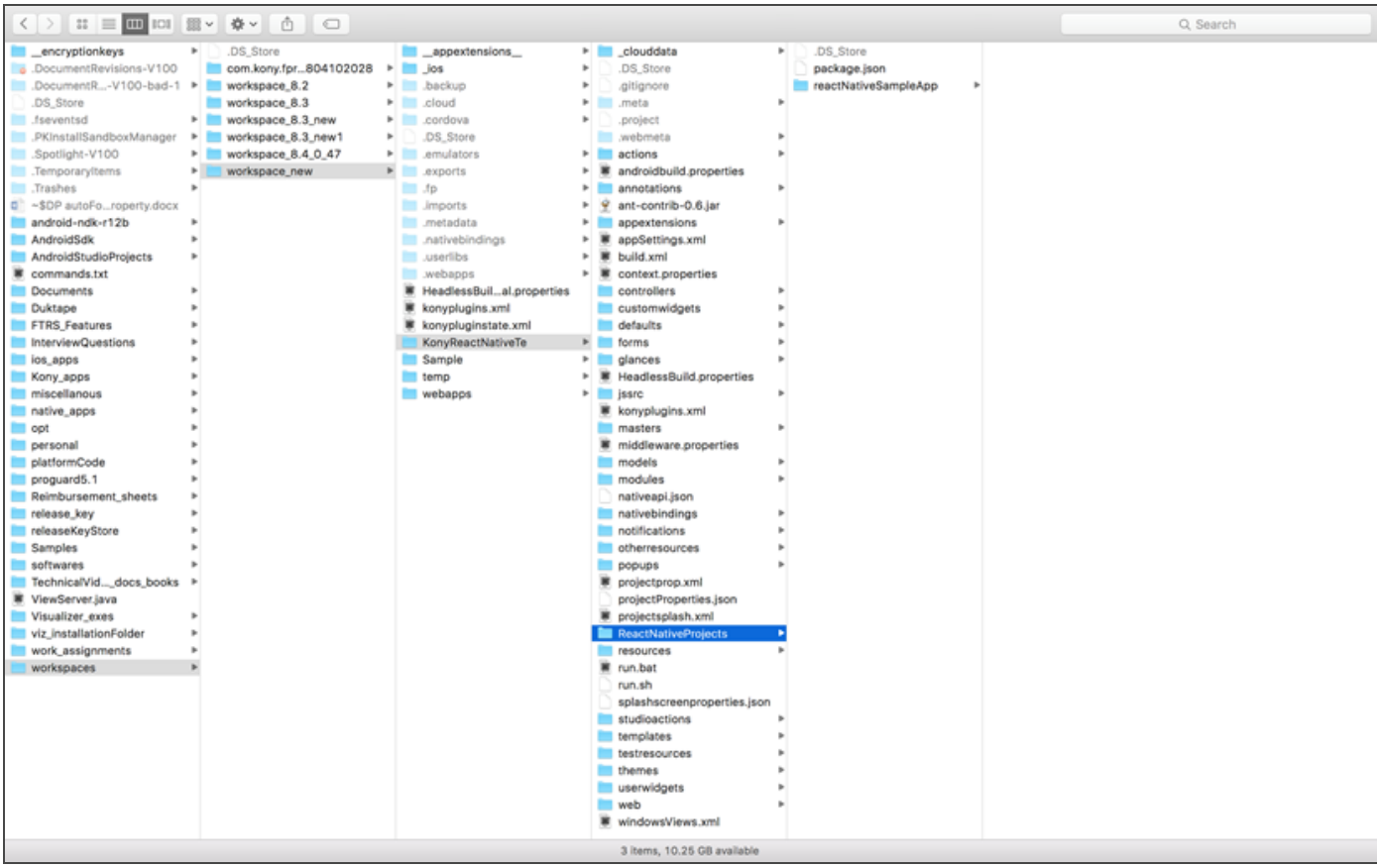
Manually Integrate a React Native App to a Kony App

In this section, you will learn how to integrate a React Native app to a Kony app manually.

> ***Important:*** Currently, Kony Visualizer does not provide the support to directly integrate a React Native app to a Kony app. As a result, you must perform the steps mentioned in the following sections to execute the integration process.

Common Steps

1. In the root folder of your Kony application workspace, create a folder and name it as **ReactNativeProjects**.



2. Copy the React Native app(s) to the **ReactNativeProjects** folder.
   For iOS, follow these steps:

   i. Create a package.json file in the **ReactNativeProjects**. This json file must contain a JSON object with a single key-value pair, where the key is **dependencies** and its value is an array of react-native dependencies mentioned in each package.json file of all the React Native apps. For example,

   //package.json file content

   {

   "dependencies": ["react","react-native","react-navigation"]

   }

ii. If the React Native apps contain pod dependencies, create a pod file in the ReactNativeProjects folder and mention all the pod dependencies for all the React Native apps, with the target as **KRelease** inside the pod file.

3. Run the following platform-specific React Native command from each ReactNative app root folder to generate the hierarchical JavaScript bundle:

   - For Android:
     react-native bundle --platform android --dev false --entry-file index.js --bundle-output android/app/src/main/assets/<AppName>/index.android.bundle --assets-dest android/app/src/main/res

     > ***Note:*** Create the assets folder (under <reactNativeApp>/android/app/src/main/) and also <AppName> folder under assets, if they do not exist.

   - For iOS:
     react-native bundle --entry-file index.js --platform ios --dev false --bundle-output ios/<AppName>/main.jsbundle --assets-dest ios

     > ***Note:*** Replace <AppName> with the app's **name** key value that is available in the app.json file, which is located at the root folder of the each React Native project.

Enable React Native for Andorid

You can enable the React Native feature for Android by adding its corresponding properties to the androidbuild.properties file. For detailed information on the manual steps of this process, click here.

Enable React Native for iOS

The manual steps to enable the React Native feature for iOS are as follows:

1. In Kony Visualizer, go to **Product > Build**, select the iOS Build option, and then click **Build**.
2. In the Console, a link will be generated, which is the path to the created KAR file.
3. Click the link. The KAR file location opens.

4. Rename the **konyappiphone.KAR** file to **konyappiphone.zip**
5. Unzip the konyappiphone.zip file, and then paste the ReactNativeProjects folder (which is present at root folder of app. workspace) in the unzipped folder.
6. Compress the contents of the unzipped folder to create an archive (zip).
7. Rename the zip with extension as KAR; this creates a KAR file that contains React Native apps.
8. Navigate back to Visualizer, go to **Product** > **Run As**, and then select the iOS emulator. The relevant logs in the Console are generated.
9. In the Console, a link to the path of **VMAppWithKonylib** is created.
10. Click the link to open the VMAppWithKonylib location.
11. Go to the **gen** folder inside the VMAppWithKonylib folder in the terminal.
12. Enter the **perl extract.pl modified_KAR_file location** command to perform the KAR file extraction.
13. After the Perl extract is complete, under the **Edit Scheme Build** options, deselect the **Parallelize Build** option for the release scheme.
14. For pods after the Perl extract, the **VMAppWithKonyLib.xcworkspace** file will be created; use this file to build the application.
    If this file is present, the you must add the **$(inherited)** flag under the **Other Linker** flag setting.
15. The third-party React Native libs that depend on pods should set the Framework Search Path, Library Search Path, and Header Search Path as **$(SRCROOT)/../../../../../Pods/** recursive.
16. If the pods require any specific build settings, such as enable modules, deployment target, and swift version, you must set them.
17. If the third-party React libraries have specific fonts, you must add those files to the project and also to the Info.plist file.

18. If the app depends on any other framework or library file other than pods, you must add them to the project.

19. If you are building in DEBUG mode, set BUILD ACTIVE ARCHITECTURES to YES.

Resolutions for Android Build Failure Issues

For Android, React Native apps are added as libraries to the Kony app, which might result in the failure of the build due to many reasons. Here are a few build failure issues when the React Native app is embedded and built for Android:

- minSDKVersion of the Kony app is lesser than one of the integrated React Native apps
  Resolution: Ensure that the Kony app's minSDKVersion value is equal to or greater than that of the integrated React Native app.
    - Build Failure due to conflict of Gradle dependencies versions (caused by using different versions of Gradle dependencies) in the Kony app and the React Native app and its libraries. For example, com.android.support:appcompat-v7:26.1.0.
      Resolutions:
        - Modify the build.gradle file of the React Native apps and its corresponding libraries to use the dependency versions same as that of Kony versions.
            i. Location of the build.gradle file in the React Native app:
            <reactNativeAppRootFolderPath>/android/app:build.gradle

> ***Note:*** The latest React Native apps use support libraries version variable as ext block of the build.gradle file, which is located at *<reactNativeAppRootFolderPath >/android*. You can modify the supportLibVersion from this location.
> The relevant example is as follows.

```
ext {.............
    supportLibVersion = "26.1.0"
```

```
}
```

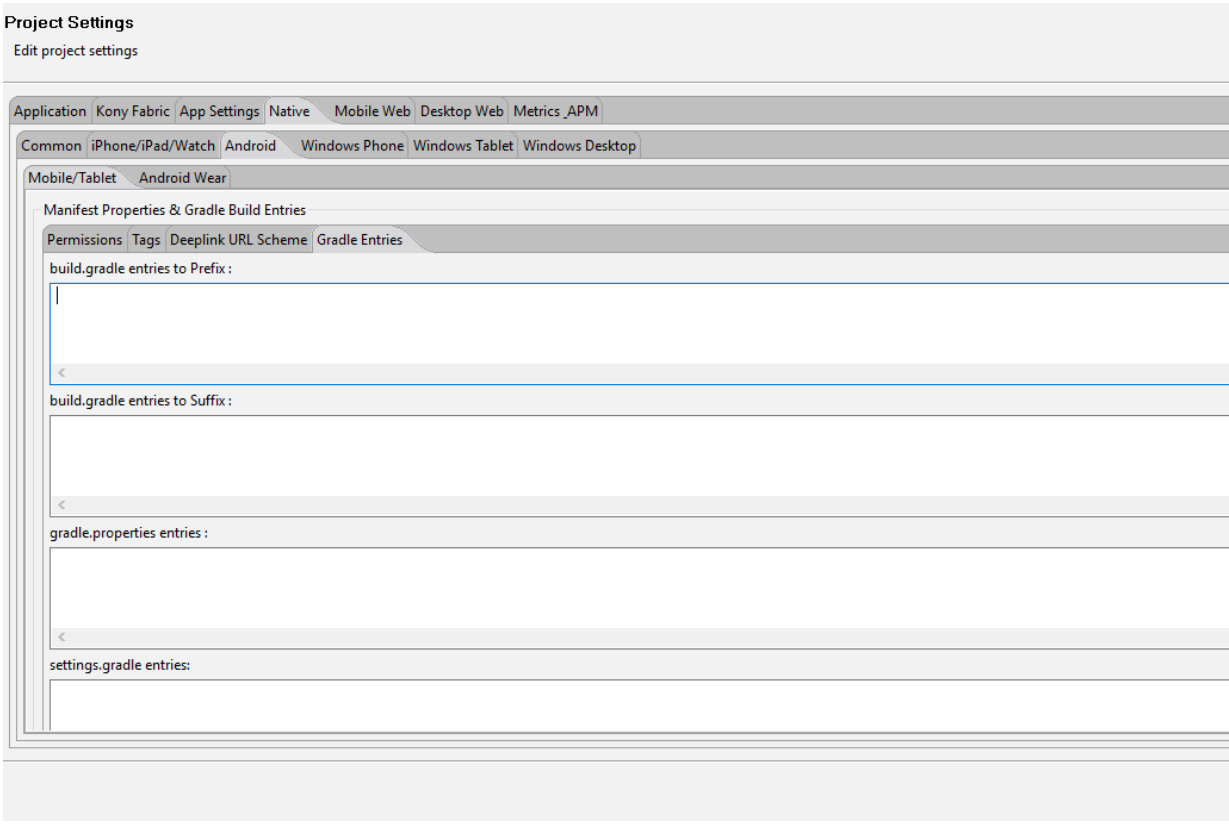
i. The build.gradle location of the Libraries that is embedded in the React Native app is as follows: *<reactNativeAppRootFolderPath>/nodemodules/<libraryName>/android/build.gradle*

> ***Note:*** You can view which libraries/modules are added to the React Native app by opening the settings.gradle file, which is located at *<reactnativeApp>/android*.

```
ext {.............
    supportLibVersion = "26.1.0"
}
```

```
ext {.............
    supportLibVersion = "26.1.0"
}
```

- Specify the Gradle system to use the given dependency version through the configuration strategy by adding the following code in the build.gradle file of the Kony app. To add the code in the build.gradle file, do the following:
  Go to **Project Settings > Native > Android**, scroll down to the **Manifest Properties & Gradle Build Entries** section, and then select the **Gradle Entries** tab.

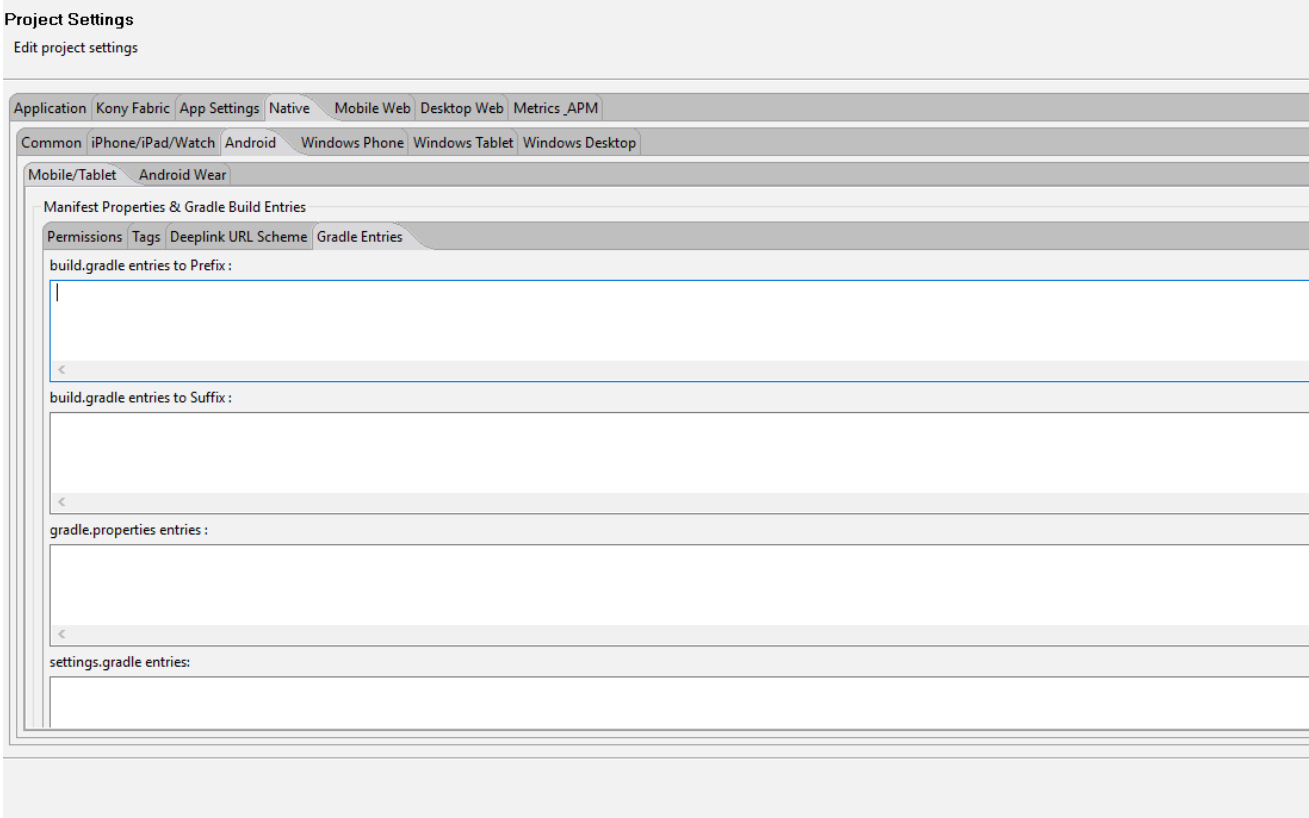



```
configurations.all {
    resolutionStrategy {
        force "com.android.support:recyclerview-
v7:27.1.1"
        force "com.android.support:appcompat-v7:27.1.1"
    }
}
```

Click here to resolve other types of Gradle-dependency issues, such as duplicate classes.

- Build failure due to not identifying the React Native app's Gradle dependencies.
  Resolution: Add all respositories of the React Native app mentioned in the build.gradle (<reactNativeApp>/android) file to the Kony app's build.gradle file. To add the repositories in the build.gradle file, do the following:
  Go to **Project Settings > Native > Android**, scroll down to the **Manifest Properties &**

Gradle Build Entries section, and then select the **Gradle Entries** tab.




```
allprojects {
    repositories {
        mavenLocal() google() jcenter() maven { // All of
React Native (JS, Obj-C sources, Android binaries) is
installed from npmurl "$rootDir/../node_modules/react-
native/android"}maven {url 'https://maven.google.com/'name
'Google'}maven { url "https://jitpack.io" }}}
```

- Build failure due to using different Build Tools versions in the Kony app and the React Native app.
  Resolution: Use the same Build Tools version in the React Native app as that of the Kony Build Tools version.

> ***Note:*** The Build Tools versions are mentioned in the app module build.gradle file.

```
android {
        ..........
    buildToolsVersion '27.0.3'
}
```

- Manifest merge conflicts

  Resolution: Click here to solve Manifest merge conflicts.

Kony's Gradle Plugin, Build Tools, and Gradle Dependencies

The relevant information on Kony's Gradle Plugin, Build Tools, and Gradle Dependencies for V8 SP4 is as follows.

> ***Note:*** The versions will change based on the plugins, so always refer the plugin corresponding to the build.gradle file for the exact Kony versions used .

Location of the build.gradle file:
<workspace>/temp/<appName>build/<luaandroid/luatabrcandroid>dist/<appName>:

Here is an example of a build.gradle file.

```
/** gradle android plugin version **/
classpath 'com.android.tools.build:gradle:3.2.1'
/** compileSDKVersion **/
28
/** build_tools_version **/
28.0.3
/** gradle version **/
gradle 4.6
/** support library dependencies **/
implementation 'com.android.support:recyclerview-v7:28.0.0'
implementation 'com.android.support:appcompat-v7:28.0.0'
/** Google play services dependencies versions **/
//maps
implementation 'com.google.android.gms:play-services-maps:11.6.0'
//locationlibversion
```

```
implementation 'com.google.android.gms:play-services-location:11.6.0'
//multidex_dependency
implementation 'com.android.support:multidex:1.0.0'
//dependencies_google_play_wearable_version
implementation 'com.google.android.gms:play-services-wearable:11.6.0'
//dependencies_android_support_wearable_version
implementation 'com.google.android.support:wearable:2.0.2'
//dependencies_android_wearable_wearable_version
compileOnly 'com.google.android.wearable:wearable:2.0.2'
//dependencies_google_fcm_messaging
implementation 'com.google.firebase:firebase-messaging:11.6.0'
//dependencies_google_play_pay_version
implementation 'com.google.android.gms:play-services-wallet:11.6.0'
//classpath_google_services
classpath 'com.google.gms:google-services:3.1.0'
```

Disable Print Statements in Builds

Using print statements is a common way of evaluating a build and assisting in its debugging. However, you may wish to disable the generation of print statements during a build, especially when generating a release build. Depending on your debugging methodologies, you may even want to disable print statements for debug builds as well.

By default, Kony Visualizer generates print statements for both Debug Mode and Release Mode builds.

To disable the generation of print statements during a build, do the following:

1. On the **Edit** menu, click **Preferences**.
2. From the left pane of the **Visualizer Preferences** dialog box, click **Build**.
   The Build options appear in the right pane.
3. Do one or both of the following, depending on whether you want to disable print statements for Debug Mode and Release Mode builds:

- **Release Mode builds.** To disable the generation of print statements during Release Mode builds, select the check box of the option titled **Stub the print statements in custom modules during release build**.
- **Debug Mode builds.** To disable the generation of print statements during Debug Mode builds, select the check box of the option titled **Stub the print statements in custom modules during debug build**.

4. Click **Done**.

Build an App Offline

Applies to *Kony Visualizer Classic*.

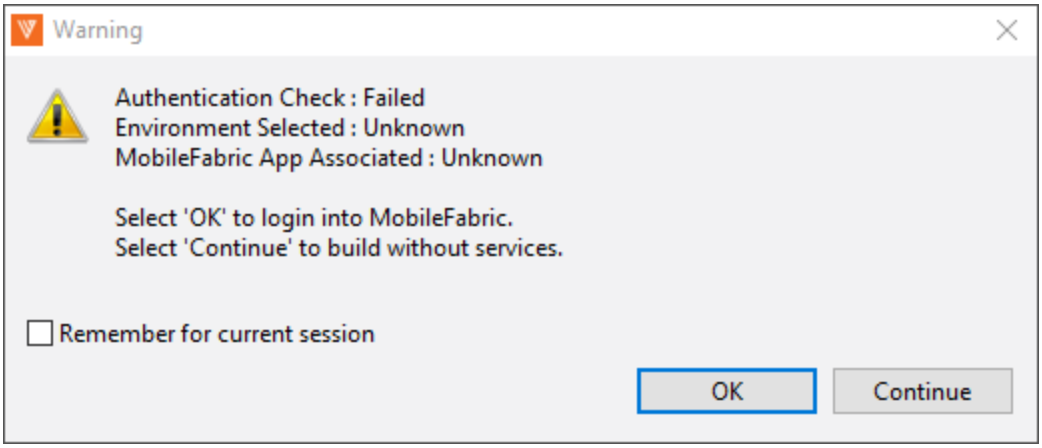
In a typical build, Kony Visualizer checks to ensure that the following three conditions are met:

- That you are logged in to Kony Fabric.
- That a valid environment is associated with the cloud account.
- For SPA, that a Kony Fabric application has been associated with the project.

These constitute an online build. However, it is possible to bypass these three conditions so that Kony Visualizer performs an offline build.

To perform an offline build, do the following:

1. If you are logged in to your Kony account, log out. To do so, near the right edge of the status bar, click the account name, and then click **Logout**.
2. On the **Product** menu, click **Build**.
3. In the Build Generation dialog box, select the channels and platforms for which you want to build your app. For example, you may want to build a native type of app for Mobile (phone) devices and Tablet devices for the iOS and Android platforms. For more information about native and SPA apps, see Types of Applications.
4. Select the build mode, either debug or release.
    - **Debug mode.** To help you identify and fix errors, Kony Visualizer emits the complete symbolic debug information . To lessen the amount of time necessary to complete the build, the build is not optimized for code execution, so it may tend to execute slower than a build optimized for release. Also, the inclusion of the symbolic debug information causes the final executable to be larger than a release build.
    - **Release mode.**Kony Visualizer optimizes the build for execution, requiring more time to generate the build. It also does not emit the complete symbolic debug information, making the final executable smaller than a debug build.
5. Click **Build**.
6. A Warning dialog box appears indicating that the three conditions necessary for the build to be online have not been met. It's in this dialog box that you indicate that you want to build offline.



In this dialog box, select among the following options:

- If you want Kony Visualizer to remember your selection for the rest of the current session (i.e. until you close Kony Visualizer), check the **Remember for current session** checkbox. This sets a property located in the Preferences dialog box. If you want to reset your decision for this session so that this Warning dialog box returns when you attempt to build without being logged in, on the **Window** menu, click **Preferences**. In the left pane of the Preferences dialog box, expand **Kony Visualizer**, click **Build**, and clear the check mark from the **Continue build even of no valid Kony Fabric configuration is found** checkbox. Click **Apply**, and then click **OK**.
- To perform an offline build, click **Continue**.
- To log in to Kony Fabric for a conventional, connected build, click **OK**.
- To cancel the build process all together, click the **X** in the upper right corner of the dialog box.

Perform a Headless Build

Applies to *Kony Visualizer Classic*.

A headless build is a method of building and publishing an app without launching the Eclipse environment that hosts Kony Visualizer. Instead, Ant[1] manages the build and publish process.

Conducting headless builds has a number of advantages.

- A developer does not have to open Kony Visualizer to build the application. This helps to integrate the compilation and building of Kony Visualizer projects with continuous build and integration tools like Cruise Control, Maven, and Hudson.
- A non-developer can handle release management.
- You can specify the binaries to be copied to a specific location of your choice, which reducing the manual effort of copying the files to a specific location.
- You can easily deploy an application to multiple servers.
- You can build the applications for multiple platforms with a single command.
- You can build applications with binaries protected mode enabled.
  You will need to configure few other settings for this to work. See Application Security for more information.

Running a headless build involves the following tasks:

1. Prerequisites for a Headless Build
2. Configure the HeadlessBuild.properties File
3. Configure the HeadlessBuild-Global.properties File

---

[1]An open-source software tool developed by the Apache Software Foundation that automates software build processes.

4. Increase Heap Memory
5. Build the Application

> ***Note:*** Before using conducting a headless build, ensure you have the files mentioned in Prerequisites

For headless builds, the following two files are required:

- HeadlessBuild.properties
- HeadlessBuild-Global.properties

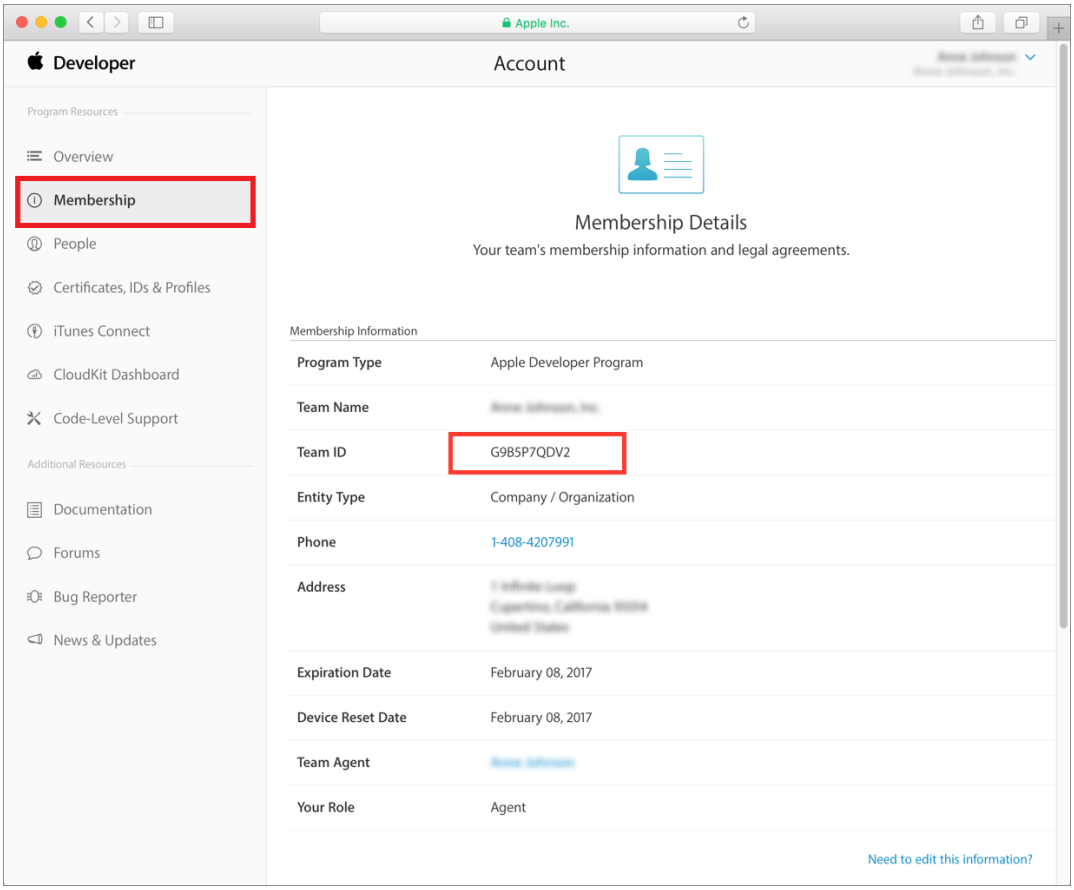
Most of the effort in preparing a headless build involves configuring these files. Ant gives you the flexibility to build an app or publish it. The decision of building or publishing is configured using Modes. The modes are:

- **Mode 0** - The application is built, and all the binaries are generated.
- **Mode 1** - The application is published on Kony Fabric.
- **Mode 2** - Through this combination of modes 0 and 1, the application is built and published on the Kony Fabric.
- **Mode 3** - This mode generates a web archive file that is combined with the Kony Fabric archive and also generates either a `.war` or `.ear` file, depending on what you specify. You must manually deploy the app on the server, or use a separate script.
- **Mode 4.** A combination of modes 0 and 3. The application is built and a web archive file that is combined with the Kony Fabric archive also is generated, along with either a `.war` or `.ear` file, depending on what you specify. You must manually deploy the app on the server, or use a separate script.

Prerequisites

Following are the prerequisites for doing headless build:

- Before doing headless builds, you have to do a blank publish to Kony Fabric to use services in your app. Building the application is not required in this case.
- Kony Visualizer is installed on your computer.
- Headless builds are supported only on Kony Studio 4.1 plug-ins and above.
- Ensure that there is no space in the file or folder names, and in the parents folder name.
- Applications imported form 6.5 version of Kony Visualizer needs to be built once in Kony Visualizer Classic before performing a headless build.
- If you are doing headless builds for iPhone or iPad, ensure that the Xcode in the Mac machine is available in the `/Applications` directory instead of the `/Developers` directory. To change the directory, use the command. `xcode-select`.
- For iOS, make sure to obtain a 10-digit development team identifier from Apple. The Team ID can be found in the Membership Details of your Apple Developer account.



Note: The Team ID must not include spaces. Otherwise, the IPA generation in Kony Visualizer fails.

Configure the HeadlessBuild.properties File

The following sections explain where to find the `HeadlessBuild.properties` file and how to configure it. Depending on what mode you are building (either 0, 1, 2, 3, or 4), you need to set particular values in the `HeadlessBuild.properties` file. These are detailed in "Configure the Mode" on page 1551.

1. How to Find the HeadlessBuild.properties File
2. Configure the Mode

How to Find the HeadlessBuild.properties File

The file `HeadlessBuild.properties` is stored in the project folder of your workspace. For example, if the name of your project is HelloWorld, and the name of your workspace folder is `Workspace`, the path in Microsoft Windows will be:

```
C:\Users\<username>\Workspace\HelloWorld
```

Example of HeadlessBuild.properties file

Following is an example of sections that are present in the `HeadlessBuild.properties` file and the values that can be used.

> ***Important:*** If the values for any variables are left blank, then the values set through Kony Visualizer will be retained while building the application.

```
#Note: Please escape '\' with '\\' in all file paths#
#This file represents the Application level properties used by
headless build.
project.name=app1

#mode 0-Build; 1-Publish; 2-Build & Publish; 3-Combine Web
Application+Kony Server Archive; 4-Build & Combine Web
Application+Kony Server Archive;
#mode-0: Application will be built for the selected build platforms
#mode-1: Based on publish.web, publish.service properties, app will
be published & services will be published
#mode-2: Application will be built for selected build platforms and
publish will be performed.
#mode-3: Combines web archive of app & kony server archive and
generates combined war/ear file
#mode-4: Application will be built & Generated web archive will be
combined with Kony server archive and final war/ear will be
generated
mode=0
```

```
#build mode [release | debug]
build.mode=release

#Application details
appid=app1
#Please ensure that this is incremented while doing multiple
Headless builds involving the modes 0 or 1 or 2
version=1.0.0
map_google_key=
default_locale=
#The android packagename can follow the pattern
com.<orgname>.<appid>
android.packagename=com.orgname.app1

#Cloud Mode credentials (Applicable only for cloud)
cloud.username=
cloud.password=

#Provide Kony Fabric specific details
konyfabric.url=

# Environment.Name used for publishing example using the format
LocalDevEnv
# For example, if your Kony Fabric Environment URL is:
# https://mycompany.konycloud.com
# then the value is as follows: environment.name=mycompany
environment.name=
account.id=
mf.appname=
mf.app.version=
#Specify the platforms for which the headless build needs to run.
#Mobile Channel.
```

```
iphone=false
android=false

spa.iphone=false
spa.android=false

#Tablet Channel.
ipad=false
androidtablet=false

#Selecting Windows10 will also trigger builds for X86,X64,ARM
architectures.
windows10=false

spa.ipad=false
spa.androidtablet=false
spa.windowstablet=false


#Desktop Channel.
desktop_kiosk=false
desktopweb=false

#Wearbles channel.
iphonewatch=false


#Universal Channel.
universal.iphone=false
universal.android=false

#App Extensions
iosappextension=false
```

```
#Provide iOS deployment target details in the case of iPhone or
iPad.

#Ex: 8.0, 9.0, 10.0
mac.iosdeploymenttarget=
#Ex: 2.0, 3.0, 4.0
mac.watchosdeploymenttarget=
#Ex: 4.0, 5.0
mac.swiftversion=

#Provide following details to generate IPA file in the case of
iPhone or iPad.
mac.ipaddress=
mac.username=
mac.password=

keychain.password=
#Possible values for method are "app-store", "ad-hoc", "enterprise",
"development"
method=
#Examples for development.team are "G9B5P7QDV2", "PM7453S8QE"
development.team=
#Possible values are true/false
genipaiphone=false
genipaipad=false

#middleware server properties
middleware_server_ip=192.168.251.1
middleware_http_port=80
middleware_https_port=443
#used in cloud mode
cloud.middleware.url=
```

```
middleware_web_context=middleware
mobileweb_web_context=app1

#To build binaries with protected mode enabled.
protectedmodeenabled.ios=true
protectedmodeenabled.android=true
#If not specified, by default final binaries will be copied to
'binaries' folder inside project
binaries.location=

#Combine Web Application+Kony Server Archive. Applicable for mode =
3 or 4 #
#Full path of middleware archive (war/ear).If project has Kony
session Manager,
#provide with-cache archive, for Http session Manager, provide
without-cache archive.
combinewar.middlewarearchive=
combinewar.context=
#combinewar.war,combinewar.ear are mutually exclusive(Only one of
them should be true)
combinewar.war=false
combinewar.ear=false
#Provide full path of dependant libraries,Separate with semicolon(;)
if there are multiple libraries
combinewar.dependencylibraries=

# Supported context paths for Kony Fabric components, if customized.
context.path.identity
context.path.workspace
context.path.accounts
context.path.console
```

```
#Siteminder login URL if your on-premise Kony Fabric is protected by
siteminder.
login.siteminder.url
```

Important: If you do not want to store your password in the headless build.properties file, you can use mfcli to encrypt your password. You can download the mfcli.jar from https://community.kony.com/downloads.

Ensure that you use the corresponding version of MFCLI as that of the Visualizer. i.e 7.x viz, 7.x mfcli, 8.x viz, 8.x mfcli.

To encrypt the password using mfcli (using default password.encryption.key),

```
java -jar mfcli.jar encrypt "Kony@1234"
Encrypted password is: en1801f1abee7b9e12426c062509e1b18epd
```

Configure the Mode

Depending on what mode you are building for (either 0, 1, 2, 3, or 4), you need to set values in the `HeadlessBuild.properties` file.

Mode 0 Properties

In Mode 0, the application is built, binaries are generated at the location specified in HeadlessBuild.properties, additionally if no location is specified they will also be copied to <drive>/<workspace>/<project>/binaries, apart from the temp folder. To configure `build.properties` for mode 0, in addition to setting `mode=0`, simply set the platform you are building for to *true*.

```
For example, iphone=true
```

Mode 1 Properties

In mode 1, the application or services are published, the application or services are published to Kony Fabric, depending on the Kony Fabric details in `HeadlessBuild.properties` file. The properties are in addition to setting `mode=1`.

> ***Note:*** In order to publish, initial build has to be done in mode 0 before you execute mode 1 and mode 2.

Mode 2 Properties

Because mode 2 is a combination of modes 0 and 1, you must set the properties described in the Mode 0 properties and Mode 1 properties, and set the property `mode=2` in order to build and publish to Kony Fabric. For example:

```
#Cloud Mode credentials (Applicable only for cloud)
cloud.username=username@kony.com
cloud.password=Abc@123

#Provide Kony Fabric specific details
konyfabric.url=https://mbaastest25.konylabs.net:443

# Environment.Name used for publishing example using the format
LocalDevEnv
# For example, if your Kony Fabric Environment URL is:
# https://mycompany.konycloud.com
# then the value is as follows: environment.name=mycompany
environment.name=
```

Mode 3 Properties

In mode 3, Kony Visualizer combines the web archive and the Kony Fabric archive which presumes the web archive is already built and generates a combined war/ear file. If the application is not built or the web archive is not available, then Kony Visualizer displays an error. To configure `HeadlessBuild.properties` for mode 3, in addition to setting `mode=3`, the following properties must be set:

```
#Full path of middleware archive (war/ear). If project has Kony
session Manager, provide with-cache archive, for Http session
Manager, provide without-cache

archive.combinewar.middlewarearchive= <Path to middleware archive
(war/ear, based on whether combinewar.ear / combinewar.war
properties)>
combinewar.context= <Provide context name with which final war/ear
will be generated.>

#combinewar.war and combinewar.ear are mutually exclusive(only one of
them should be true)
combinewar.war=false
combinewar.ear=false

#Provide full path of dependent libraries, separated by semicolons(;)
if there are multiple libraries
combinewar.dependencylibraries= <All MobileWeb dependency jars (full
path) separated by semicolons(;) should be provided.>
```

Mode 4 Properties

Because mode 4 is a combination of modes 0 and 3, you must set the properties described in the headings *Mode 0 Properties* and *Mode 3 Properties*, and set the property `mode=4`.

Configure Kony Fabric Parameters

To configure Kony Fabric parameters, follow these steps:

1. Open HeadlessBuild.properties file.
2. Provide user-name at *cloud.username*.
3. Provide password at *cloud.password*.
4. Provide Kony Fabric URL at *konyfabric.url*.

5. Provide Kony Fabric environment name at *environment.name*.

> ***Note:*** Providing user-name, password, and Kony Fabric URL are mandatory to build an application from the command line.

```
#Cloud Mode credentials (Applicable only for cloud)
cloud.username=username@kony.com
cloud.password=Abc@123

#Provide Kony Fabric specific details
konyfabric.url=https://mbaastest25.konylabs.net:443

# Environment.Name used for publishing example using the format
LocalDevEnv
# For example, if your Kony Fabric Environment URL is:
# https://mycompany.konycloud.com
# then the value is as follows: environment.name=mycompany
environment.name=
```

6. In order to enable services, you need to publish the app at least once from IDE.

Configure the HeadlessBuild-Global.properties File

The following sections explain where to find the `HeadlessBuild-Global.properties` file and how to configure it.

How to Find the build.properties File

The file `HeadlessBuild-Global.properties` is stored in the root folder of your workspace. For example, if the name of your workspace folder is Workspace, the path in Microsoft Windows would be something like this:

```
C:\Users\<username>\Workspace
```

Example of HeadlessBuild-Global.properties File

The following illustrates the sections that comprise the global.properties file and the values that can be used.

```
#The file represents the workspace level properties relevant for
Headless Build
#Note: Please escape '\' with '\\' in file paths (Ex:
C:\\workspace\\project)

 #Specify the eclipse workspace path where the project exists.
workspace.location=

#Full Path to a jar file whose name starts with
'org.eclipse.equinox.launcher_' in eclipse plugins folder
#Example: D:\\eclipse\\plugins\\org.eclipse.equinox.launcher_
1.1.0.v20100507.jar
eclipse.equinox.path=

#Android Home-Please specify the path of Android SDK. This is needed
only if Android Builds are being triggered
android.home=
```

Increase Heap Memory

If required, you can increase the heap memory available to your build by modifying the `run` file (either `run.bat` or `run.sh`). To increase the heap memory, type the following memory options in the `run` file.

```
java -Xms40m -Xmx512m -Dinput.file=%1 -Dglobal.file=%4 -cp %3
org.eclipse.core.launcher.Main -data %2 -application
com.pat.tool.keditor.konyapplication
```

Generating IPA

To generate IPA, following fields need to be configured before triggering the build.

1. Open HeadlessBuild.properties file.
2. Provide Mac User-name at *mac.username*.
3. Provide Mac password at *mac.password*.
4. Provide IP address at *mac.ipaddress*.

```
#Provide following details to generate IPA file in the case of
iPhone or iPad.
mac.ipaddress=xx.xx.xx.xx
mac.username=Username
mac.password=p@ssword
```

5. Provide profile key chain password, export method, and Team ID.

```
keychain.password=p@ssword
#Possible values for method are "app-store", "ad-hoc",
"enterprise", "development"
method="development"
#Examples for development.team are "G9B5P7QDV2", "PM7453S8QE"
development.team="G9B5P7QDV2"
```

6. Based on the requirement, user should set **genipaiphone** or **genipaipad** value to true.

```
#Possible values are true/false
genipaiphone=true
genipaipad=false
```

7. Value of the respective **channel** should be set to true.

```
#Mobile Channel.
iphone=True
android=false
windowsphone8=false
windowsphone81s=false
```

8. Build the application to generate IPA.

Build the Application

To build the application, follow these steps:

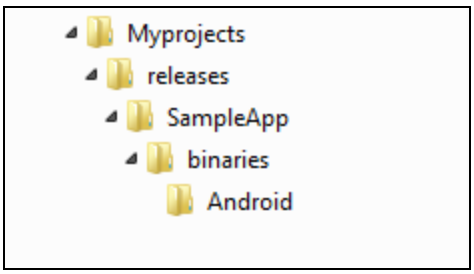
1. Navigate to your application's workspace.
2. Make sure the files *run.bat*, *build.xml*, *HeadlessBuild.properties*, and *HeadlessBuild-Global.properties* are present in the workspace.
3. Update the *HeadlessBuild-Global.properties* file with details like workspace location, *eclipse.equinox.path*, and preference. For example.

```
workspace.location = c:\\Myprojects\\Releases
eclipse.equinox.path = c:\\Eclipse_
6.5\\plugins\\org.eclipse.equinox.launcher_1.1.1.R36x_v20101122_
1400.jar
```

4. Update the *HeadlessBuild.properties* file with details like project.name, application, and platforms that you are building the application for.
5. Open the command prompt in the workspace.
6. Type `ant` in the command prompt. You can see the status of the build on the command prompt window.

> ***Note:*** If you want to write the console statements to a file for later viewing, such as for evaluating errors, use the command ***`ant > <drive:\filename>`***

7. If the app is built for Android, you can expect the folder to have the following structure (unless otherwise defined in the binaries section of the *HeadlessBuild.properties* file).



Continuous Integration

Introduction

CI Build allows the user to build and publish an app from the command line without any eclipse/installer dependency.

Prerequisite

Following are the prerequisites to install CI Build:

1. Ant (version >1.8.2) and Node (version > 7.10.0) should be installed in the system.
2. Copy the following plugins from `<<install_location>>\KonyVisualizerEnterprisex.x\Kony_Visualizer_Enterprise\plugins` to a new folder.
    - `com.kony.desktopweb_x.x`
    - `com.kony.ios_x.x`
    - `com.kony.mobile.fabric.client.sdk_x.x`

- com.kony.spa_x.x
- com.kony.studio.viz.core.win64_x.x(for Windows 64-bit users)
- com.kony.studio.viz.core.win32_x.x (for Windows 32-bit users)
- com.kony.studio.viz.core.mac64_x.x (for Mac 64-bit users)
- com.kony.studio.viz.core.mac32_x.x (for Mac 32-bit users)
- com.kony.thirdparty.jars_x.x
- com.kony.windows_x.x
- com.kony.windows8_x.x
- com.kony.windows10_x.x
- com.kony.windowsphone8_x.x
- com.pat.android_x.x
- com.pat.tabrcandroid_x.x
- com.pat.tool.keditor_x.x
- com.kony.cloudmiddleware_x.x
- com.kony.webcommons_x.x
- com.kony.cloudthirdparty_x.x

> ***Note:*** Windows platform is not supported in the version 8.0.

3. Copy `package.json` and `build.js` files from `download.kony.com/visualizer_enterprise/citools/<fix pack/service pack version>/visualizer-ci-tool-<fix pack/service pack version>.zip` in the project location. For example,

For Fix Pack 8 of V8 the URL is http://download.kony.com/visualizer_enterprise/citools/8.0.8/visualizer-ci-tool-8.0.8.zip

4. Open the command prompt in the project location and perform the `npm install`.

Configure `HeadlessBuild.Properties`

`HeadlessBuild.properties` file is present in the project location.

New Entries

Upgrade Kony Visualizer Classic project to 8.x or add the below new entries in `HeadlessBuild.properties` file:

`protectedmodeenabled.ios`	If the user wants to build for iOS in protected mode, change the value to true.
`protectedmodeenabled.android`	If the user wants to build for android in protected mode, change the value to true.
`plugin.dir`	Points to the directory, where the plugins required for the build are copied.
`javaloc`	Java home (provide the folder location consisting bin where the Java is installed.) Example: `<Install_location>\KonyVisualizerEnterprise8.0\Java\jdk`
`androidHome`	Android SDK path

For Proxy setup, the following new entries are applicable:
`proxy.host`
`proxy.port`
`proxy.username`
`proxy.password`
	If you are running CI build on a system behind a proxy, provide proxy details.

Existing Entries

Add the following existing entries in `HeadlessBuild.properties` file:

Key Name	Description
`project.name`	Project name
`mode`	Only modes 0, 1 and 2 are supported
`build.mode`	The mode of the build. Release or Debug.
`appid`	ID of the application

`#cloud mode credentials`
`cloud.username`
`cloud.password`
	Applicable only for cloud

`#mobilefabric specific details`
`mobilefabric.url`
`environment.name`
`accountd.id`
`mf.appname`
`mf.app.version`
	Applicable only when you are trying to publish the app.

Key Name	Description
#specify the environment you want to publish Example:qa cloud.environment	Applicable only when you are trying to publish the app.
#The platforms for which the headless build need to run **#Mobile Channel** iphone android spa.iphone spa.android **#Tablet** ipad androidtablet spa.ipad spa.androidtablet spa.windowstablet **# Desktop Channel** desktopweb **# Wearables** iphonewatch androidwearos	The value is either true or false. Enter true to build the platform else false. Please note that you cannot build Apple watch alone. When you build for Apple watch ensure that you build for iPhone as well.
#Provide the following details for IPA generation: mac.ipaddress mac.username mac.password keychain.password development.team.id method genipaiphone (to generate IPA for iPhone) genipaipad (to generate IPA for iPad)	Enter true to generate IPA for iPhone and iPad. For the method, possible values supported are **app-store**, **ad-hoc**, **enterprise**, and **development**.

Key Name	Description

Universal build for iOS:
`universal.iPhone`
`universal.android`
	Enter true to perform universal build for iPhone.
	
	Enter true to perform universal build for Android.

#Windows platform for headless build:
#Mobile Channel
`windowsphone10`
#Tablet
`windows10`
#Desktop Channel
`desktop_kiosk`
	Window platforms

`binaries.location`	Location, where the binaries are saved after the app is successfully built.

`version`
`default_locale`
`android.packagename`
`android.versioncode`
`ios.bundleversion`
	Support has been added for the following items from V8 SP1 onwards.

`keyAlias`
`keyPassword`
`keyStoreFilePath`
`keyStorePassword`
	Support has provided for the Android signing keys from V8 SP1 onwards for signing.

`context.path.identity`
`context.path.workspace`
`context.path.accounts`
`context.path.console`
	Supported context paths for Kony Fabric components, if customized.

Key Name	Description
`login.siteminder.url`	Siteminder login URL if your on-premise Kony Fabric is protected by siteminder.
`iosappextension`	App extension of the iOS.

Important: There are many keys available in the **HeadlessBuild.properties** file. However, not all of them are applicable for the CI Build. The keys mentioned above are the only ones applicable for CI Build.

Important: If you do not want to store your password in the headless build.properties file, you can use mfcli to encrypt your password. You can download the mfcli.jar from https://community.kony.com/downloads.

Ensure that you use the corresponding version of MFCLI as that of the Visualizer. i.e 7.x viz, 7.x mfcli, 8.x viz, 8.x mfcli.

To encrypt the password using mfcli (using default password.encryption.key),

```
java -jar mfcli.jar encrypt "Kony@1234"
Encrypted password is: en1801f1abee7b9e12426c062509e1b18epd
```

Platforms Supported

Following platforms are supported to build the CI application:

- iPhone, iPad
- Android mobile and tablet
- SPA mobile, tablet and desktop web

- SPA/DW publish
- Windows mobile, tablet and kiosk

Build the Application

Follow these steps to build the application:

1. Copy the `build.js` file to the project location.
2. Update the `HeadlessBuild.properties` file of the project, provide the necessary entries.
3. Open the command prompt in the project location.
4. Run the node command `node build.js`

   If there is a change in the plugin use `-clean` parameter, as below:

   ```
   node build.js -clean
   ```

5. If the app is built successfully, the binaries are saved in the location defined in the binaries section of `HeadlessBuild.properties` file.

   > ***Note:*** If the binary location is not specified in the file, the binaries are saved in the following default location.
   > `<projectLocation>/Binaries.`

Error Codes

The error codes are indicative of failed stage or operation. The actual error messages differ from the description mentioned below:

Example: Error code 50 describes as one of the mandatory field is missing. Actual error message will list the fields missing.

Error Code	Description
50	One or more mandatory fields are missing in HeadlessBuild.properties.
51	At least one platform should be selected for the build in HeadlessBuild.properties.
52	Plugin extraction failed (or) one or more mandatory plugins are missing.
53	Publishing Kony Fabric application failed.
54	Kony Fabric configuration details are missing in HeadlessBuild.properties. Example: error message will be (cloudname, cloudpassword, envname, accountID, mfAppName) is (or) are missing.
55	Project porting failed.
56	There are no forms created to build the selected channels. Example: There are no forms created for: Desktop
57	JAVA_HOME not found in environment variables (or) expected Java version is not found.
58	ANT_HOME not found in environment variables (or) expected ant version is not found.
59	Expected node version not found. Example: node version mismatch: required 7.10.0, found 6.10.2.
60	Expected Xcode version not found.
61	Expected Finalizer version not found.
62	Build failed for one or more selected platforms.

Continuous Integration for Kony Visualizer

Introduction

CI Build allows the user to build and publish an app from the command line without any eclipse/installer dependency.

Prerequisites

Following are the prerequisites to install CI Build:

1. Ant (version >1.8.2) and Node (version > 7.10.0) should be installed in the system.

   > ***Note:*** Windows platform is supported from the Visualizer V8 SP1

2. Download and extract the zip file from the location:`download.kony.com/visualizer_enterprise/citools/<fix pack/service pack version>/visualizer-ci-tool-<fix pack/service pack version>.zip` into the project location.
   For example, For V8 SP3 the URL is http://download.kony.com/visualizer_enterprise/citools/8.3.0/visualizer-ci-tool-8.3.0.zip

3. Copy plugindownload.js, `package.json` and `build.js` files from visualizer-ci-tool<fix pack/service pack version> folder.

4. Open the command prompt in the project location and perform the `npm install`.

5. Update plugin.dir and javaloc properties in HeadlessBuild.properties file.

CI Build for Kony Visualizer

Starting with Kony Visualizer V8 SP3 GA, CI build is supported in Kony Visualizer.

> ***Note:*** You must upgrade to V 8 SP3 to get CI build in applications created with a lower version of Kony Visualizer than SP3

The following properties have a default value. You can configure them if required.

Property Name	Property Key	Default Value	Property File Name
Accessibility config	isA11yConfigEnabled	false	projectprop.xml
Android wear minimum sdk version	andwearminsdkkey	7.1 (25)	projectprop.xml
Android wear target sdk version	andweartargetsdkkey	7.1(25)	projectprop.xml
Android wear maximum sdk version	andwearmaxsdkkey	None	projectprop.xml
Android (mobile/tablet) minimum sdk version	andminsdkkey	4.0(14)	projectprop.xml
Android (mobile/tablet) target sdk version	andtargetsdkkey	4.0(14)	projectprop.xml
Android (mobile/tablet) maximum sdk version	andmaxsdkkey	None	projectprop.xml
iOS Deployment target version	mac.iosdeploymenttarget		HeadlessBuild.properties

Property Name	Property Key	Default Value	Property File Name
Apple watch Deployment target version	mac.watchosdeploymenttarget		HeadlessBuild.properties
iOS swift version	mac.swiftversion		HeadlessBuild.properties
Splash screen related changes			splashscreenproperties.json

In case you want to modify any of the listed properties, you can find them in the following location: <workspace name>/<application name>.

Configure `HeadlessBuild.Properties`

`HeadlessBuild.properties` file is present in the project location.

New Entries

Upgrade Kony Visualizer project to 8.3.x or add the below new entries in `HeadlessBuild.properties` file:

`protectedmodeenabled.ios`	If the user wants to build for iOS in protected mode, change the value to true.
`protectedmodeenabled.android`	If the user wants to build for android in protected mode, change the value to true.
`plugin.dir`	Points to the directory, where the plugins required for the build are copied.

`javaloc`	Java home (provide the folder location consisting bin where the Java is installed.) Example: `<Install_ location>\KonyVisualizerEnterprise8.0\Java \jdk`
`androidHome`	Android SDK path

For Proxy setup, the following new entries are applicable:
`proxy.host`
`proxy.port`
`proxy.username`
`proxy.password`
	If you are running CI build on a system behind a proxy, provide proxy details.

Existing Entries

Add the following existing entries in `HeadlessBuild.properties` file:

Key Name	Description
`project.name`	Project name
`mode`	Only modes 0, 1 and 2 are supported
`build.mode`	The mode of the build. Release or Debug.
`appid`	ID of the application

`#cloud mode credentials`
`cloud.username`
`cloud.password`
	Applicable only for cloud

Key Name	Description
#mobilefabric specific details mobilefabric.url environment.name accountd.id mf.appname mf.app.version	Applicable only when you are trying to publish the app.
#specify the environment you want to publish Example:qa cloud.environment	Applicable only when you are trying to publish the app.
#The platforms for which the headless build need to run **#Mobile Channel** iphone android spa.iphone spa.android **#Tablet** ipad androidtablet spa.ipad spa.androidtablet spa.windowstablet **# Desktop Channel** desktopweb	The value is either true or false. Enter true to build the platform else false.

Key Name	Description

#Provide the following details for IPA generation:
`mac.ipaddress`
`mac.username`
`mac.password`
`keychain.password`
`development.team.id`
`method`
`genipaiphone` (to generate IPA for iPhone)
`genipaipad` (to generate IPA for iPad)
	Enter true to generate IPA for iPhone and iPad.
	
	For the method, possible values supported are **app-store**, **ad-hoc**, **enterprise**, and **development**.

Universal build for iOS:
`universal.iPhone`
`universal.android`
	Enter true to perform universal build for iPhone.
	
	Enter true to perform universal build for Android.

#Windows platform for headless build:
#Mobile Channel
`windowsphone10`
#Tablet
`windows10`
#Desktop Channel
`desktop_kiosk`
	Window platforms

`binaries.location`	Location, where the binaries are saved after the app is successfully built.

`version`
`default_locale`
`android.packagename`
`android.versioncode`
`ios.bundleversion`
	Support has been added for the following items from V8 SP1 onwards.

Key Name	Description

`keyAlias`
`keyPassword`
`keyStoreFilePath`
`keyStorePassword`
	Support has provided for the Android signing keys from V8 SP1 onwards for signing.

`context.path.identity`
`context.path.workspace`
`context.path.accounts`
`context.path.console`
	Supported context paths for Kony Fabric components, if customized.

`login.siteminder.url`	Siteminder login URL if your on-premise Kony Fabric is protected by siteminder.
`iosappextension`	App extension of the iOS.

Important: There are many keys available in the **HeadlessBuild.properties** file. However, not all of them are applicable for the CI Build. The keys mentioned above are the only ones applicable for CI Build.

Important: If you do not want to store your password in the headless build.properties file, you can use mfcli to encrypt your password. You can download the mfcli.jar from https://community.kony.com/downloads.

Ensure that you use the corresponding version of MFCLI as that of the Visualizer. i.e 7.x viz, 7.x mfcli, 8.x viz, 8.x mfcli.

To encrypt the password using mfcli (using default password.encryption.key),

```
java -jar mfcli.jar encrypt "Kony@1234"
Encrypted password is: en1801f1abee7b9e12426c062509e1b18epd
```

Build the Application for Kony Visualizer

Follow these steps to build the application:

1. Open the command prompt in the project location.
2. Run the node command `node build.js` in any of the following formats. These commands will download the required plugins as well as build the application.
    - node build.js --version <version number> // This command will download the specified plugin version to'plugin.dir' location and build the app.
    - node build.js // This command will build the app with existing plugins located in 'plugin.dir'.
    - node build.js -clean // This command will re-extract the plugins from plugin.dir folder

    > ***Note:*** For Delta download or when the 'plugin.dir' location already has a set of plugins then, run the following command
    > node build.js -c --version <version number>
    >
    > This command will download the upgraded plugins and build the project.

3. If the app is built successfully, the binaries are saved in the location defined in the binaries section of `HeadlessBuild.properties` file.

    > ***Note:*** If the binary location is not specified in the file, the binaries are saved in the following default location.
    > `<projectLocation>/Binaries`.

Error Codes

The error codes are indicative of failed stage or operation. The actual error messages differ from the description mentioned below:

Example: Error code 50 describes as one of the mandatory field is missing. Actual error message will list the fields missing.

Error Code	Description
50	One or more mandatory fields are missing in HeadlessBuild.properties.
51	At least one platform should be selected for the build in HeadlessBuild.properties.
52	Plugin extraction failed (or) one or more mandatory plugins are missing.
53	Publishing Kony Fabric application failed.
54	Kony Fabric configuration details are missing in HeadlessBuild.properties. Example: error message will be (cloudname, cloudpassword, envname, accountID, mfAppName) is (or) are missing.
55	Project porting failed.
56	There are no forms created to build the selected channels. Example: There are no forms created for: Desktop
57	JAVA_HOME not found in environment variables (or) expected Java version is not found.
58	ANT_HOME not found in environment variables (or) expected ant version is not found.
59	Expected node version not found. Example: node version mismatch: required 7.10.0, found 6.10.2.
60	Expected Xcode version not found.
61	Expected Finalizer version not found.
62	Build failed for one or more selected platforms.

Platforms Supported

Following platforms are supported to build the CI application:

- iPhone, iPad
- Android mobile and tablet
- SPA mobile, tablet and desktop web
- SPA/DW publish
- Windows mobile, tablet and kiosk

Publish a Kony Fabric App

Until an app is published, all its services and features are limited to just the Kony Visualizer development environment. To connect the app to live services, you have to publish the application to Kony Fabric. After you have signed in to Kony Fabric, you can select and publish your app to any of your cloud accounts. You can publish your app from the Kony Fabric Console by using the Publish tab.

Prerequisites

Before you can publish an app to Kony Fabric, you must meet the following prerequisites:

- Have a Kony Fabric account.
- Configure Kony Fabric in Kony Visualizer.
- Create a Kony Fabric app corresponding to the Kony Visualizer app that you are publishing.

For more information, refer Connecting to Services and Getting Started with Kony Fabric.

> ***Important:*** If your app contains deprecated widgets, it is possible that their skins may refer to the Helvetica font. If you are not explicitly using Helvetica in your app, you must verify your app's configuration and manually remove references to Helvetica before submitting it to the store.

This topic contains the following sections:

- Directly publish an app to Kony Fabric
- Publish the app from Kony Fabric

Directly Publish an App to Kony Fabric

From Kony Visualizer V8 SP4 FP 35 onwards, an option to directly publish your app to Kony Fabric has been provided. This feature is available on both Kony Visualizer and Kony Visualizer Classic.

To directly publish an app to Kony Fabric, follow these steps:

1. In Kony Visualizer, sign in to your Kony Cloud account. To do so, from the upper-right corner of the Kony Visualizer window, click **Login**. The Kony Account sign-in window opens. Type your Kony Cloud email and password credentials, and then click **Sign in**. Kony Visualizer uses the configured Kony Fabric URL to sign in to Kony Fabric.

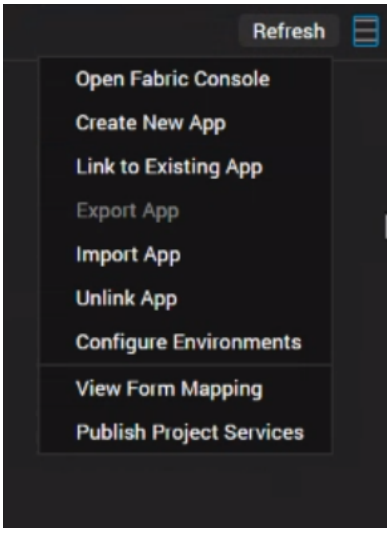
   > ***Note:*** In Kony Visualizer Classic, you can configure the Kony Fabric URL by going to: **Window > Preferences > Kony Visualizer > Kony Fabric**.

   > ***Important:*** If you are not able to get beyond the login page for the Kony Fabric Console, it could be because you set up Kony Fabricby using a **self-signed certificate**. The self-signed certificate allows you to install Kony Fabric, however, which Windows and Google Chrome do not trust the allow you to sign in. To resolve this issue, locate the certificate (you may need to contact your system administrator to do so), and then import it to the **Trusted Root Certification Authorities** folder of the Windows Certificate Store. For more information on how to import a certificate into the Windows Store, refer Import or export certificates and private keys on the Microsoft web site.

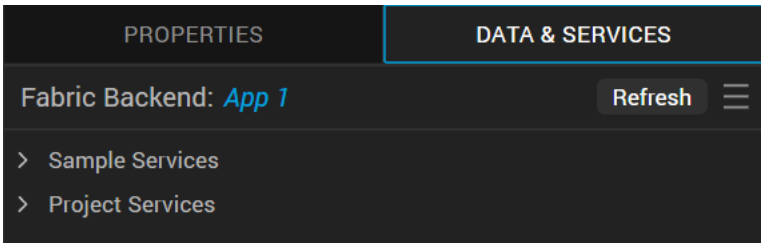
2. Select a default environment for Kony Fabric. To do so, on the **Project Explorer**, click **Project Settings**. Then, click the **Kony Fabric** tab. At the top of this tab, under Kony Fabric Environment, select an environment from the drop-down list. Click **Done**. If you do not see any environments listed, you need to create one. For more information, refer Environments in the Kony Fabric Console User Guide.

> ***Note:*** If an environment contains an alias hostname, the alias is used in the generated App Service Document. The alias is used to make service calls from the client app.

3. To publish to Kony Fabric, your Kony Visualizer client app must be associated with a Kony Fabric app, which means that you need to either create a new Kony Fabric app or use an existing one. To do so, at the upper-right corner of the **Data & Services** panel, click the hamburger menu icon hamburger menu icon  , and then click either **Create New App** or **Link to Existing App**. From the Kony Fabric Application dialog box, select the app to which you want to associate your Kony Visualizer app. For more information about how to create a new Kony Fabric app, refer How to Add Applications in the Kony Fabric Console User Guide.



Once you link your Kony Visualizer client app with a Kony Fabric app, the name of the linked Kony Fabric app is displayed beside **Fabric Backend** on the **Data & Services** panel. Here, *App 1* is the associated Kony Fabric app.



4. Right-click **Kony Fabric** for the linked app, and then click **Publish Project Services**. The app is successfully published to Kony Fabric.

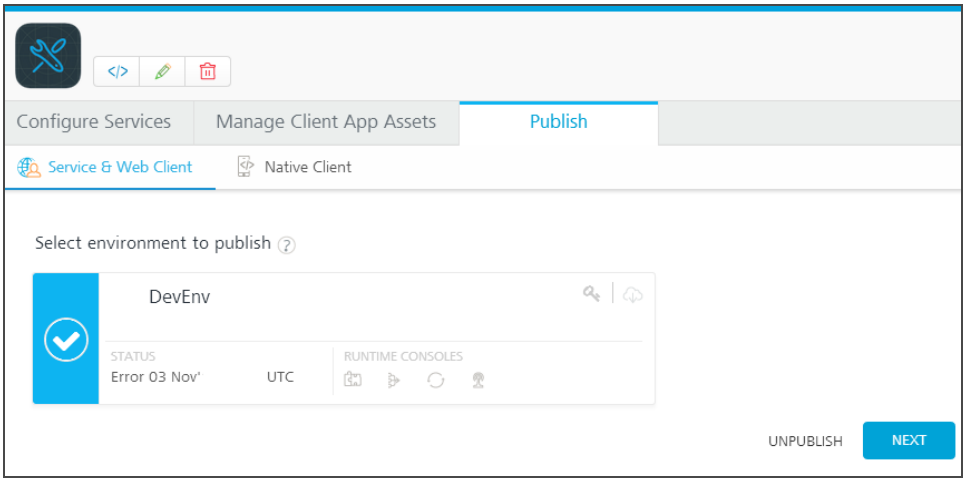
> ***Note:*** If you have not done so earlier, a dialog box appears asking you to select a valid Kony Fabric environment.

Publish the App from Kony Fabric

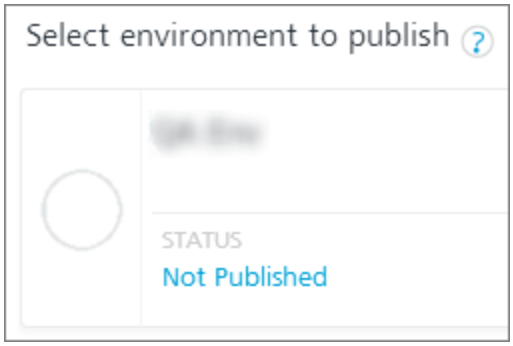
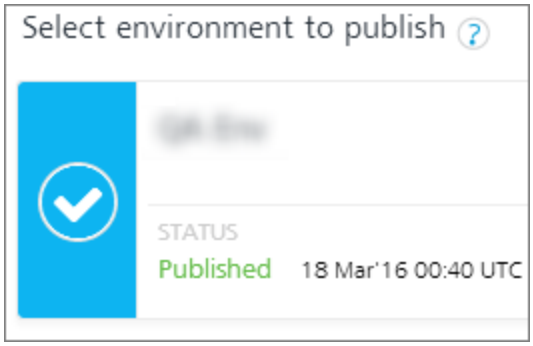
After you have published your app to Kony Fabric from Kony Visualizer, you can publish the app from Kony Fabric. This feature is applicable only in Kony Visualizer Classic.

To publish your app from Kony Fabric, follow these steps:

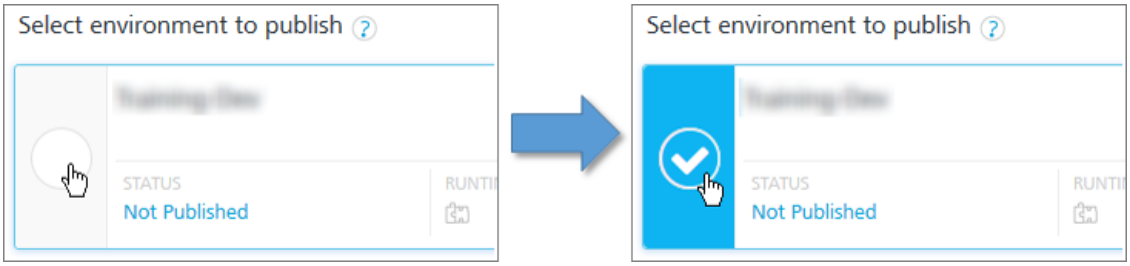
1. Once you have selected a Kony Fabric app to which you want to bind your Kony Visualizer app, the Kony Fabric Console opens. On the Kony Fabric console, click the **Publish** tab.



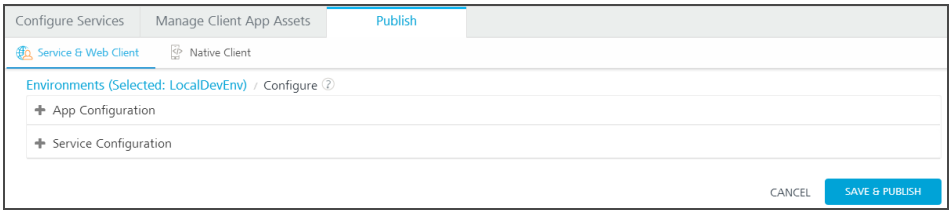
The status of the environment to which you want to publish the client app indicates whether the Kony Fabric app has been published.

Status when Not Published	Status when Published
	

2. If the Kony Fabric app to which you have bound the client app has not yet been published, select the app for publish.



3. Click **Next**. The **Environments** page appears.



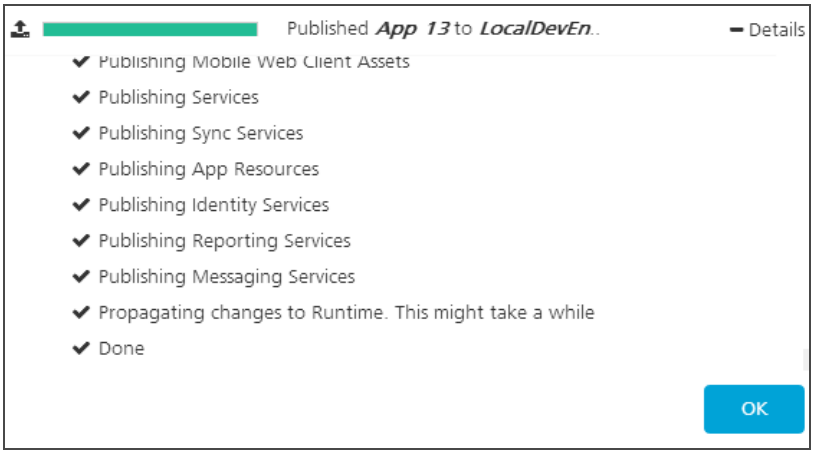
4. The **Configure** page displays options to change the configuration of your Services. To make any changes, click **+ Service Configuration**., and then click any of the blue-text fields.

> ***Note:*** If your app is an upload of web app (such as SPA, Desktop Web/Responsive Web, or Progressive Web App) binary , the Configure page displays a drop-down list of options specific to the web client environment.

5. If available, you can reconfigure the application-level settings and service configuration settings from this page .

6. Click **Save and Publish**. The app and any associated services are published to the selected environment.

> ***Note:*** If you are building a Kony Visualizer web app (such as SPA, Desktop Web/Responsive Web, or Progressive Web App), you must first define and publish any required Kony Fabric services. After you build any web application and upload its binaries to Kony Fabric, you cannot publish a service to it.
> If you encounter an error while publishing the Kony Fabric, try publishing again.



7. To close the Kony Fabric Console and return to Kony Visualizer, from the Quick Launch Bar along the upper left edge of Kony Visualizer, click the Visualizer icon  . Since you are still logged in to your Kony account, Kony Visualizer continues to have access to your Kony Fabric apps and services.

8. Launch the app. To do so, on the **Product** menu, navigate to **Run As**, and then select an emulator to run the app on.

Environment List in Visualizer

Prior to Visualizer V9 release, it lists only the environments to which you have full access. From Visualizer V9 onwards, you can view the list of all environments associated with your Fabric account. This list includes the environment with both full and read-only access.

The different types of access that a cloud account admin can provide, are:

- **Full Access**: When your cloud account has **Full Access**, you can build your application and publish your services to Kony Fabric. The environment with full access is listed in Visualizer.
- **No Access**: When your cloud account has **No Access**, you cannot build and publish your application to Kony Fabric. The environment with no access is not listed in Visualizer.
- **Custom**: When your cloud account has **Custom** access, your cloud account is provided with permissions depending on the features/components. These features/components include Server, Engagement, etc. You must have one of the following feature level permissions to build your application and see these environments listed in Visualizer:
    - Build Client App
    - Server

The following table shows the actions you can perform for each access.

User Access Type	Environment Access Type	View Environments in Visualizer	Build native Application	Build Web Application	Publish the Services
Full Access	Full Access	Yes	Yes	Yes	Yes
No Access	No Access	No	No	No	No

User Access Type	Environment Access Type	View Environments in Visualizer	Build native Application	Build Web Application	Publish the Services
Custom	Build Client App	Yes	Yes	No	No
Custom	Server	Yes	Yes	Yes	Yes

For more information about the different environment permissions of your cloud account, click here.

Build and Publish in Kony Visualizer

Overview

Build and Publish, a feature introduced in Kony Visualizer V8 SP4, is used to build and publish apps in Kony Visualizer . So far, Kony Visualizer could be used to preview your apps, now you have the feature to build the app and generate native app binaries.

Kony Visualizer supports the following types of cloud builds:

- **Build and Publish Native** - Builds the application for the selected native platforms and performs the selected Post Build Action. There are three types of Post Build Actions:
  - Publish to my App Store - This action publishes the application to your Enterprise App Store (EAS).
  - Run on my Device - This action installs the application to your connected device and enables you to view your app on your device.
  - Generate Native App - This action generates the binaries and build logs for your Native application and saves it on your file system.
- **Build and Publish Web** - Builds the web application for selected web platforms and publishes to Kony Fabric. For more details refer to Build and Publish Web Apps on Kony Visualizer.

> ***Note:*** You can configure Push Notifications for applications built using cloud build. For information on how to do so, click here.

Build and Publish Native

With the Build and Publish Native feature the dependency on Xcode for building iOS applications and the dependency on Android SDK for building Android applications has been removed.

The *build* process is a process through which application components are collected and repeatedly compiled for testing purposes, to ensure a reliable final product. Build process creates new resources, updates the existing resources, or does both.

After you develop an application, you must build the application to do the following:

- Test the application for its performance and appearance on a device or on emulators.
- Install the application on your devices.

The *publish* process makes the built application accessible from Kony Enterprise App Store. When you build and publish a native application, it is hosted on the Enterprise App Store associated to your Cloud account and can be accessed from the server.

To understand the build and publish process for native applications in Kony Visualizer, go to Build and Publish a Native App.

Prerequisites

Following are the prerequisites to build and publish a native app within Kony Visualizer:

- 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. By default, new users get access to the Cloud build environment. Existing users need to request for access.
- Kony Visualizer V8 SP4 or later.
- For publishing to the Enterprise App Store you must have Kony Fabric V8 SP4 or later.

- 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 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 signing 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.

Post Build Actions

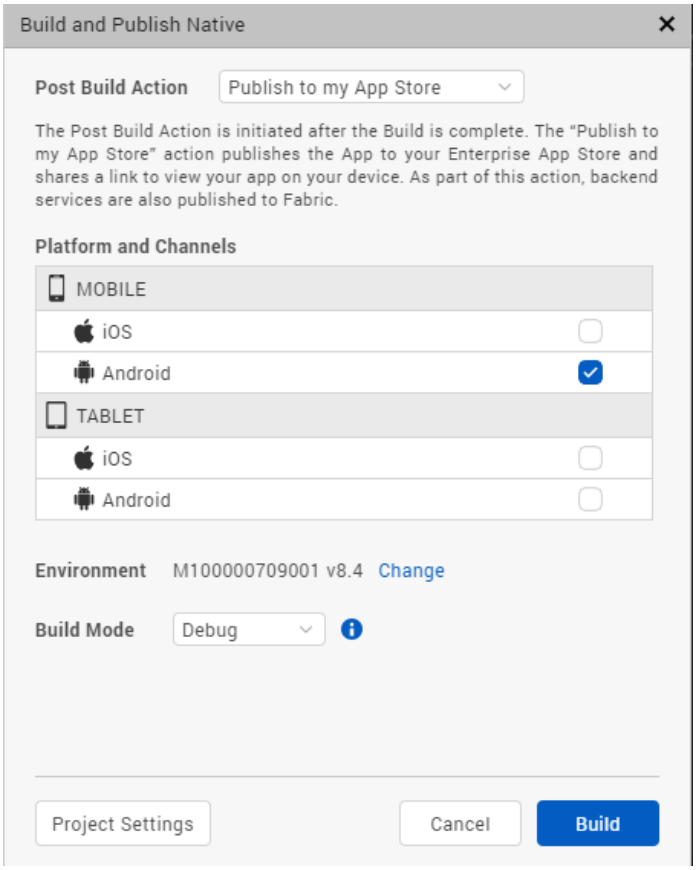
The Post Build Action is initiated after the Build is complete. You must choose the Post Build Action in the Build and Publish window, before the build process begins. There are three types of Post Build Actions, they include:

1. Publish to my App Store
2. Run on my device
3. Generate Native App

Publish to my App Store

The Publish to my App Store action generates native app binaries and publishes the application to your Enterprise App Store. After a successful publish, a confirmation window appears, which shares a link to view the Enterprise app store on your device.

To publish an app to the Enterprise App Store logging in to your Kony Account is mandatory.



> ***Note:*** You cannot build apps for the universal channel using this option.

For more information on Publish to my App Store, click here.

Run on my device

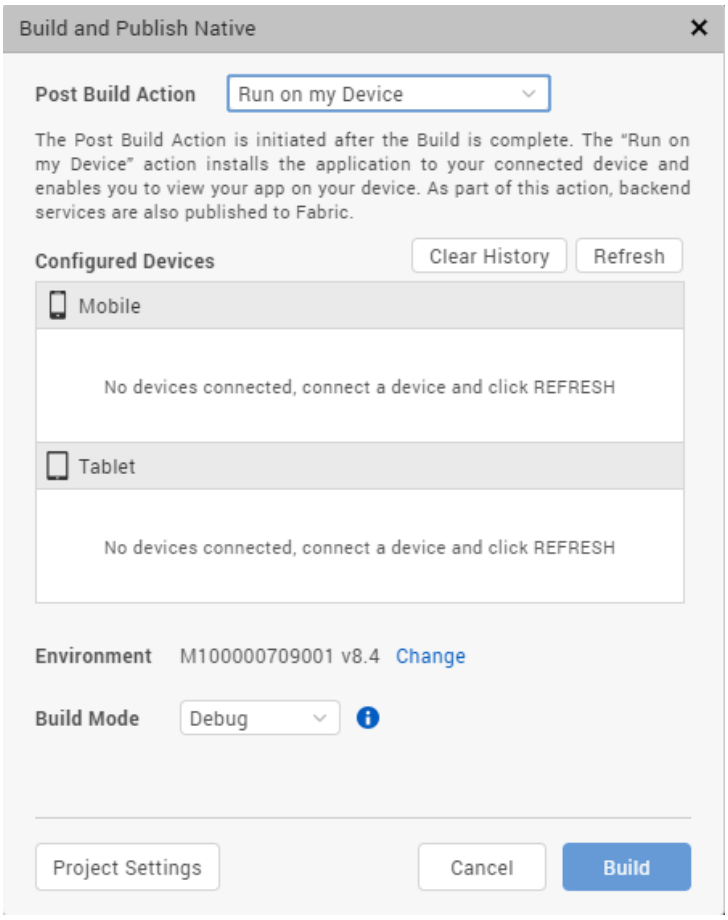
The Run on my Device action installs the application to your connected device and enables you to view your app on your device.

Establish a USB connection between the computer that built the app, and your device.

> ***Important:*** **USB Tethering for iOS devices on Windows Machine:**
> **Prerequisites** - Ensure that the latest version of iTunes is installed on the Windows machine. Before you start viewing the app on your iOS device by using the USB feature on Kony Quantum App, open iTunes on your Windows machine.

If you connect your device to the system after selecting the post build action, use the **Refresh** option to refresh the list of available devices that are connected to the system.

Use the **Clear History** option to clear out old entries of devices that are not connected to the system.



Once this action is completed, by default Android devices launch the app. Whereas, for iOS devices you need to explicitly launch the app by tapping on the app icon.

Generate Native App

The Generate Native App action generates the binaries and build logs for your Native application and saves it on your file system. The Visualizer project does not have to be linked to Kony Fabric to complete this action.

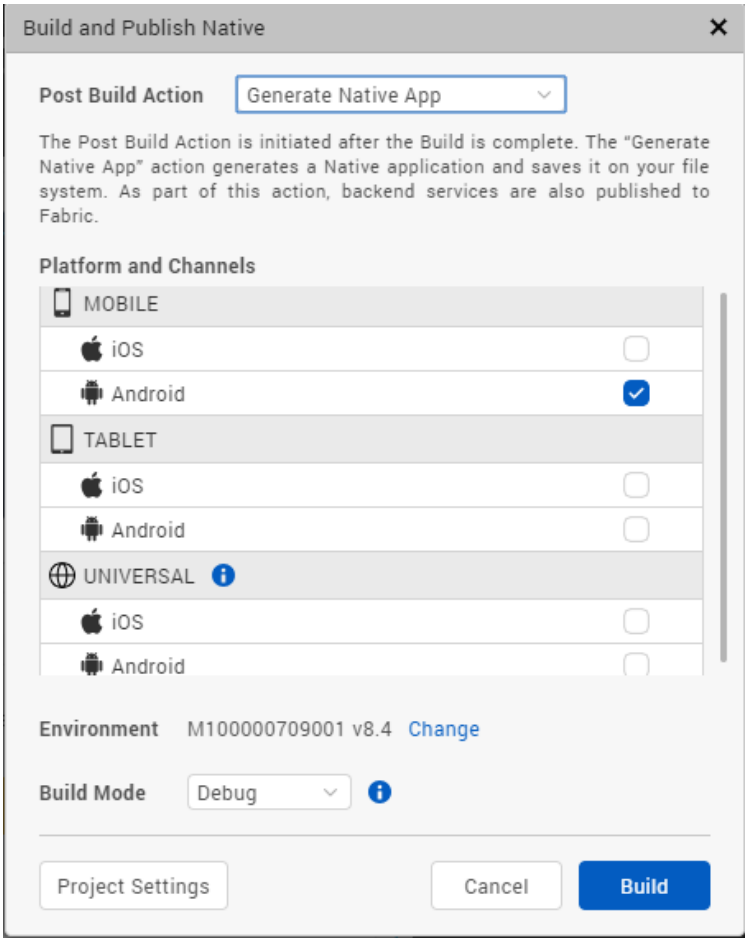
Once the build is completed,

- If you have successfully built your Visualizer project for the Android channel, you will get Android mobile and/or tablet native APKs in your project's Kony Visualizer workspace > binaries folder.
- If you have successfully built your Visualizer project for the iOS channel, you will get iOS mobile and/or tablet native IPAs in your project's Kony Visualizer workspace > binaries folder.
- You will also get the build logs in your project's Kony Visualizer workspace > binaries folder. You can refer to the logs to analyze the build for failures or success.

Alternatively, once the build is complete, you will get notified by an email from "Kony - Build Service", with download links for all these binaries.

You can generate native apps even for the Universal channel using option. This option will generate the APK or IPA for each of the platforms and channels selected.

To understand any build failures, you can go through the log file. To understand Run and Publish actions related to this type of Build, go to Post Successful Build.



Build and Publish a Native App

To build an application, follow these steps:

1. On your Kony Visualizer, from the main menu select **Build**.
2. From the context menu, select **Build and Publish Native**.
3. Select the platforms and channels for which you want to build the application.
4. From the **Post Build Action** drop-down menu, select the desired **Post Build Action**. For more details about the Post Build Action, click here.

5. You can choose to change the cloud environment on which your app will be published. To do so click on **Change** beside the **Environment** option. By default, the Environment displayed is the one that is last selected.

6. From the **Build Mode** drop-down list, select your desired build mode.

7. Click **Build**. The build generation begins. You can check the status of your build in the Build tab. If there are any errors, they appear in the Build tab. Switch to the Console tab to view a detailed log of the errors.

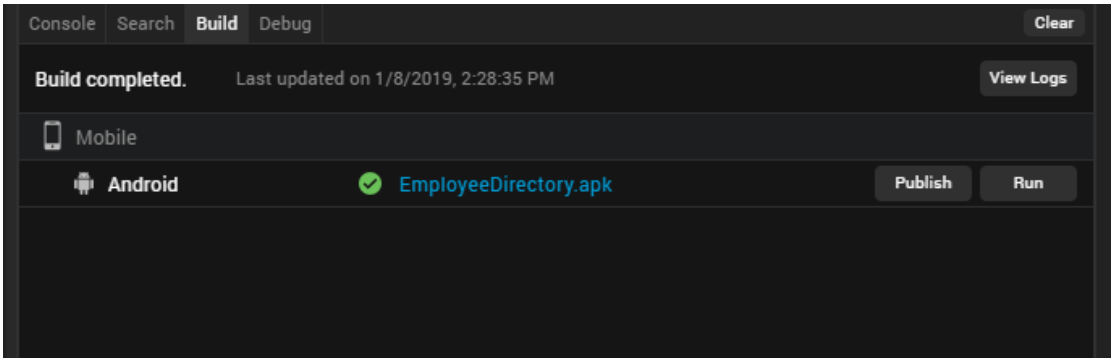
   You can check the status of your build in the Build tab. It undergoes various actions, like Project compression, uploading the compressed project to the cloud, then the actual build begins. This process may take some time.

   If there are any errors, they appear in the Build tab.

8. From the Build tab, click on **View logs** to open the build logs for the build service on your system.

Post Successful Build

Once the build is completed, the generated binaries with the download link are provided in the Build tab.



1. Click on the binaries to open them in your default web browser.
2. You can click on **Run** to run the app on your local device. If your post build action is already selected as Run on my Device, then the Run option is not enabled and there is no need to explicitly run the app again.
3. You can click on **Publish** to publish the App on Enterprise App Store. If your post build action is already selected as Publish to my App Store, then the Publish option is not enabled and there is no need to explicitly publish the app again.
4. Click on **View Logs** to view the build related logs.

Alternatively, once the build is complete, you will get notified by an email from "Kony - Build Service", with download links for all the successfully built app binaries. The mail contains details related to the Build. It contains the Project name, Build Action Triggered by, Date of Build, Build duration details. The Build Information section contains details about the Channels for which the App has been built.

- If you have successfully built your Visualizer project for the Android channel, you will get Android mobile and/or tablet native APKs link in the mail.
- If you have successfully built your Visualizer project for the iOS channel, you will get iOS mobile and/or tablet native IPAs link in the mail. The mail will also contain the OTA (plist) link, through which you can directly install the app on your device.
- You will also get the build logs. You can refer to the logs to analyze the build for failures or success.

> ***Note:*** The artifact links will be available only for 24 hours.

Publish Apps to the Enterprise App Store

Overview

Starting with Kony Visualizer V8 SP4, all Kony Cloud users have access to an Enterprise App Store that enables users to securely distribute their apps within an Enterprise.

You can access Kony Apps published from Visualizer and Fabric directly from the Enterprise App Store. Just like how you can access apps from the Android Play Store and the Apple App Store.

This feature allows you to maintain your own app store for your organization. To do so, you can create an app store on your Fabric cloud account. After an app is built, you can choose to publish the app to the Enterprise App Store available in your Kony cloud account. You can then enable and customize access for certain users, using which they can access the various apps published on the Enterprise App Store.

This feature works with the help of the Build and Publish feature of Kony Visualizer V8 SP4. Once you have built your app using the build and publish feature, you can then choose to publish it to your EAS, using the Publish to my App Store action in Visualizer.

You can publish Native Apps and Web Apps to the Enterprise App Store. For information about how to publish apps to the Enterprise App Store, refer the following links:

- Publish Native Apps to the Enterprise App Store
- Publish Web Apps to the Enterprise App Store

Prerequisites

- 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 V8 SP4 or later.
- Kony Visualizer V8 SP4 or later.
- For publishing to the Enterprise App Store, you must have Kony Fabric V8 SP4 or later.
- 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 signing details, click here.

    > ***Note:*** If you choose to build an application for the **Android** platform in **Debug mode**, no changes are required in the Project Settings. The build process begins immediately.

- 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.

Publish Native Apps to the Enterprise App Store

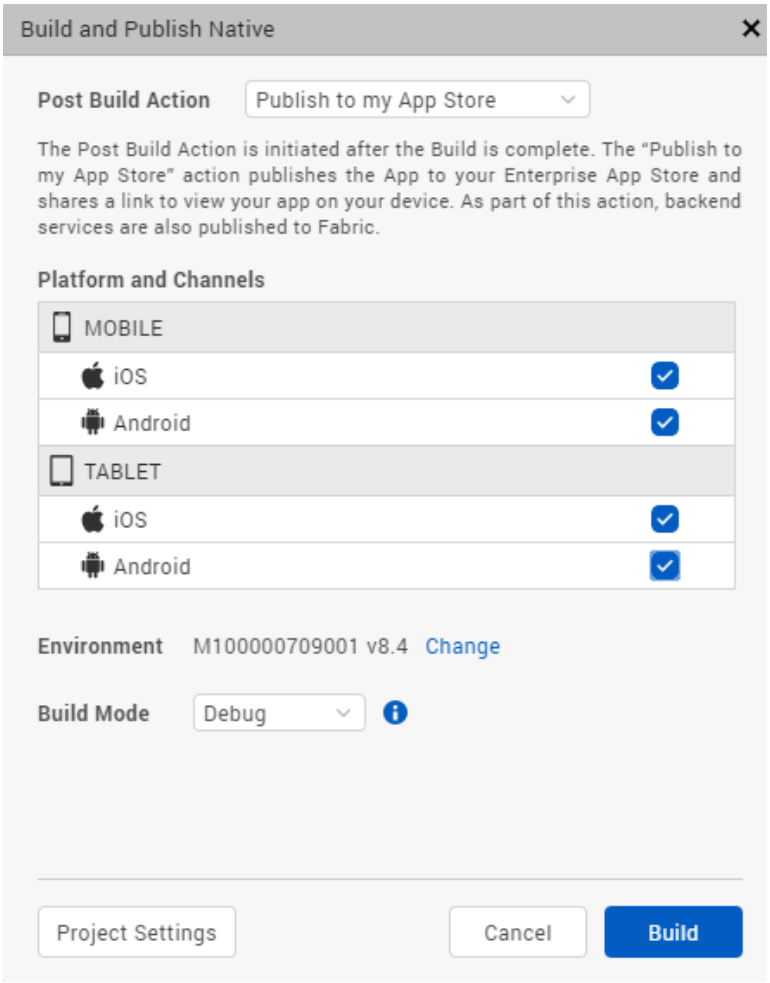
There are two ways to publish an app to the Enterprise App Store using Visualizer:

- Single click approach(Build and Publish)
- Build and then Publish

Single click approach

To build and publish an application in one step, follow these steps:

1. From the main menu on Kony Visualizer, select **Build**.
2. From the context menu, select **Build and Publish Native**.
3. Select the platforms and channels for which you want to build the application.
4. From the **Post Build Action** drop-down menu, select **Publish to my App Store**.



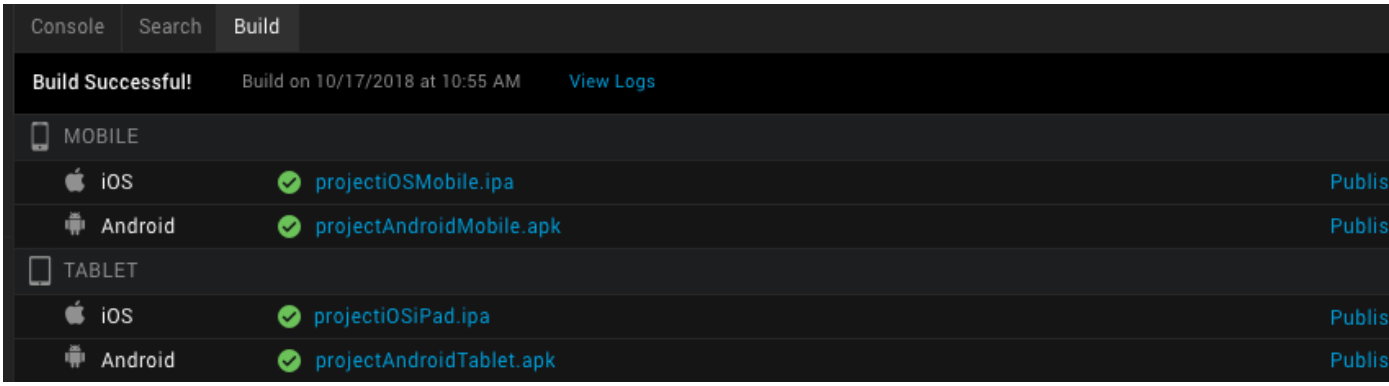
The Publish to my App Store action publishes the application to your Enterprise App Store and shares a link and a QR code to view the app on your mobile. As a part of this action, backend services are also published to Kony Fabric.

> ***Note:*** To publish an app to the Enterprise App Store logging in to your Kony Account is mandatory.

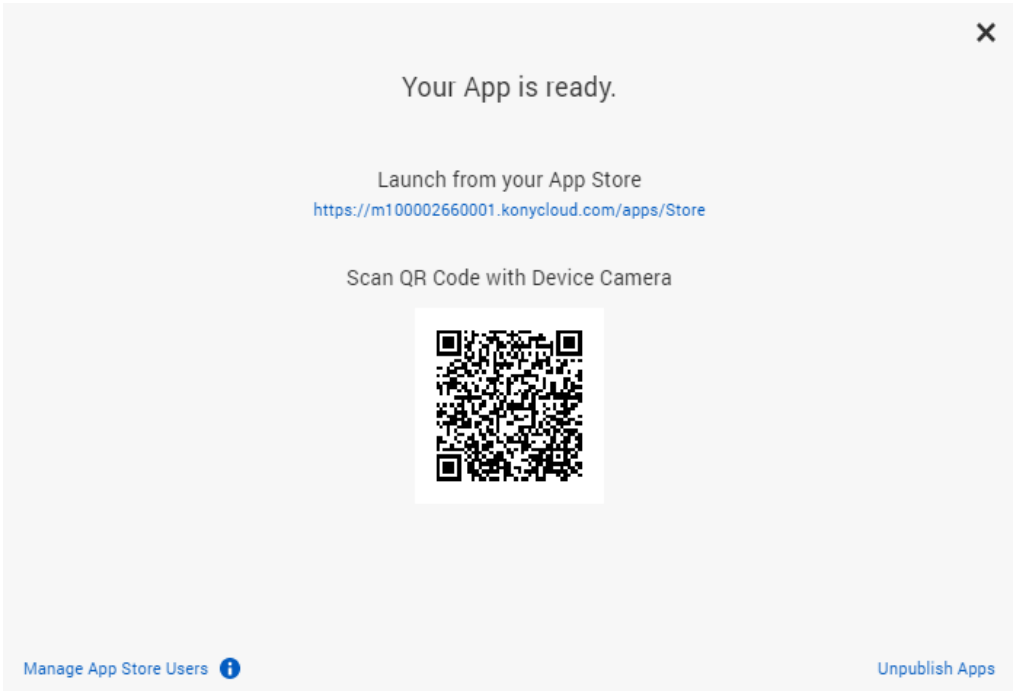
5. You can choose to change the cloud environment on which your app will be published. To do so click on **Change** beside the **Environment** option.
   To change the Environment from the Project Settings, go to **Project Settings** > **Kony Fabric** and from the drop-down menu, select your desired environment.

6. From the **Build Mode** drop-down list, select your desired build mode.

7. Click **Build**. The build generation begins.
   You can check the status of your build in the **Build** tab. It undergoes various actions, like Project compression, uploading the compressed project to the cloud, then the actual build begins. This process may take some time.



8. Upon successful build and publish, a confirmation window appears, which displays a link and a QR code to view the Enterprise App Store on your mobile device.



9. Type the App Store link on a browser on your mobile or scan the displayed QR code to launch the Enterprise App Store on your mobile.

> ***Note:*** For the first time, any user can access the published app, unless access to the app store is restricted by managing app store users.

10. Your published app will be listed with the same name as your Visualizer Project. Click on **GET** to fetch the app from the Enterprise App Store and launch it on your device.

    The logo of the application will be the image you set in **Project Settings** > **Native** > **Application Logo.** If you do not set an Application Logo, a default icon is displayed as the application logo.

Build and then Publish

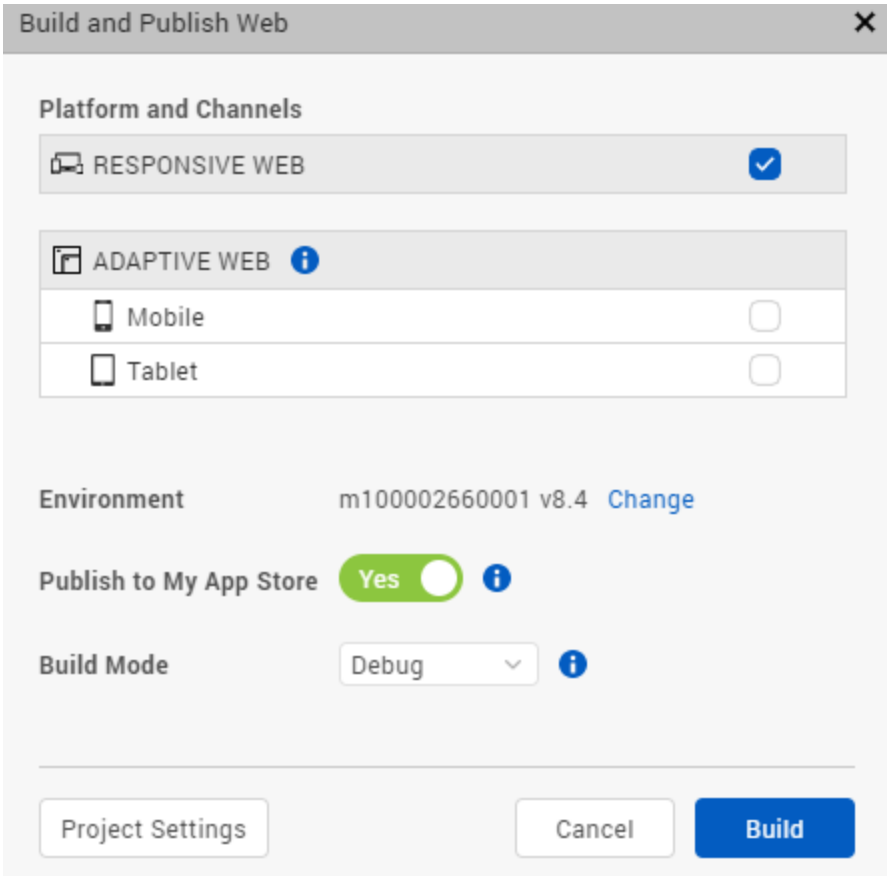
To build and then publish an application you can choose either of the other Post Build Actions available. The other two **Post Build Actions** are:

- Run on my Device: The Run on my Device action installs the application to your connected device and enables you to view the app on your device.
- Generate Native App: The Generate Native App action generates the binaries for your Native application and saves it on your file system.

Publish Web Apps to the Enterprise App Store

To build and publish a Web application, follow these steps:

1. From the main menu on Kony Visualizer, select **Build**.
2. From the context menu, select **Build and Publish Web**.
   The Build and Publish Web window appears.



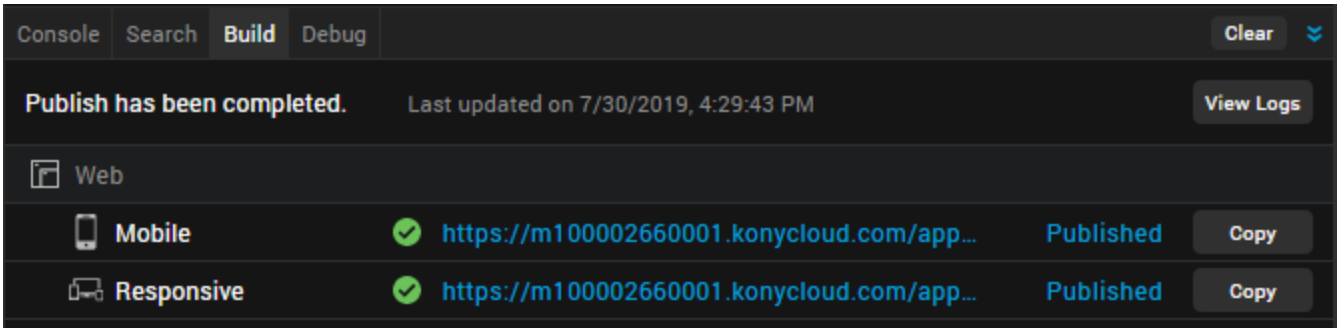
3. Select the channels for which you want to build the application.
4. You can choose to change the cloud environment on which your app will be published. To do so, click **Change** next to the **Environment** option.
   To change the **Environment** from the Project Settings, go to **Project Settings** > **Kony Fabric** and from the drop-down menu, select an environment.
5. Enable the **Publish to my App Store** option.

   > ***Note:*** To publish an app to the Enterprise App Store, it is mandatory to log on to your Kony Account.

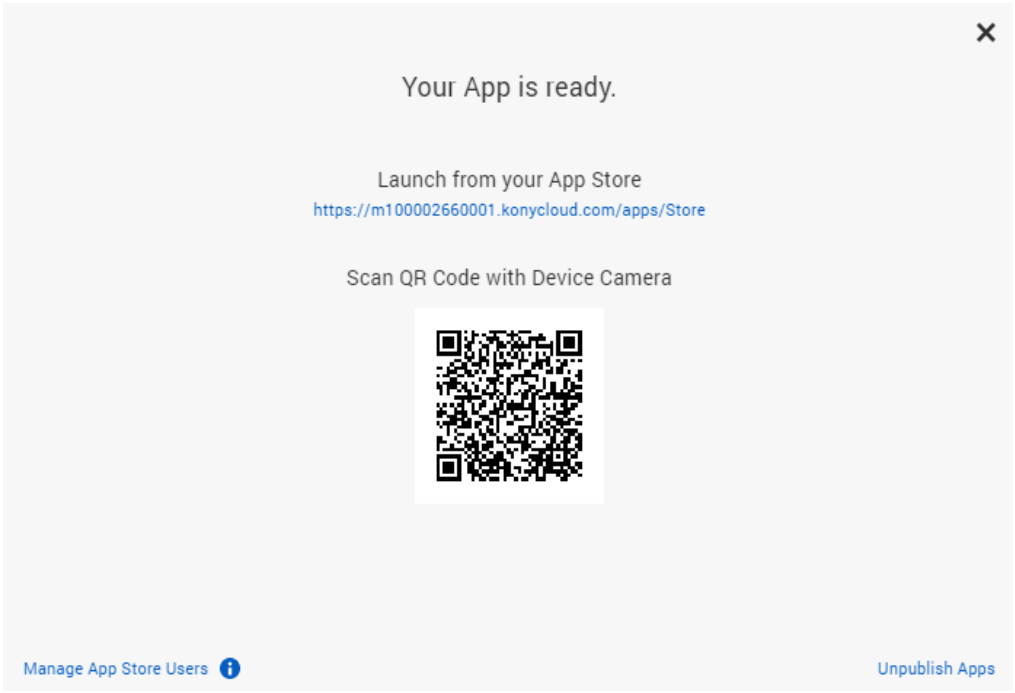
6. From the **Build Mode** drop-down list, select a build mode.

7. Click **Build**. The build generation begins. As a part of this action, backend services are also published to Kony Fabric.

   You can check the status of your build in the **Build** tab. Before the actual build starts, various actions such as project compression and upload of the compressed project to the cloud are performed. This process may take some time.



8. Upon successful build and publish, a confirmation window appears. The window will display a link and a QR code to view the Enterprise App Store on a browser of your mobile device.

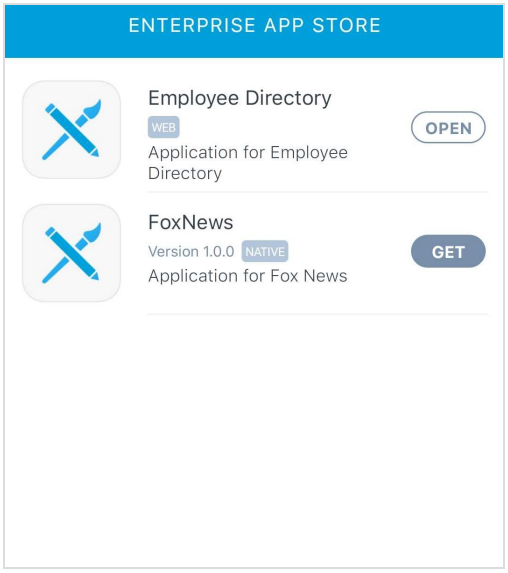


9. Type the App Store link on a browser of your mobile or scan the displayed QR code to launch

the Enterprise App Store on your mobile device.

> ***Note:*** When an App is published for the first time, any user can access the app, unless access to the app store is restricted by managing app store users.

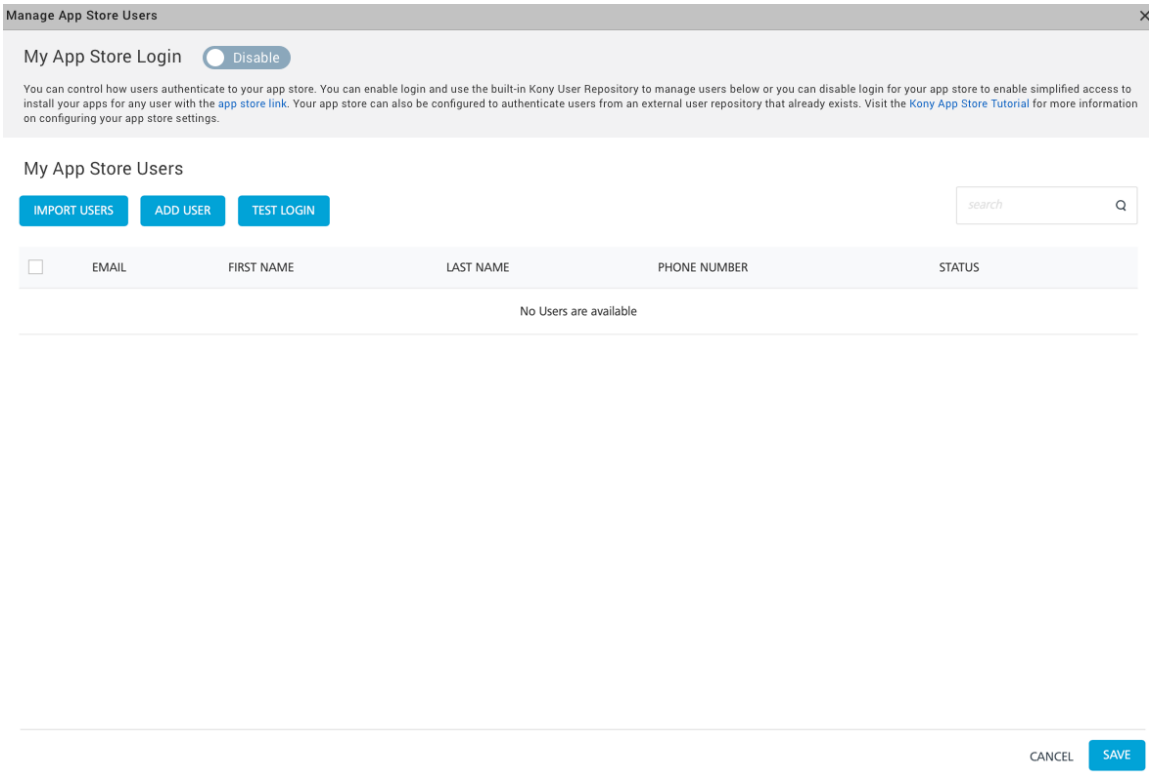
10. The published app will be listed on the Enterprise App Store with the same name as your Visualizer Project. Click on **Open** to fetch the app from the Enterprise App Store and launch it on a browser of your mobile.



Manage App Store Users

The Enterprise App Store(EAS) is an application to distribute your native apps. An organization that uses Kony EAS can customize the user access based on their requirements. To view and manage the active user list for any Kony EAS account there exists the Manage App Store Users window within Kony Visualizer.

You can access this window from the confirmation window that appears after the successful publish of an app to the EAS. The link to the application on the Enterprise App Store can be accessed by clicking on the **My App Store** button on the Header of the Visualizer App Canvas. You can also access the link to the application on the Enterprise App Store from the **Project Settings** > **Kony Fabric** page.



To Manage app store users from Visualizer, do the following:

1. From the confirmation window, click on the **Manage App Store Users** link. The Manage App Store Users window appears.
   Using this window, you can manage users, enable access to install your apps for any user that has the app store link.

2. You can import users, add new users and perform a test login from the Manage App Store window. These actions are specific to Kony User Repository.

   > ***Note:*** By default, the Manage Users Login is disabled. The default authentication type for Kony EAS is Kony User Repository. So when an app is published to the EAS, all users registered in the Kony User Repository are authorized to access this app for the first time. You can change the authentication from Kony Fabric.

3. On the **Manage App Store Users** window, toggle the **My App Store Login** switch to enable the access settings. You can now control the user authentication and access to your app store.

4. You can go to Kony Fabric and enable authentication to authenticate users with the Identity services available in the Manage App Store Users window.

   You can set up an identity service based on the type of the users who are allowed to access your application. For example: To restrict access to your organisation'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, SiteMinder or Kony User Repository authentication) and social identity providers (Google, LinkedIn, Instagram, Amazon, Microsoft, Yahoo, BOX, Facebook).

   > ***Note:*** Setting up an identity service is optional. You can choose not to implement any authentication services for your application.

> ***Important:*** For more information on Managing the Enterprise App Store, refer to Enterprise App Store on Kony Fabric.

Cancel publish to the EAS

You can cancel the publish during the build and publish process. Click on the **Cancel** button available in the Build tab in Visualizer to cancel the build and publish process at any time.

You cannot open a different project when one project is being built in Visualizer.

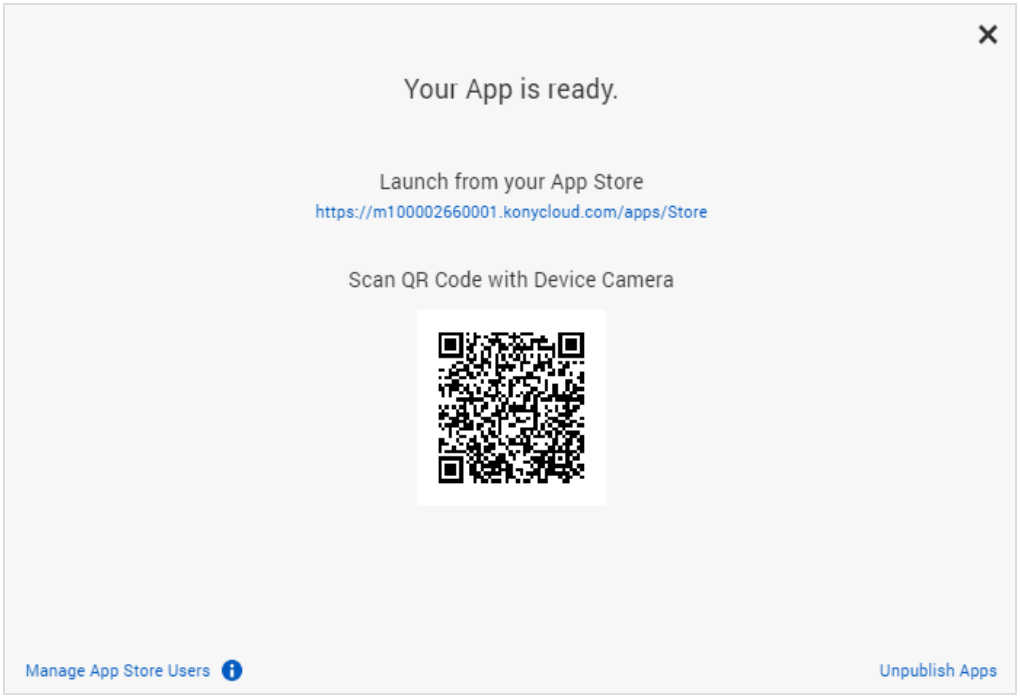
Unpublish an App from the EAS

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 Visualizer. The Unpublish option is available in the confirmation window that appears after the successful publish of an app to the EAS.

To unpublish a Kony Visualizer app from the EAS, follow these steps:

1. In Visualizer, from the blue header above the App Canvas, click **My App Store**.

   The My App Store dialog box appears.



2. Click **Unpublish App**. The **Select Apps to unpublish** dialog box appears.
3. Select the channels from which you want to unpublish the app, and then click **Unpublish**. The progress of the app-unpublish process appears in the Build tab. Once the app is successfully unpublished, a confirmation message appears.

> ***Note:*** You can also unpublish the published app through the Kony Fabric console. For more information, refer Enterprise App Store on Kony Fabric.

Publishing a Web App in Kony Visualizer

Applies to *Kony Visualizer*.

Overview

Web Publish, a feature introduced in Kony Visualizer V8 SP4, enables you to build and publish a web app from Kony Visualizer to your Kony Fabric cloud.

Seamless in-app Build and Publish Experience

Starting with Kony Visualizer V8 SP4, you can publish a web app to Kony Fabric, without leaving the Visualizer window. Earlier, one had to design their web app in Visualizer and then open the Kony Fabric console to the publish the web app. This process has now been simplified.

The Build and Publish Web feature is available in the Build menu of the Kony Visualizer application.

The publish process occurs in the background, and you can continue to use the Visualizer canvas while the app is being built and published.

Access to the Published URL within Kony Visualizer

Once the publish is complete, the published URL is displayed on the Build tab in Kony Quantum App. When you click the URL, the Visualizer Preview window opens. For more information on the Visualizer Preview window, click here.

You can also click on **Copy** beside the URL to copy it and view the app in a web browser of your choice.

The Build tab in Visualizer displays the status of your app publish. If there are any errors, they appear in the Build tab. You can switch to the Console tab for a detailed view of the error logs.

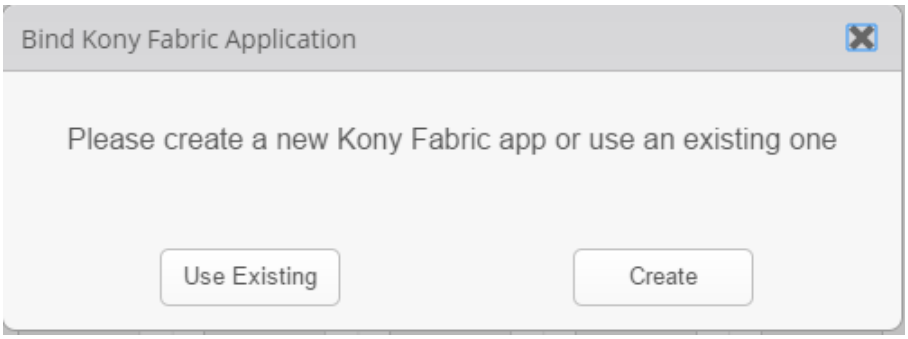
Prerequisites

Following are the prerequisites to publish a Web App within Kony Visualizer:

1. Sign in to your Kony Cloud account.
2. Link a new or an existing fabric app to your Visualizer Project.

When you try to publish the app without logging into Kony Fabric, the login window appears.

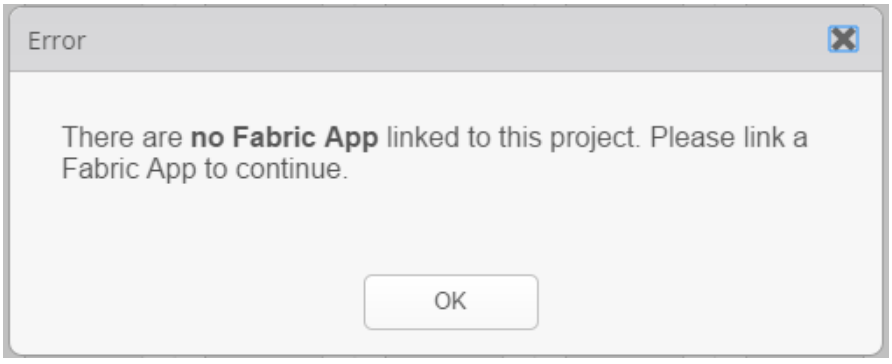
If you try to publish the app, even when you do not have an associated Kony Fabric app, you will get a prompt to link a Kony Fabric app.



To change the selected Cloud account or Environment, go to **Project Settings** > **Kony Fabric**. hen, click the **Kony Fabric** tab. At the top of this tab, under Kony Fabric Environment, select an environment from the drop-down list. Click **Done**. If you do not see any environments listed, you need to create one. For more information, refer Environments in the Kony Fabric Console User Guide.

To publish the Kony Visualizer web app to Kony Fabric, the app needs to be associated with a Kony Fabric app.

If no Fabric App is associated to your Visualizer project, an error message is displayed.



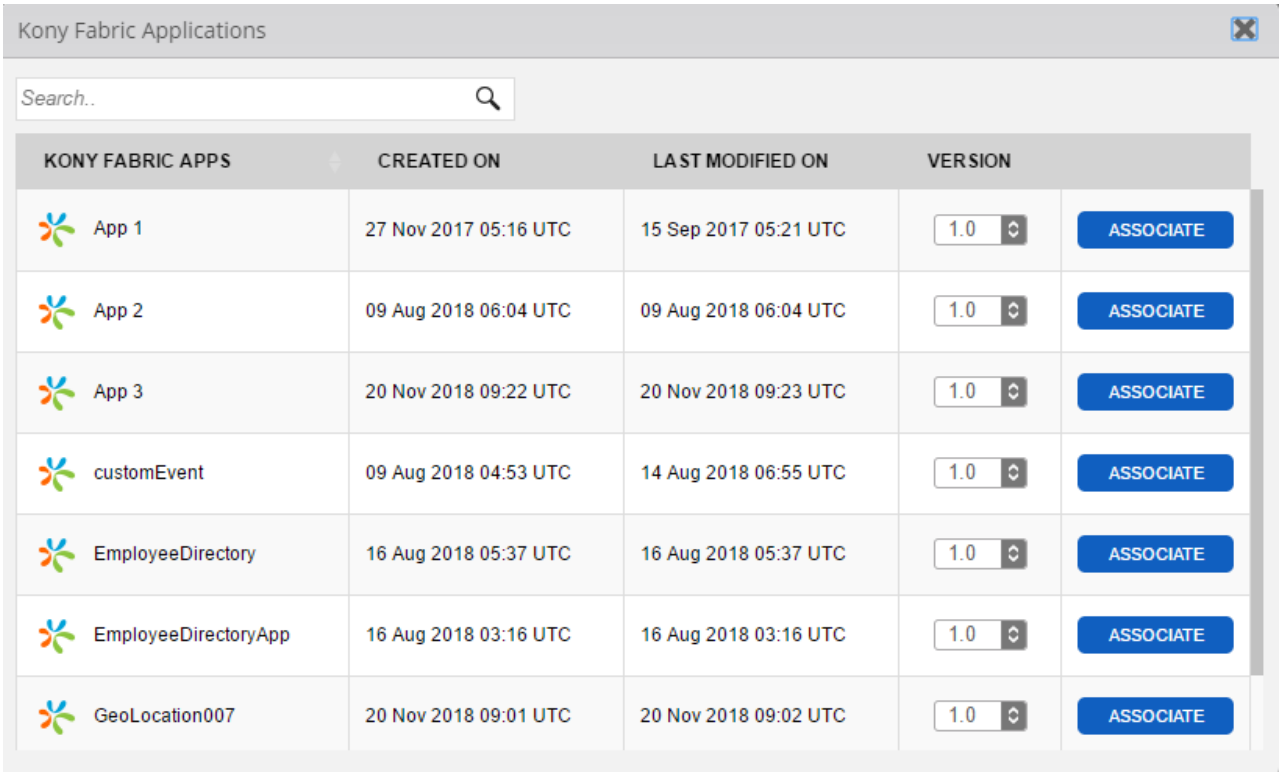
You must either create a new Kony Fabric app or use an existing one. For information on how to create a new Kony Fabric app, see How to Add Applications in the Kony Fabric Console User Guide.

Publish a Web app

> *Important:* Before you publish a web app, ensure that you have a project in which you have a Web app designed.

To publish a Web app, follow these steps:

1. Open the Kony Visualizer project in which you have designed a web app.
2. Sign in to your Kony Cloud account.
3. Link your Visualizer Project to a Kony Fabric app. You can link your Project to a new or an existing Fabric app.
4. On the **Data & Services** panel, click the hamburger menu near the **Refresh** button.
5. Click either **Create New App** or **Link to Existing App**. The Kony Fabric console appears.
6. If you have not already, sign in to your Kony Fabric account by using your cloud credentials.
7. If you choose:
    a. **Create New App**, a new Fabric app with the same name as the Visualizer Project is created.
    b. **Link to Existing App**, a list of existing apps from your Kony Fabric account are displayed.
8. To select the Kony Fabric app that you want to associate your Kony Visualizer app with, click **Associate** beside the App Name.

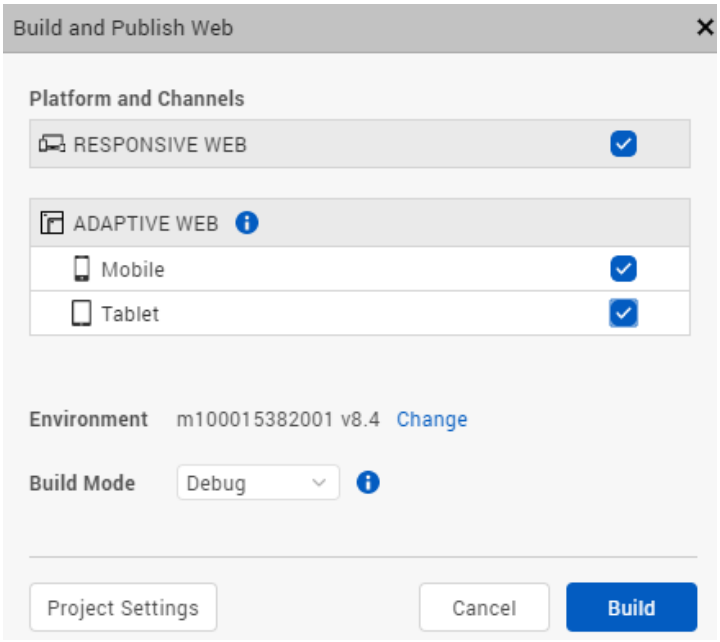


The app you chose/created is successfully linked to your Visualizer Project.

9. Close the Kony Fabric window and return to Kony Visualizer by clicking on the Visualizer icon on the left navigation pane.

10. On your Kony Visualizer, from the main menu select **Build**.
    A context menu appears

11. From the context menu, select **Build and Publish Web**.
    The Build and Publish Web window appears.



12. Select the required Platforms and Channels for which you want to build and publish your web app.

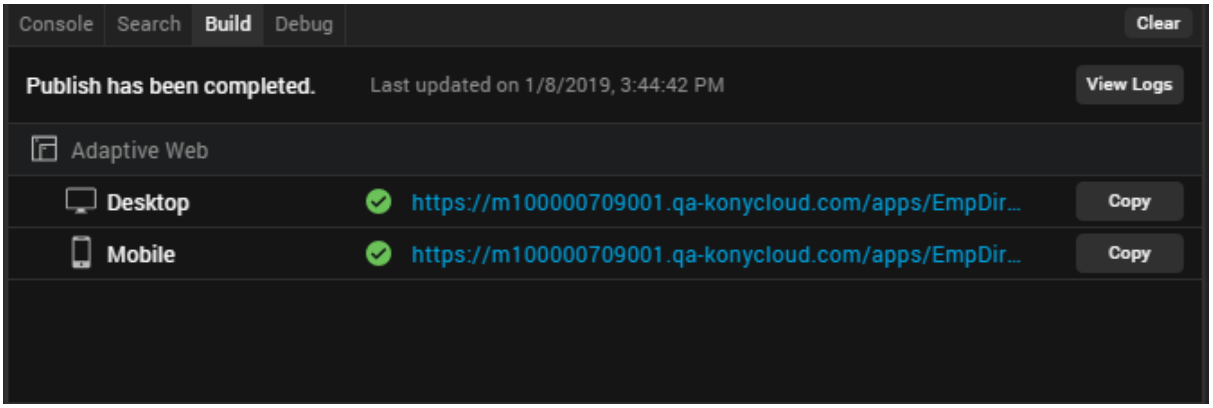
13. From the **Build Mode** drop-down list, select your desired build mode.

14. You can choose to change the cloud environment on which your app will be published. To do so click on **Change** beside the **Environment** option.

15. Once all the required settings are ready, click **Build**.
    The progress of the Web App Publish is displayed in the Build tab of Visualizer.
    Once the publish process is complete, a Web App URL appears on the Build tab.

16. Click on the generated URL to preview your Web app in the **Live Preview** mode.
    The Visualizer Preview window appears. You can use this window to debug the application.

17. You can also click on **Copy** to copy the URL to your clipboard and view it in a web browser of your choice.



18. Click on **View Logs** to access the logs of the publish process.

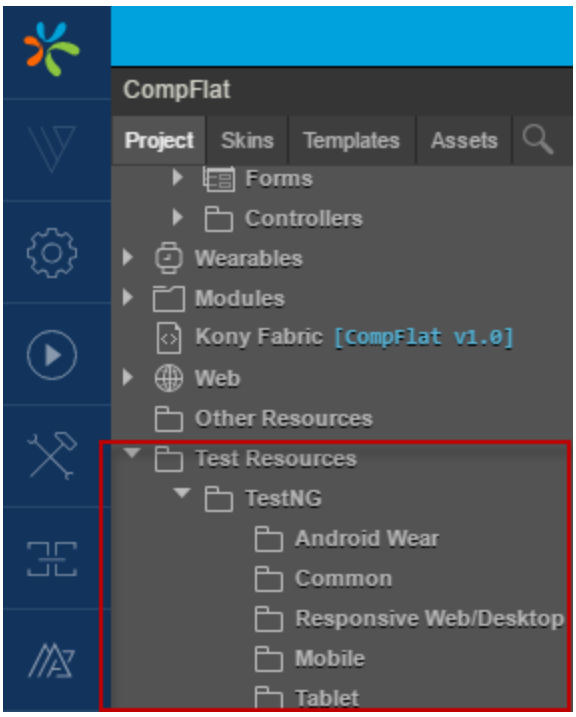
19. An **Application was deployed mail** will also be sent to your registered e-mail account. This mail will contain the App details along with the App URL. Click on the URL in the mail to open the web app in your default web browser.

Use Test Scripts in Kony Visualizer

You can create test scripts and use them in Kony Visualizer from Kony Visualizer SP3 GA onwards. Even though testing scripts were supported by Kony earlier, they were never supported in Kony Visualizer directly. In previous releases of Kony Visualizer, you could create scripts and place them in the Other Resources folder in the Project View of Kony Visualizer.

The new Test Resources folder in Kony Visualizer allows you to create test scripts for specific channels such as Mobile, Tablet, Desktop Web etc. The new feature also allows you to use multiple testing frameworks. For each one of the testing framework, the test resources folder has separate child folders for each channel. This allows you to easily identify the script types and use the ones which are appropriate for the channel you are testing. You can now manage your scripts directly from the Kony Visualizer UI.

The Test Resources folder contains TestNG and Jasmine folders by default. The folders further contain default channel folders.



You can create a new test script specific to the channel by placing a test script within the channel's sub folder. If the file is created under the TestNG or Jasmine sub directory, the new file contains an empty shell with the basic TestNG or Jasmine test script. When you make edits to a script, it is saved when the project is saved.

The following sections in the document describe about creating and executing testscripts for TestNG and Jasmine test frameworks.

Create a test script for TestNG

Execute test script for TestNG

Create a test script for Jasmine

Execute test script for Jasmine

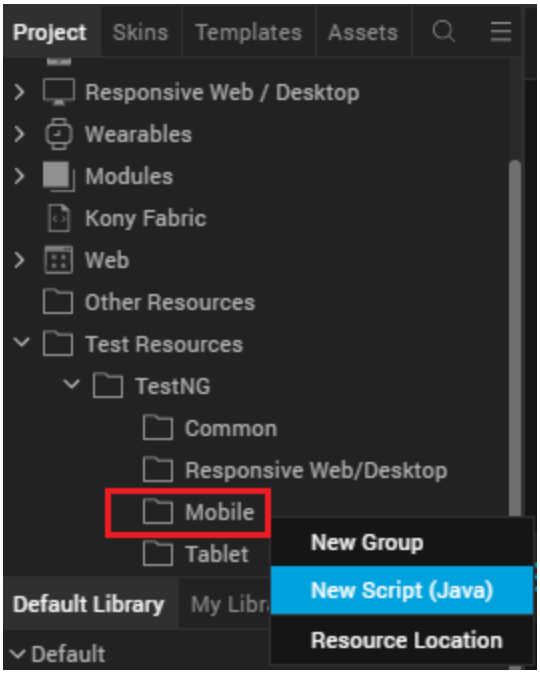
Debug test script for Jasmine in Android

Test Automation in Jasmine

Create a New TestNG Test Script

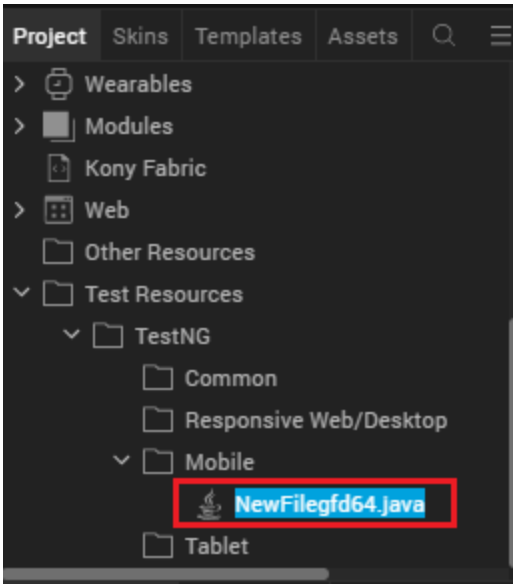
To create a new test script in TestNG folder, do the following:

1. In Kony Visualizer, open the project you want to create the test scripts.
2. Navigate to the TestNG folder in the Project Panel.
3. Click on the drop-down icon of the channel in which you want to create the test script.



4. Select **New Script (Java)**

   A new test script is created



5. Edit your script using Java.

Execute your Test Script in TestNG

To execute test scripts in TestNG, do the following:

1. Open **Visualizer**.
2. In the File menu, navigate to **Product** > **Build**.

   Build the app for the channel you are testing the script.
3. Run your app on the simulator.
4. Copy the workspace location of your test scripts.
5. Open Command Prompt on your system.
6. Run the following command

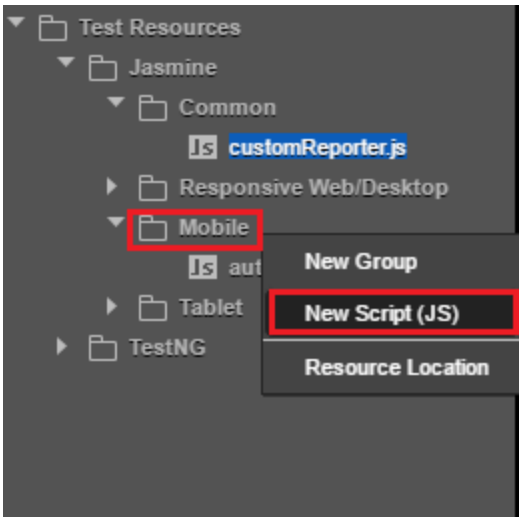
ant build.xml/Project name/<location_of_scripts_in_workspace>

Your test scripts are executed.

Create a New Jasmine Test Script

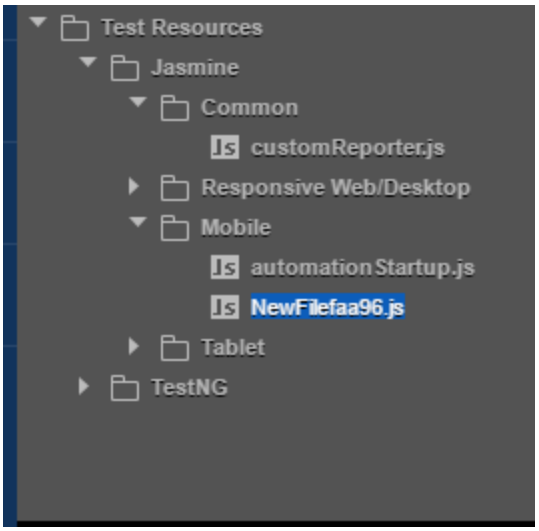
To create a new test script in the Jasmine folder, do the following:

1. In Kony Visualizer, open the project that you want to create the test scripts for.
2. From the Project explorer, go to **Test Resources** > **Jasmine**.
3. Click on the drop-down icon of the channel in which you want to create the test script and then select **New Script (JS)**.



> ***Note:*** It is not mandatory to create the new file. You can write all your automation scripts in the **automationStartup.js** file, that is auto generated. You must create a new file only when you want to segregate test scripts for each of your app flow.

4. A new test script is created.



5. Enter your automation code in the new test script created. For instance, following is the code for testing a button on-click action.

```
describe("sample", function() {

    beforeEach(function() {
        //gets executed once before each spec in the describe
    });

    afterEach(function() {
        //gets executed once after each spec.
    });

    it("sample_testCase", async
        function() {
            kony.automation.button.click(["Form1",
"Button0ec5b6258f1504e"]);
        });

});
```

To execute the script, enter the code in the following format, in the automationStartup.js file.

```
require(["testScripts/sample-suite"], //prepending testScripts
    function() {
        jasmine.getEnv().execute();
    });
```

You must prepend each file that is being required with a key word **testScripts**. In the require callback, you must appropriately call the jasmine execution. If you want to require multiple files, jasmine execution must be called in the last required file.

You will notice an auto generated script **customReporter.js** in the Common folder. The customReporter.js file contains several jasmine test reporting callbacks (jasmineStarted, suiteStarted, specStarted, specDone, suiteDone, and jasmineDone). Using the customReporter test script, you can configure where you want to export your test results. When you build any of your app, the file customReporter.js is added to mobile, tablet, and desktop web channel test folders.

> *Important:* The name **customReporter.js** file is reserved for the Jasmine test scripts common js file. If you name any of your test scripts with customReporter.js, testing feature as explained in this section will not work. Ensure that you give unique test javascript file names across channels.

With the customReporter.js test script, you can access all Kony APIs and methods used in Kony Visualizer . In any test script that you create on your own in the common folder, you can only access the kony.print API. Ensure that you do not modify anything in the customReporter.js. You can write your code in the jasmine callbacks.

Execute your Test Script in Jasmine

To execute test scripts in Jasmine, do the following:

1. Open **Visualizer**.
2. From the **Project** Explorer, right click on **Jasmine** and then select **Deploy**.

3. From the File menu, go to **Product** > **Build**.

   Build the app for the channel that you are testing the script, in **debug mode**.

4. Run your app on the target simulator or device.

   > ***Note:*** The test scripts are hosted on the Visualizer node server. The target device should have access to this server (Visualizer needs to be running). Ensure that the system on which the Kony Automator runs and the target device are on the same network. If the network changes, you must rebuild the app.
   >
   > Although the system and the target device are connected on the same network, you may have trouble executing your test script when the app uses an inaccessible IP address. To modify the IP address, follow these steps:
   >
   > 1. Open the **konyviz_preferences.json** file located at **Users\<user ID>\Kony Visualizer\ vizdata**.
   > 2. Set the "**autoDetectIP**" key as **false**.
   > 3. Under the "**general**" Object, specify the required IP address against the "**staticIP**" key.

> ***Note:*** In Jasmine, the `jasmine.DEFAULT_TIMEOUT_INTERVAL` property is not considered for Native channels. This is because the Async/Await command support in Jasmine is only applicable for the Responsive Web channel, but not for Native. However, if you specify an Async/Await command as part of your test script for Native channels, your script still works as expected.

For more information about APIs and automation in Kony, click here.

Debug Test Script for Jasmine in Android

To debug an Android application while using Jasmine test scripts, follow these steps.

1. In Kony Visualizer, from the File menu, go to **Product** > **Build**.
2. In the **Build Generation** dialog box, select Android platform.
3. Select the **Build Mode** as debug and click **Build**.
4. If you have not set the debugging as true:
    i. Go to the <workspace location>/<app name> folder.
    ii. Open the **projectProperties.JSON**file and then add the following script.

```
"enableJasmineJSDebuggingForAndroid" : "true"
```

5. Switch back to Kony Visualizer. From the **Project** Explorer, right-click **Jasmine**, and then click **Deploy**.
6. Start the app on your device. The **Waiting for Debugger to connect...** dialog box appears on your device.
7. From Kony Visualizer, go to **Product**>**Debug As**>**Debug android application**. The Google Chrome web browser opens with the chrome devtools URL.
   For example,
   chrome-devtools://devtools/bundled/inspector.html?experiments=true&v8only=true&ws=127.0.0.1:9222

   For more information about how to debug an Android application, click here.

> ***Note:*** The chrome devtools URL uses only the **9222** port while using Jasmine test scripts.

Jasmine Test Automation

From Kony Visualizer V9 onwards, Kony introduces the test automation feature using which you can leverage the capabilities of the Jasmine test framework to record activity on an application and generate test scripts. Further, you can edit the recorded test script by adding certain actions such as Wait, Assert, and Scroll to Widget. Once you configure the actions, you can run the generated test scripts using the Live Preview option in Visualizer on your system, or on a mobile device to test the application.

There are three types of test resources in Jasmine based on their hierarchy:

- Test case
- Test suite
- Test Plan

Test Case

Test Case is a test script that is written by a user or generated by recording the user's activity on the application.

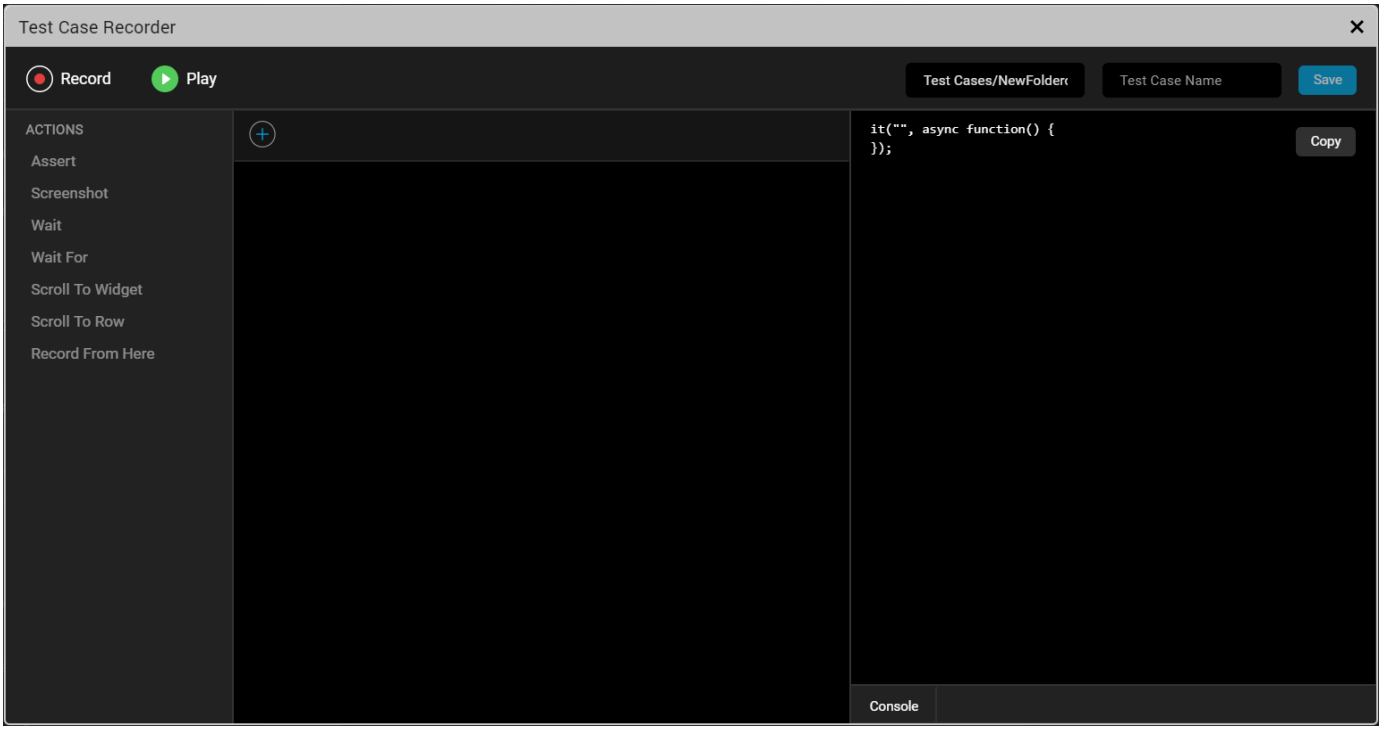
Create a Test Case

You can create a test case by recording your activity on the app. The app must be previewed by using the Live Preview feature or launched on a device by using the Build and Publish feature. You can then record the user activity on the app to create a test case. Further, you can edit the recorded test case while or after creating the test case.

Using Live Preview

If you use the live preview feature to view your app, follow these steps to create a test case for the app:

1. In Visualizer, from the main menu, navigate to **Build** > **Live Preview Settings**.
   The **Live Preview Settings** window appears.
2. Select the platforms and channels for which you want to build the application.
3. From the **Preview Mode** drop-down list, select **Test**.
4. Click **Save and Run**.
   The app launches on the device. The **Visualizer Preview** window appears.
5. In Visualizer, from the **Project** Explorer, go to **Test Resources** > **Jasmine**.
6. Expand the required <channel>, right-click on **Test Cases** or any of the sub folders and select **Create a Test Case**.
   The **Test Case Recorder** window appears.



Note:

- When you minimize the **Test Case Recorder** window, a **View** option appears in the **Test** tab on the lower pane of Visualizer. You can click **View** to re-open the window that you have minimized.
- When you minimize the **Test Case Recorder** window while playing a test case, a progress bar appears in the **Test** tab indicating the progress of test case being played. Once the test case is played completely, a toast message appears on the lower-right of Visualizer.
- After minimizing the **Test Case Recorder** window, a few options such as **Edit**, **Delete**, and **Rename** are disabled for the test files. Also, the options to create a new Test Case, Test Suite, or Test Plan are disabled.

7. Re-launch the application.
   The device connects successfully.

> ***Note:*** If the device does not connect successfully, refer this FAQ.

8. On the upper-left corner of the window, select **Record** to start the recording.

> ***Note:*** Any activity performed by a user on the application is recorded until the user clicks **Stop**.

9. After the recording starts, a **stop** option appears in the **Test Case Recorder** window. To discontinue the recording, click **Stop**.
   You can now edit the test case to add more actions, if required. To know more about editing the test case, click here.

10. On the upper-right corner of the window, type a name for the test case and click **Save**.
    The test case is created.

Using Native Devices

If you want to connect to a native device instead of viewing the app using live preview, ensure that your system and the device on which you want to run the app are on the same network.

In this case, follow these steps to create a test case:

1. In Visualizer, from the main menu, navigate to **Build** > **Build Native Local**.
   The **Build Native Local** window appears.

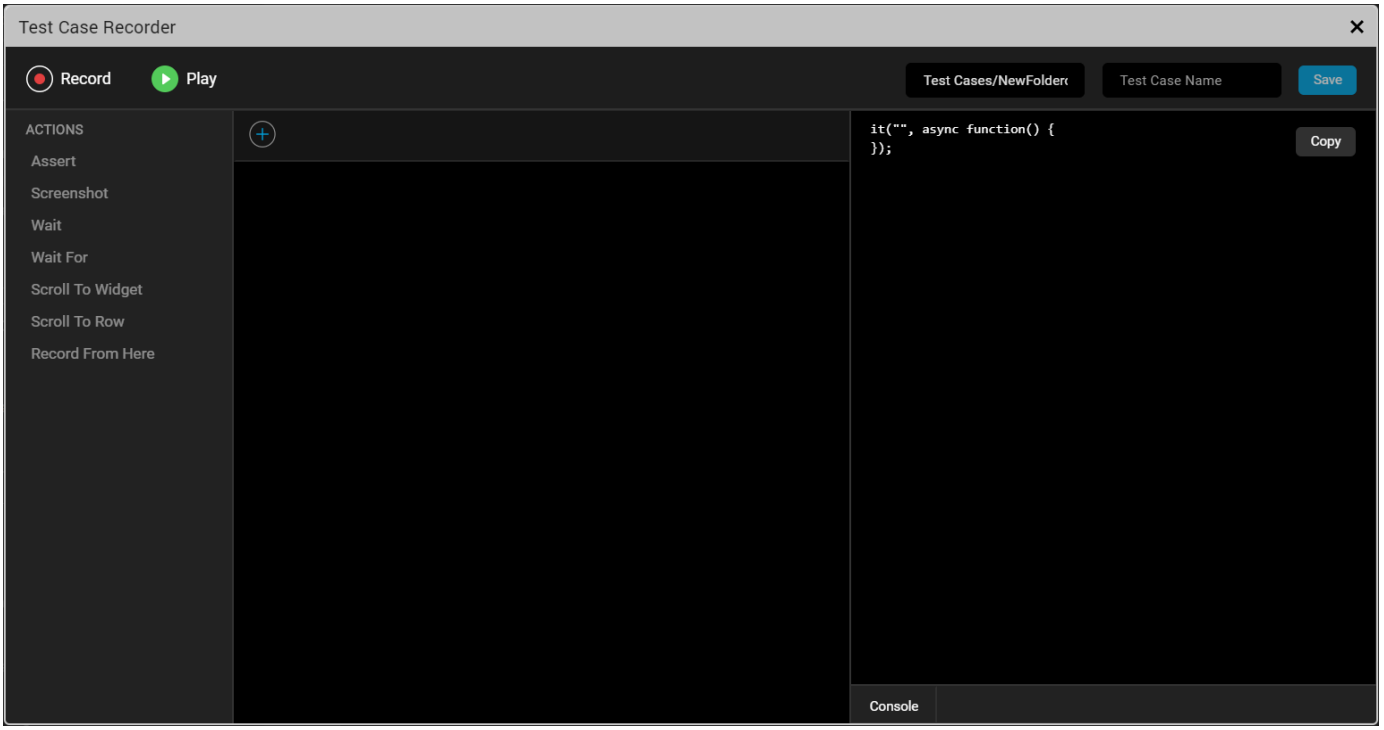
> ***Note:*** The **Build Native Local** feature enables you to build an app on the local machine. To know more about the feature, click here. If you want to build the app on the cloud, select **Build and Publish Native**.

2. Select the platforms and channels for which you want to build the application.

3. From the **Post Build Action** drop-down menu, select **Run on my Device**.

4. From the **Build Mode** drop-down list, select the **Test** mode.
5. Click **Build**. The build generation begins. You can check the status of your build in the Build tab. If there are any errors, they appear in the Build tab.

   Once the build is completed, the app is launched on the device.
6. In Visualizer, from the **Project** Explorer, go to **Test Resources** > **Jasmine**.
7. Expand the required <channel>, right-click on **Test Cases** or any of the sub folders and select **Create a Test Case**.
   The **Test Case Recorder** window appears.



> ***Note:***
>
> - When you minimize the **Test Case Recorder** window, a **View** option appears in the **Test** tab on the lower pane of Visualizer. You can click **View** to re-open the window that you have minimized. For more information, click here.

8. Re-launch the application.
   The device connects successfully.

   > ***Note:*** If the device does not connect successfully, refer this FAQ.

9. On the upper-left corner of the window, select **Record** to start the recording.

   > ***Note:*** Any activity performed by a user on the application is recorded until the user clicks **Stop**.

10. After the recording starts, a **stop** option appears in the **Test Case Recorder** window. To discontinue the recording, click **Stop**.
    You can now edit the test case to add more actions, if required. To know more about editing the test case, click here.

11. On the upper-right corner of the window, type a name for the test case and click **Save**.
    The test case is created.

To play the recorded script, re-launch the app and click **Play**. The recorded script plays in the **Visualizer Preview** window. You can view the result of the test case in the **Console** tab.

Edit the test Case

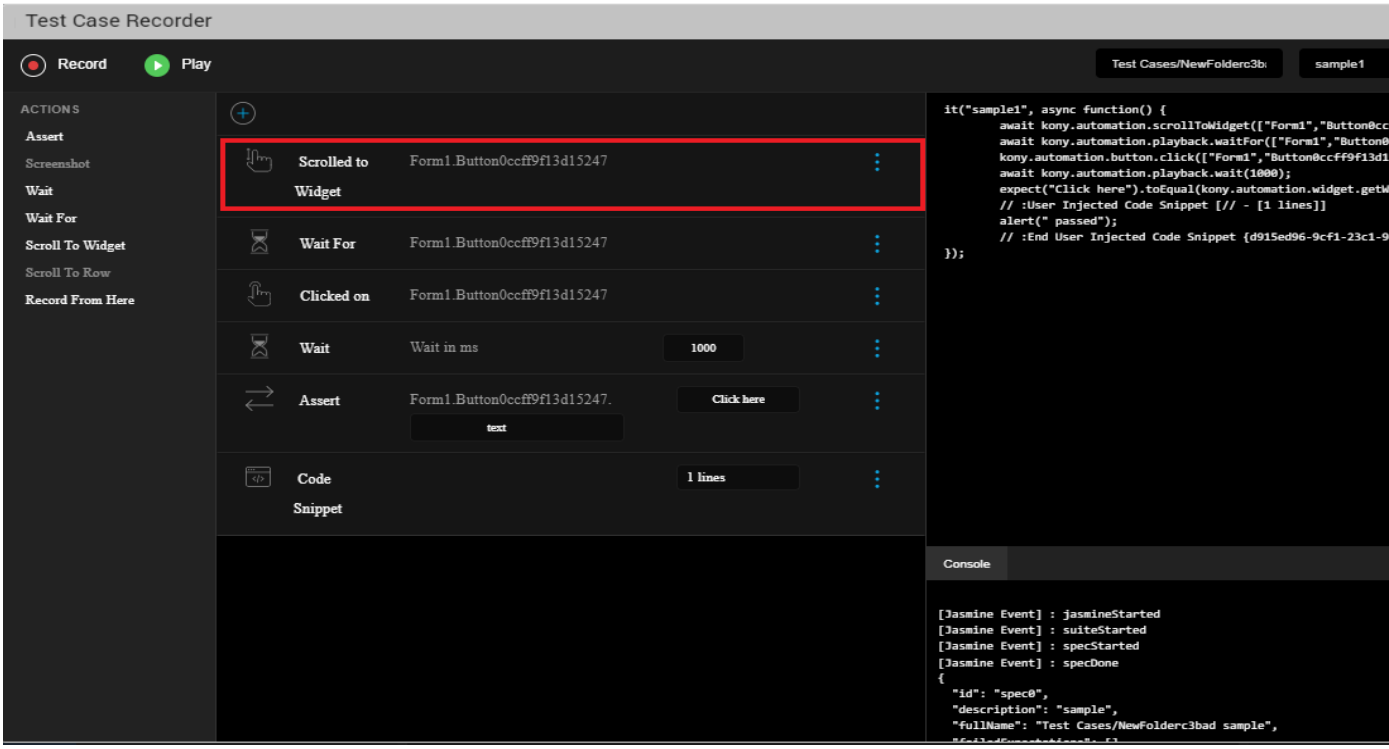
You can edit the test case to add certain actions without having to record from the device. For every activity that is recorded such as a button click or entering text, you can add several other actions before or after the activity in the Test Recorder. You can edit a test case in two ways: opening in Test Recorder, Editing manually.

Edit a Test Case Using Test Recorder

Using the Test Recorder, you can edit a test case while creating the test case or after the test case is created.

To edit a recorded test case, follow these steps:

1. From the Project explorer, navigate to **Test Resources** > **Jasmine** > **<channel>** > **Test Cases**.
2. Right-click a test case and click **Open in Test Recorder**.
   The **Test Case Recorder** window appears.
3. Select any user activity that is recorded.



4. From the **Actions** panel, select any of the following actions for the corresponding user activity.
   - **Assert:** Validates the value of a specified property for any widget.
   - **Screenshot:** Takes a screenshot of the app on the device. This action is applicable on native applications only and not on Desktop web applications.
   - **Wait:** Delays the next action for a specified period after the selected action is performed. The time period is in milliseconds.
   - **Wait For:** This is applied to the widget involved in the selected user activity. Delays the selected action until the widget being awaited appears.

- **Scroll To Widget:** Scrolls to a specified widget on the application.
- **Scroll To Row:** Scrolls to a specified row of the Segment widget.
- **Record From Here:** Starts the recording from the selected action in the recording.

5. To add a code snippet or introduce delay before the selected action, click the add icon and select any of the following:
   - **Insert Code Snippet:** Enter the code and description and click **Save**.
   - **Wait:** Delays the selected action by specified period of time.
6. On top-right corner of the window, click **Save**.
   A new <testSuite>_<testcase>.js file will be created for the respective channel.

You have successfully edited the test case. To run the edited recording, click **Play**.

Edit a Test Case Manually

Further to the ability to change the test case using the Test Case Recorder window, you can also edit the recorded test scripts manually through code. Using the edit option, you can also modify the test scripts that are written manually.

To edit a test case manually, follow these steps:

1. In Visualizer, from the **Project** explorer, navigate to **Test Resources** > **Jasmine** > **<channel>** > **Test Cases**.
2. Right-click the test case that you want to edit manually.
3. Select **Edit Test Case**.
   The corresponding js file opens in the Visualizer canvas.

> ***Important:*** Once you open a test case in this manner, for subsequent changes, you will not find this option. You can double-click the file directly from the Project explorer to open the file and make changes.

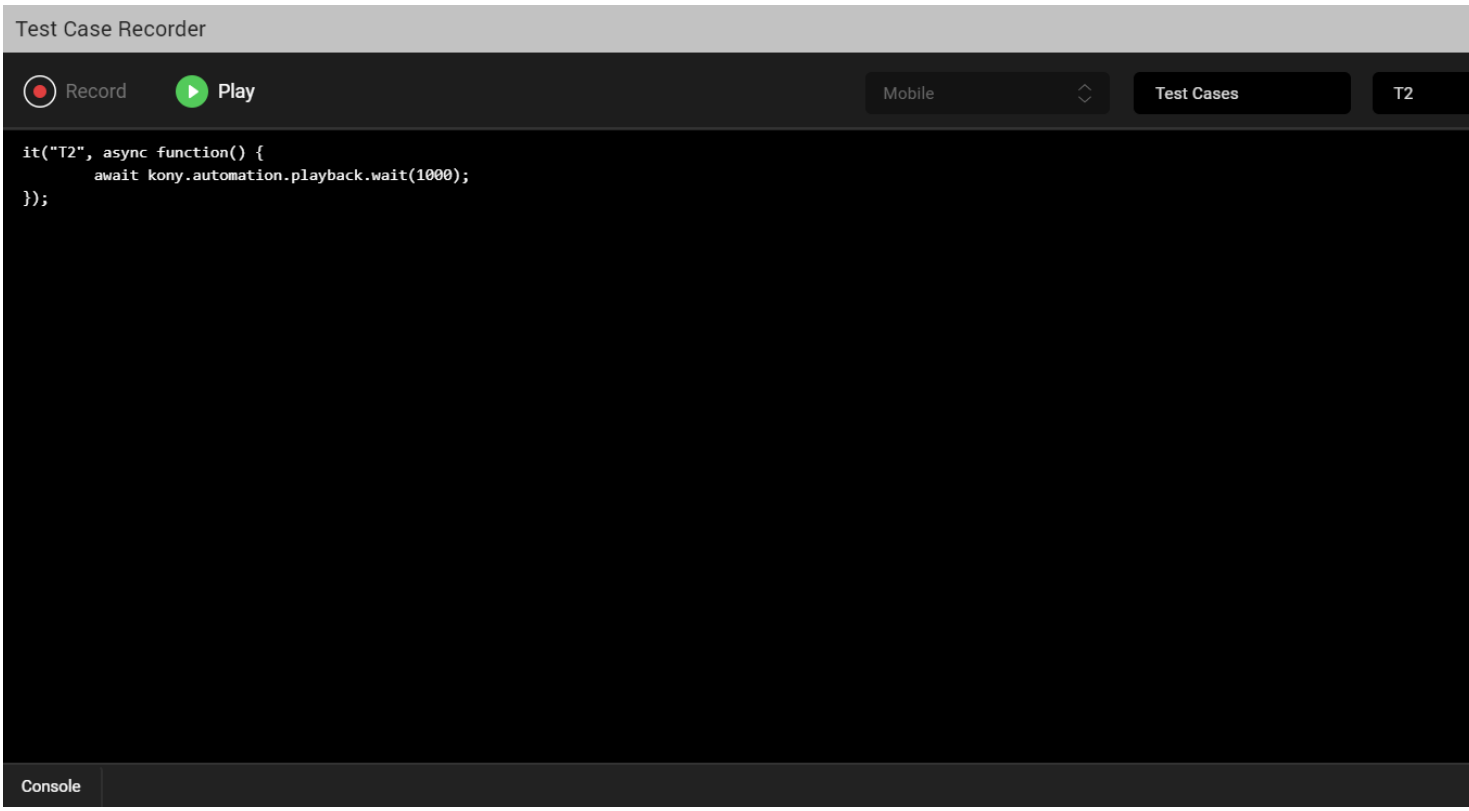
> ***Note:*** If you open a js file by double-clicking it prior to using the **Edit Test Case** option, the file will open in the read-only mode.

4. Make the required changes to the js file.
5. Save the js file.
   The changes are saved.

> ***Note:*** The read-only attribute of the test script is disabled.

You have successfully edited the test case. Once you edit a recorded test case, you can play the test script in the Test Case Recorder and view the test result.

To play the test case, use the **Open in Test Recorder** option for the same test case. The following **Test Case Recorder** window appears.



From the **Test Case Recorder** window, you can click **Play** to run the test case that has been edited manually.

You can also rename the test case, if required. The **Record** option is disabled by default.

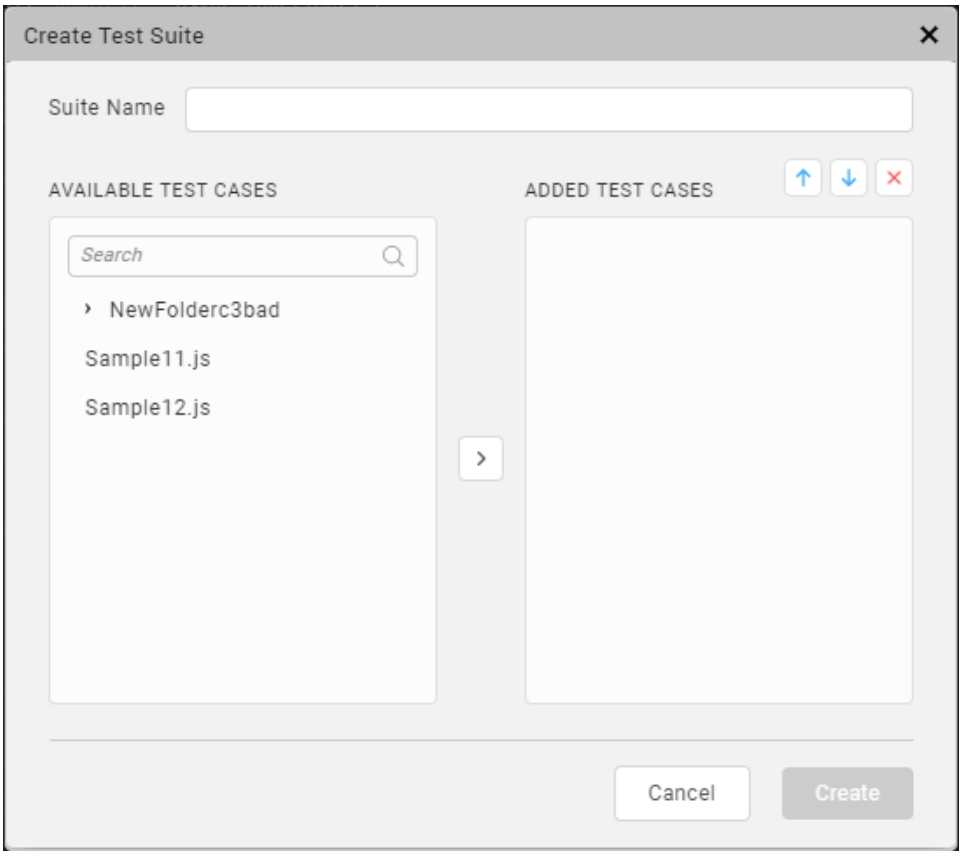
Test Suite

Test Suite is a collection of several test cases. Create a test suite to group various test cases and run them in an order in which they are grouped.

Create a Test Suite

Follow these steps to create a test suite:

1. From the Project explorer, go to **Test Resources** > **Jasmine**.
2. Expand any <channel> and go to **Test Suites**.
3. Right-click on **Test Suites** and select **Create a Test Suite**. The **Create Test Suite** window appears.



4. From the available test cases, select the test cases that you want to add to the test suite.
5. Drag and drop the selected test cases in the ADDED TEST CASES pane.
   By using the **up** and **down** icons, you can change the sequence to the order in which you want the test cases to run.
   If you want to remove a test case from the test suite, select the test case and click the **delete** icon.
6. Type a name in the **Suite Name** box and click **Create**.
   A new <testSuite.js> file is created in the **Test Suites** folder.

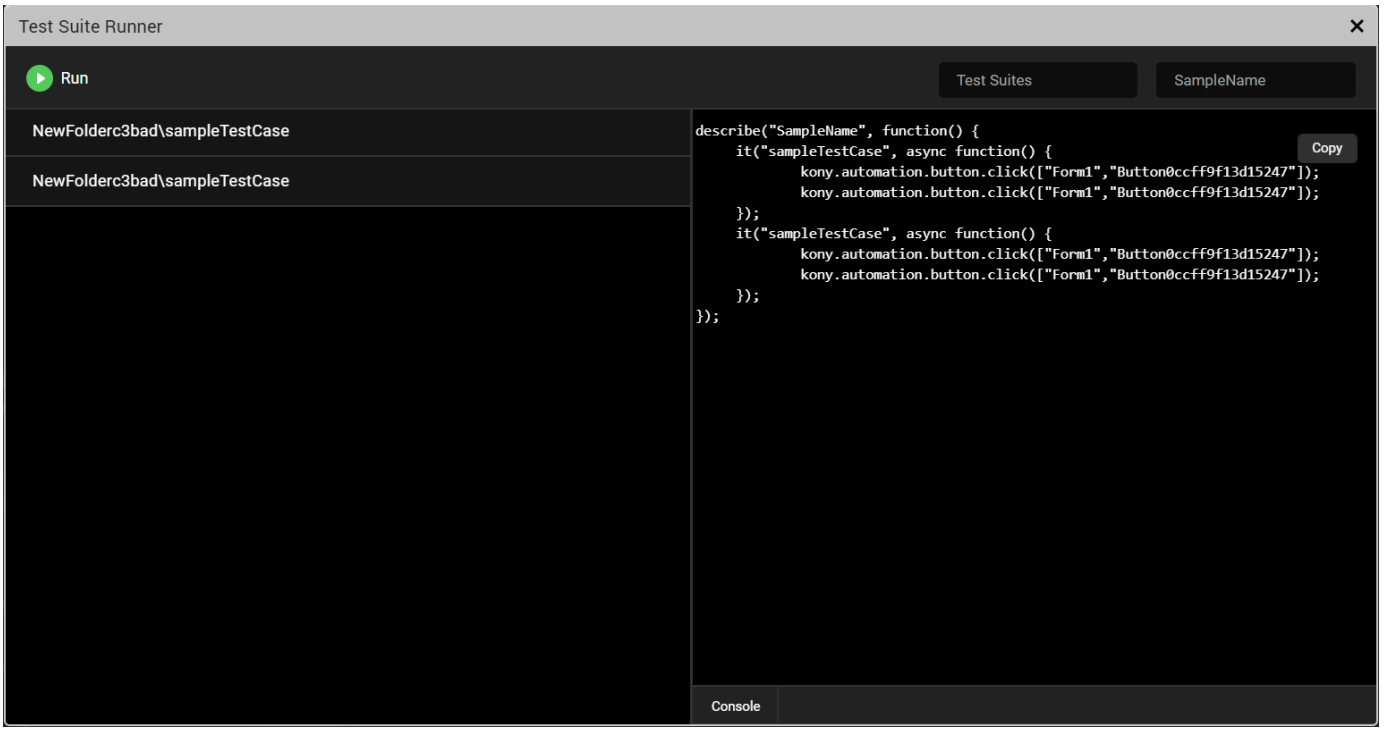
You have successfully created a test suite. To edit the test suite, navigate to the Project explorer, right-click on the test suite and select **Edit Test Suite**. Once the modifications are done, click **Save**.

Run a Test Suite

After creating a test suite, run the test suite to play the all the test cases and test the application.

Follow these steps to run a test suite:

1. From the Project explorer, navigate to **Test Resources** > **Jasmine**.
2. Expand any <channel> and go to **Test Suites**.
3. Select a <testSuite.js> file.
4. Right-click on the file and select **Run**.
   The **Kony Test Suite Runner** window appears.



The left pane of the window displays all the test cases in the specified test suite.
The right pane of the window displays the code in the order of the test cases. On selecting a test case from the left pane, the corresponding code on the right pane is highlighted.
The **Console** tab displays the results of the test.

Note:

- When you minimize the **Test Suite Runner** window, a **View** option appears in the **Test** tab on the lower pane of Visualizer. You can click **View** to re-open the window that you have minimized.
- When you minimize the **Test Suite Runner** window while running a test suite, a progress bar appears in the **Test** tab indicating the progress of test suite that is running. Once the test suite is run completely, a toast message appears on the lower-right of Visualizer.
- After minimizing the **Test Suite Runner** window, a few options such as **Run**, **Edit**, **Delete**, and **Rename** are disabled for the test files. Also, the options to create a new Test Case, Test Suite, or Test Plan are disabled.

Note: If you want to play all the test cases in the test suite, click **Run** on the upper-left corner of the **Test Suite Runner** window.

Test Plan

Test plan is a collection of several test suites. You can create a test plan to play various test suites that were created.

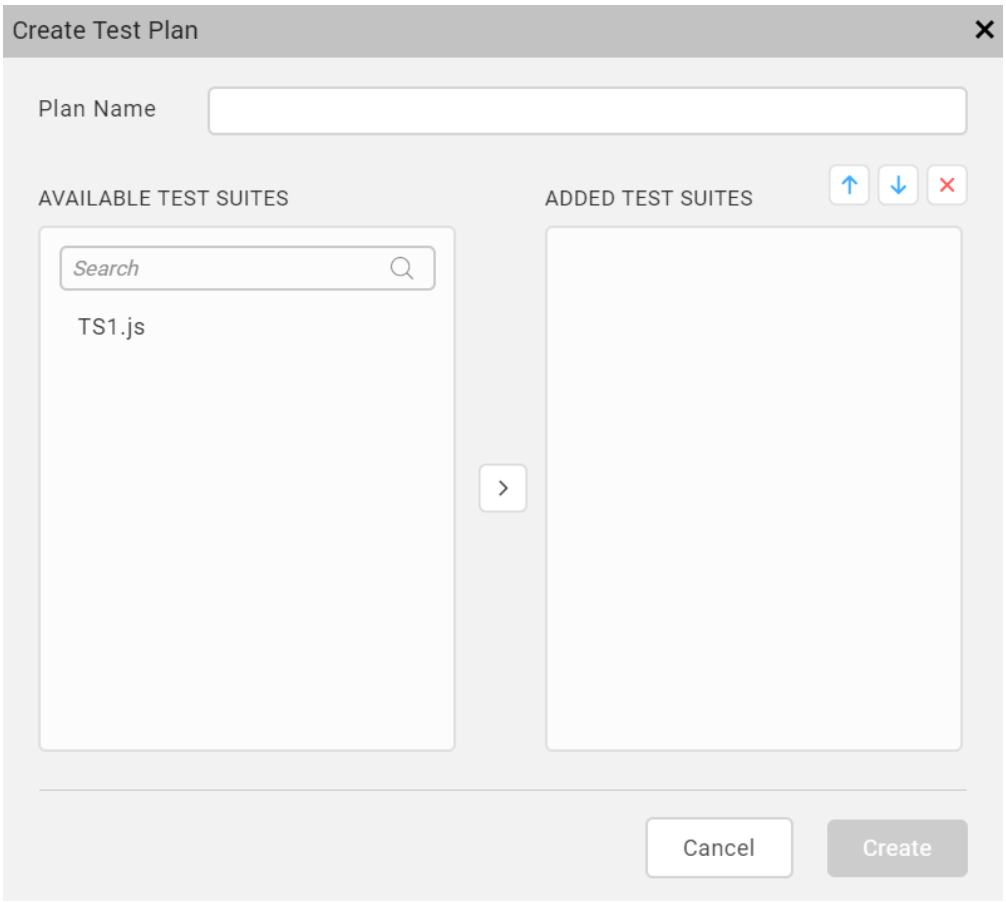
Create a Test Plan

You can create a new test plan from the Project explorer. Once a test pan is created, a corresponding JavaScript file is created.

To create a new test plan, follow these steps:

1. From the Project Explorer, go to **Test Resources** > **Jasmine**.
2. Expand the channel where you created the test cases and test suites earlier.

3. Right-click on **Test Plans** and click **Create a Test Plan**.
   The **Create Test Plan** window appears.



4. From the available test suites, select the test suites that you want to add to the test plan.

5. Drag and drop the selected test suites under ADDED TEST SUITES.
   By using the **up** and **down** arrow icons, you can change the sequence to the order in which you want the test suites to run.
   If you want to remove a test suite from the test plan, select the test suite and click the **delete** icon.

6. Type a name in the **Plan Name** box and click **Create**.
   The <PlanName.js> file is created in the **Test Plans** folder.

7. If you want to edit the test plan that you have created, right-click on the file and select **Edit Test Plan**.
   Click **Save** after the modifications are made.

Writing the testPlan.js file

To enable testing in Visualizer, a default testPlan.js file is present in theTest Plan folder at **Project > Test Resources > Mobile > Test Plans > testPlan.js**. By using this file, you can run a test plan that you have created or run selected test suites directly. The `testPlan.js` file contains the following code:

```
//This is the entry point for automation. You can either:
//1.Require any one of the created test plans like this:
require([/*<Test Plans/ file>*/]);

// or
//2. Require the test suites along with executing jasmine as below
//Nested require for test suites will ensure the order of test suite
exectuion
require([/*<Test Suites/test suite1>*/],function(){
  require([/*<Test Suites/test suite2>*/], function(){
    //and so on
       require([/*<Test Suites/last test suite>*/], function(){
                      jasmine.getEnv().execute();
        });
  });
});

//Since this is file is to be manually edited, make sure to update
//any changes (rename/delete) to the test suites/plans.
```

You can modify the testPlan.js file in the following ways:

- Invoke the require function for a test plan
- Invoke the require function for selected test suites

Invoke the require function for a test plan

While testing an application, you can create multiple test plans, but you can run only one test plan at a time. To run a specific test plan you must invoke the **require** function for the test plan.

Follow these steps to execute a test plan:

1. From the Project Explorer, navigate to **Test Resources** > **Jasmine**> **<Channel>** > **Test Plans**
2. From the **Test Plans** folder, open the `TestPlan.js` file.
3. In the `TestPlan.js` file, modify the default code in the following format.

```
require([<Test Plans/ file>]);// Write the name of a test plan
that you want to execute.
```

4. Save the file.
   When the test plan is run next time, the plan defied by you in the previous step will be executed.

Invoke the require function for selected test suites

By modifying the testPlan.js file, you can also run a group of test suites, without creating a test plan explicitly. To run a group of test suites, you must invoke the **require** function for each test suite.

Follow these steps to invoke the require function for selected test suites:

1. From the Project Explorer, navigate to **Test Resources** > **Jasmine**> **<Channel>** > **Test Plans**
2. From the **Test Plans** folder, open the `TestPlan.js` file.
3. In the `TestPlan.js` file, modify the default code in the following format.

```
// Consider a test plan with three test suites: test suite1, test
suite2, and test suite3
//Nested require for test suites will ensure the order of test
```

```
suite exectuion
require(["Test Suites/<test suite1>"],function(){
  require(["Test Suites/<test suite2>"], function(){
       require(["<Test Suites/<test suite3>"], function(){
                       jasmine.getEnv().execute();
        });
  });
});
```

4. Save the file.
   When the test plan is run next time, the test suites defined by you in the previous step will be executed.

Dashboard

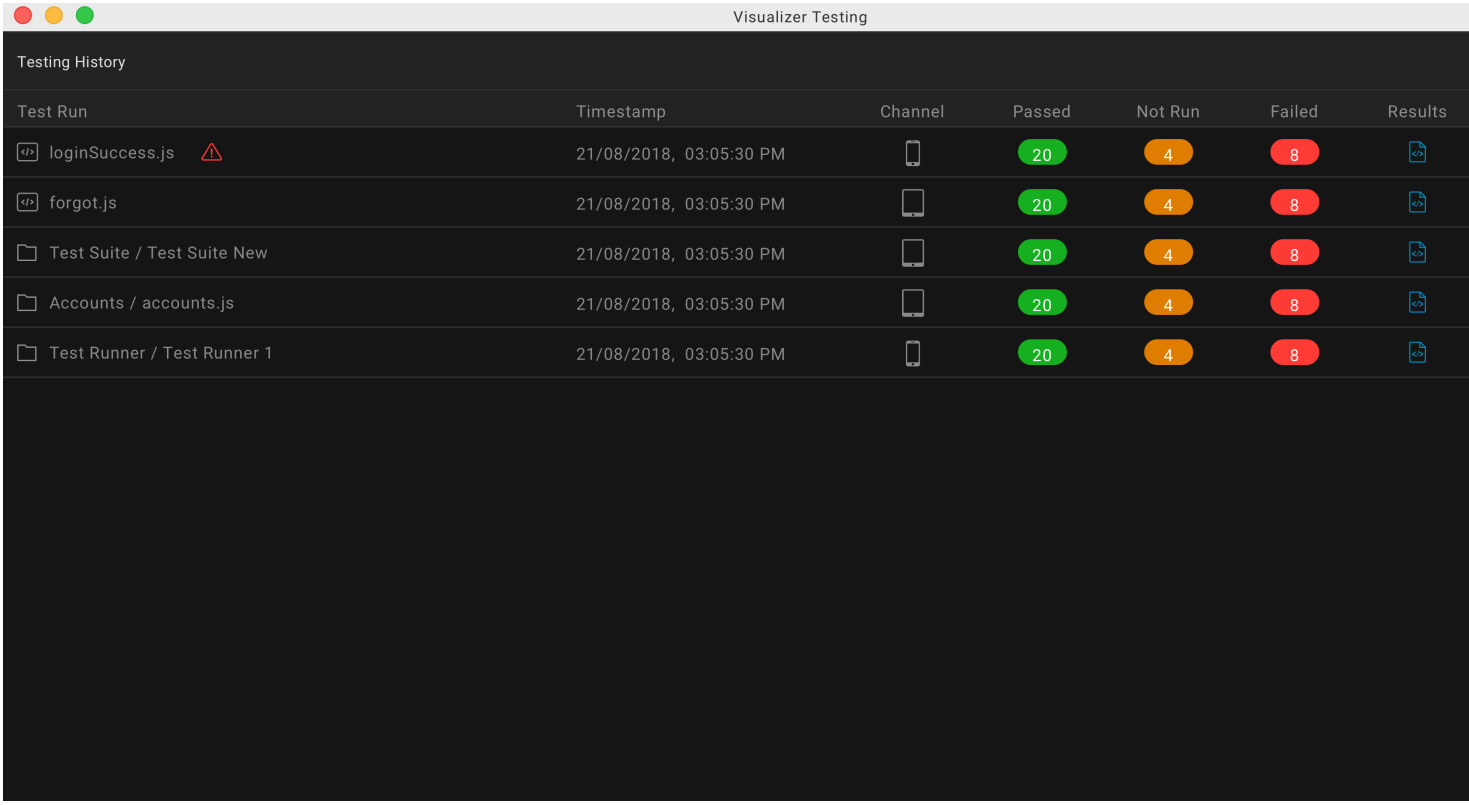
Kony Automator is a one-stop shop where you can view the latest executed test cases. You can find more details about the time of execution of a test, channel on which a test is executed, and the test results.

You can view the last five tests executed on the Dashboard and the recorded test scripts, if you want. In addition, you can record a new test case from the Dashboard without having to navigate to the **Test Resources** folder in the **Project** explorer.

Using Dashboard, you can perform the following tasks:

- View the testing history
- View the recordings of the executed test scripts
- Create a new test case

You can find the **Kony Automator** icon on the left navigation bar of Visualizer. Click the icon to open the **Dashboard** window.



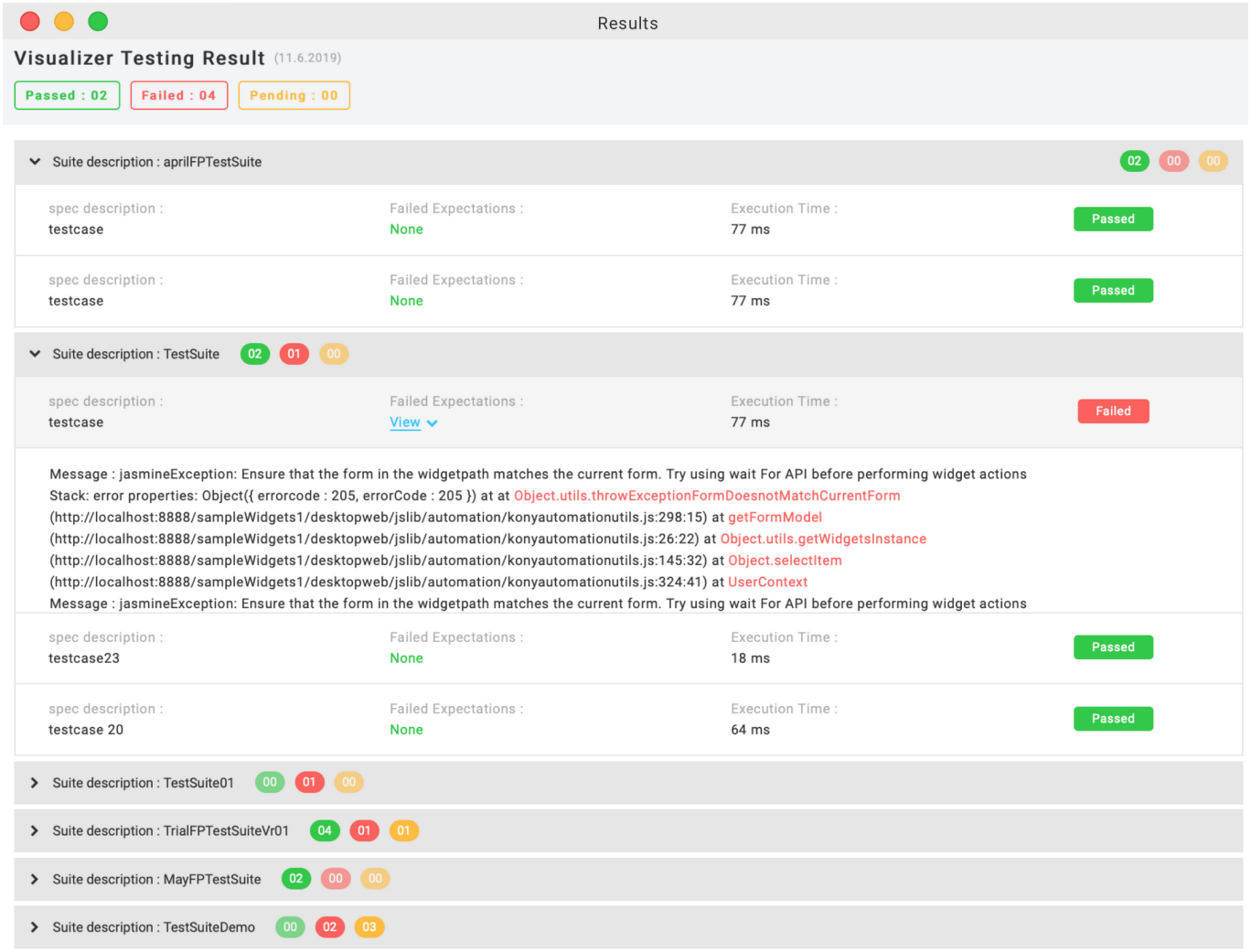
View the Testing History

The Dashboard displays the information about the last five test results of the application. The latest tests that have executed can be test cases or test suites.

The Dashboard window displays the following details for each of the five tests that have been executed:

- **Time stamp:** The time at which the test was executed.
- **Channel:** The channel on which you ran a test case or test suite.
- **Passed:** The number of the test cases that ran successfully.
- **Not Run:** The number of ignored test cases that the user do not want to run.
- **Failed:** The number of test cases that failed due to errors.

- **Results:** Click on the file to view the test results of each test suite.



The number of passed test cases, failed test cases, and pending test cases are displayed across each test suite. Furthermore, you can expand the test result of each test suite to view more information such as:

- **Spec description:** Displays the name of the test case.
- **Failed exceptions:** Displays the location of errors in the test case along with the file location.
- **Execution Time:** Displays the time taken by a test case for completing the execution.

View the Recording of an Exectuted Test Script

In the Dashboard window, a play icon is displayed for each of the latest five tests that have been executed.

You can click the play icon to run the recorded test script.

Create a New Test Case

Using Dashboard, you can start recording a new test script directly from the Dashboard window, without having to create a test case from the Project explorer.

To create a new test case from the **Dashboard** window, follow these steps:

1. From the main menu, click **Build** > **Live Preview Settings**.
2. In the **Platform and Channels** section, select the appropriate checkboxes.

   For the **Preview** Mode, select **Test**.
3. Click **Save and Run**.

   The app launches on the device. In case of Inline preview, the **Visualizer Preview** window appears.
4. From the left navigation bar of Visualizer, click the **Kony Automator** icon.
   The **Dashboard** window appears.
5. In the upper-right corner of the window, click **Start New**.
   The **Test Case Recorder** window appears.
6. From the **Visualizer Preview** window, re-launch the application.
   The device connects successfully.

7. On the upper right corner of the **Test Case Recorder** window, select the channel for which you want to create the test case.
8. On the upper-left corner of the **Test Case Recorder** window, select **Record** to start recording a test script.
   After the recording starts, a **stop** option appears.
9. To discontinue the recording, click **Stop**.
10. On the upper-right corner of the window, type a name for the test case and click **Save**.
11. To play the recorded script, re-launch the app and click **Play**.

    The recorded script plays in the **Visualizer Preview** window.
    You can view the result of the test case in the **Console** tab.

Deploy the Test Resources

A `testPlan.js` file is present for each channel and is executed when the app is launched without opening the test recorder. After the test scripts are generated, deploy the test resources to retrieve the test scripts from the server and test the application.

To deploy your application, follow these steps:

1. In Visualizer, from the Project explorer, go to **Test Resources** > **Jasmine**.
2. Right-click on **Jasmine** and select **Deploy**.

For more information on queries about Jasmine Test automation, refer FAQs.

Preview an App on a Device

The procedure for previewing an app differs between Kony Visualizer and Kony Visualizer Classic. Click the procedure that applies to the edition of Kony Visualizer that you are using.

Preview a Kony Visualizer App on a Device

Preview a Kony Visualizer Classic App on a Device

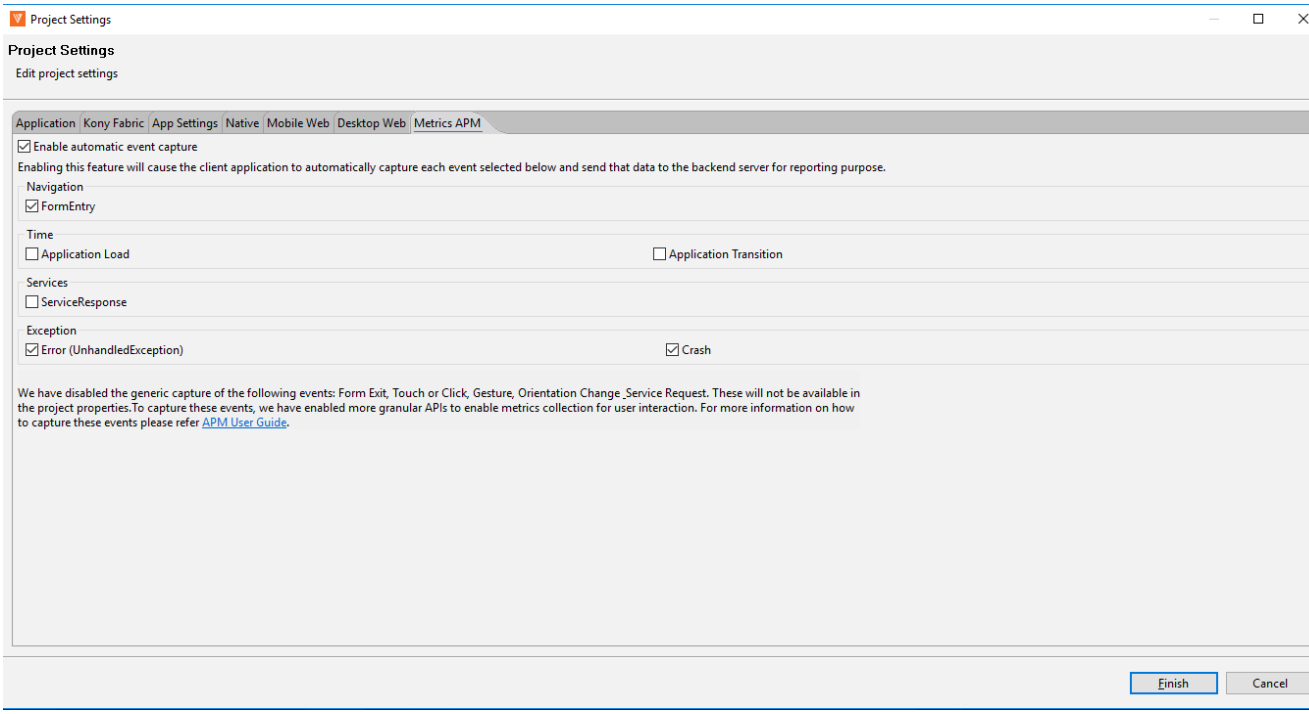
Monitor an App's Performance

With Kony Visualizer, you can conduct application performance monitoring (APM) so that you can evaluate the performance of your app and use the feedback as the basis for improving its performance. This topic only covers how to activate APM in Kony Visualizer. Capturing events and tracing the user journey through an app is handled by Kony Fabric. For more information on customizing and using APM, refer Kony Fabric Reporting and Analytics.

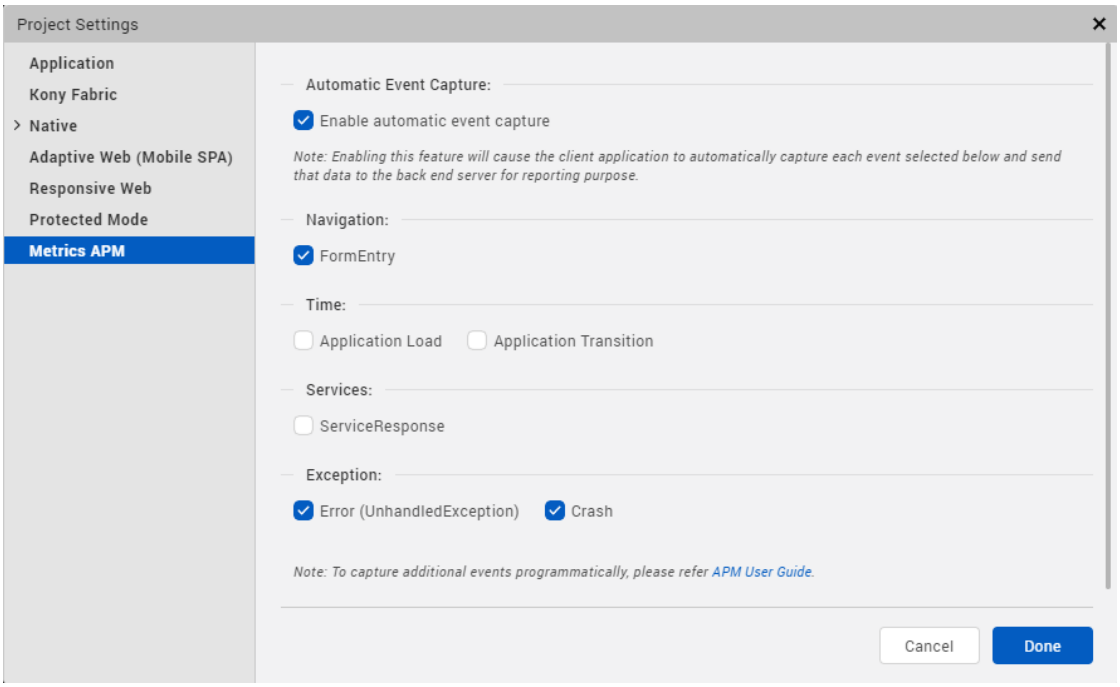
> ***Note:*** From Kony Visualizer V8 SP4 Fixpack 20 onwards, APM events are supported in Kony Visualizer. However, APM is not supported for the Run Live Preview and Publish Live Preview features in Kony Visualizer.

To activate APM, do the following:

1. In Kony Visualizer, open the Metrics APM tab. To do so, follow any of these steps:
    - In Kony Visualizer Classic: On the **Project** tab, click **Project Settings**. The Project Settings window appears. Click the **Metrics APM** tab.



- In Kony Visualizer: From the **Project** menu, click **Settings**. The Project Settings window appears. Click the **Metrics APM** tab.



2. Activate APM by selecting the **Enable automatic event capture** check box.
3. Configure the extent of the APM activity by setting the following types of events.
    - **Navigation.** To capture when the user enters a form, check **FormEntry**.To capture when the user exits a form, check **FormExit**.
    - **User Interaction.** To capture when the user touches the screen or clicks a button, check **Touch or Click**. To capture when the user changes the orientation of the device, check **Orientation Change**.To capture when the user uses a gesture, such as a swipe, check **Gesture**.
    - **Services.** To capture when the app initiates a service request, check **ServiceRequest**. To capture when the app receives a response from a service, check **ServiceResponse**.
    - **Exceptions.** To capture when an unhandled error exception takes place, check **Error**. To capture when the app crashes, check **Crash**.
4. Click **Finish**.

View an Application on an Emulator

After you create an application and build it, you can view it on a device emulator that you have set up using the resources of a given platform's SDK.

To view the application on an emulator, do the following:

1. Configure an emulator for the **platform**[1] and **channel**[2] that you want to view the app on.
2. Build the application.
3. On the **Product** menu, point to **Launch Emulator**, and then select the emulator that you want to run.
4. The emulator opens.
    - On an Android emulator, the application opens automatically.

Generate IPA for a Native iOS Application

In addition to creating iPhone and iPad applications directly from source code, you can use Kony Visualizer to build an iPhone or iPad application and deploy it to a Mac automatically. You need a dedicated Mac to enable creation of iPhone and iPad application binaries.

> ***Important:*** Currently, Kony Visualizer supports Xcode 9.0 and later.

Prerequisites

Before you begin, ensure that the following conditions are met:

- Xcode and command line tools are installed on the Mac.
- Xcode path is set to `/Applications/Xcode.app/Contents/Developer/`.

---

[1]The operating system of a given device. iOS and Android are examples of platforms.
[2]Device types available within a given platform. These include mobile (i.e. phone), tablet, and desktop.

To check the Xcode path or modify the path:

1. Open the **Terminal**.

   ```
   xcode-select --print-path
   ```

   Output must be: `/Applications/Xcode.app/Contents/Developer/`

2. If the path is not set, enter the following command:

   ```
   sudo xcode-select --switch
   /Applications/Xcode.app/Contents/Developer/
   ```

- You must have a valid Apple developer account, and you must be signed in to the Accounts section of XCode Preferences.
- You must have obtained a 10-digit development team identifier from Apple. The Team ID can be found in the Membership Details of your Apple Developer account.

IPA Generation

Building an iPhone or iPad application and deploying it to a Mac involves generating an archive file, and then exporting the archive file to an IPA file based on the specified distribution method.

To configure Kony Visualizer to build an iOS native application, follow these steps:

1. From the **Project** menu of Kony Visualizer, click **Settings**.
   The **Project Settings** window appears.
2. From the navigation pane on the left, expand **Native**, and click **iPhone/iPad.**
3. Under the **Certificates** section, in the **Build Local** settings, specify values for the following fields:
   - **Development Method**: The application's distribution method. The method can be one of the following:

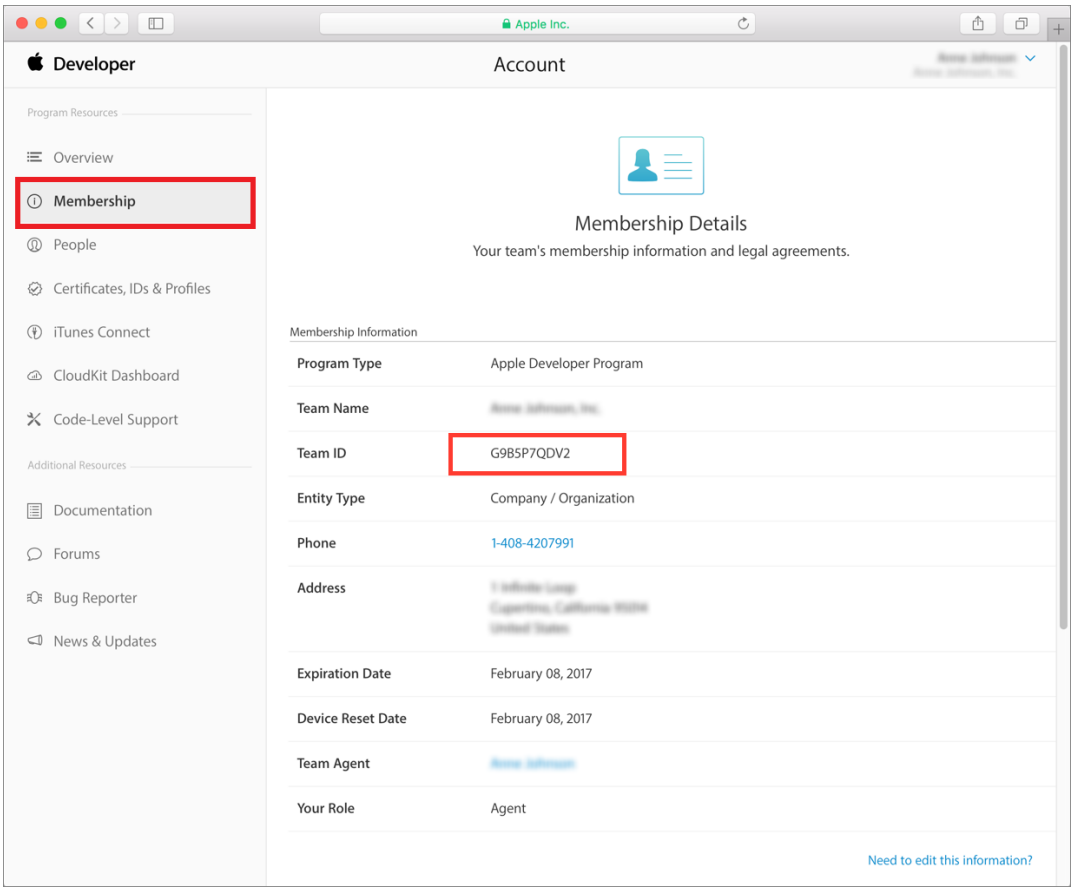
- **development**: Signs and packages the application for development distribution outside the iOS App Store.
- **app-store**: Signs and packages the application for distribution in the iOS App Store.
- **ad-hoc**: Signs and packages the application for ad-hoc distribution outside the iOS App Store.
- **enterprise**: Signs and packages the application for enterprise distribution outside the iOS App Store.

An archive file is always generated initially using development code signing. Xcode then automatically generates the required development provisioning profiles. For development code signing, you must have a valid development certificate installed in your keychain.

When exporting the archive to an IPA using the *development* option, Xcode reuses the development certificate and provisioning profile that are used for archive code signing. It is unnecessary to create profiles from the development center and install them on your system.

For *app-store*, *ad-hoc*, and *enterprise* options, make sure that you have a valid developer certificate, distribution certificate, and distribution profile installed on your system. Xcode will not automatically generate these distribution profiles.

- **Development Team**: Your 10-digit development team identifier obtained from Apple. The Team ID can be found in the Membership Details of your Apple Developer account.



- **Keychain Password**: The Keychain password for your application. On the iPhone, Keychain rights depend on the provisioning profile used to sign your application. Ensure that you consistently use the same provisioning profile across different versions of your application.

4. Once you have configured all the required details, click **Done**.

   You can now build your application for iPhone or iPad which generates the IPA file.

> ***Note:*** All errors, warnings, and confirmations can be seen on the Console.

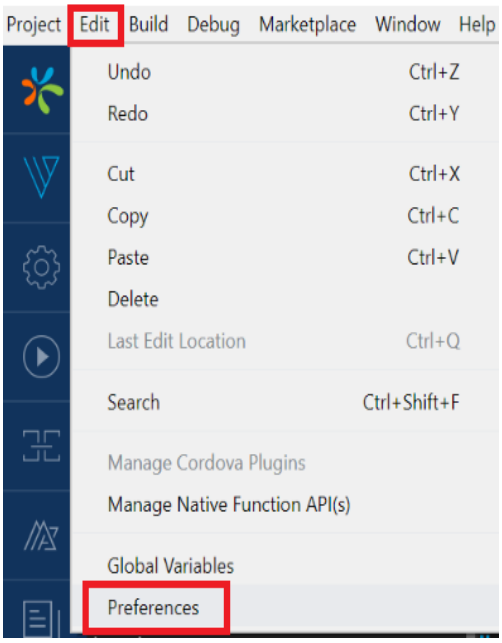
Create a remote connection to a Mac device

While making use of the **Build Native Locally** option to create an iOS native application, if you are using a Windows machine, you can configure Kony Visualizer to make a remote connection to a Mac machine.

Before you configure the **Mac Details**, ensure that you have followed the steps mentioned at IPA Generation.

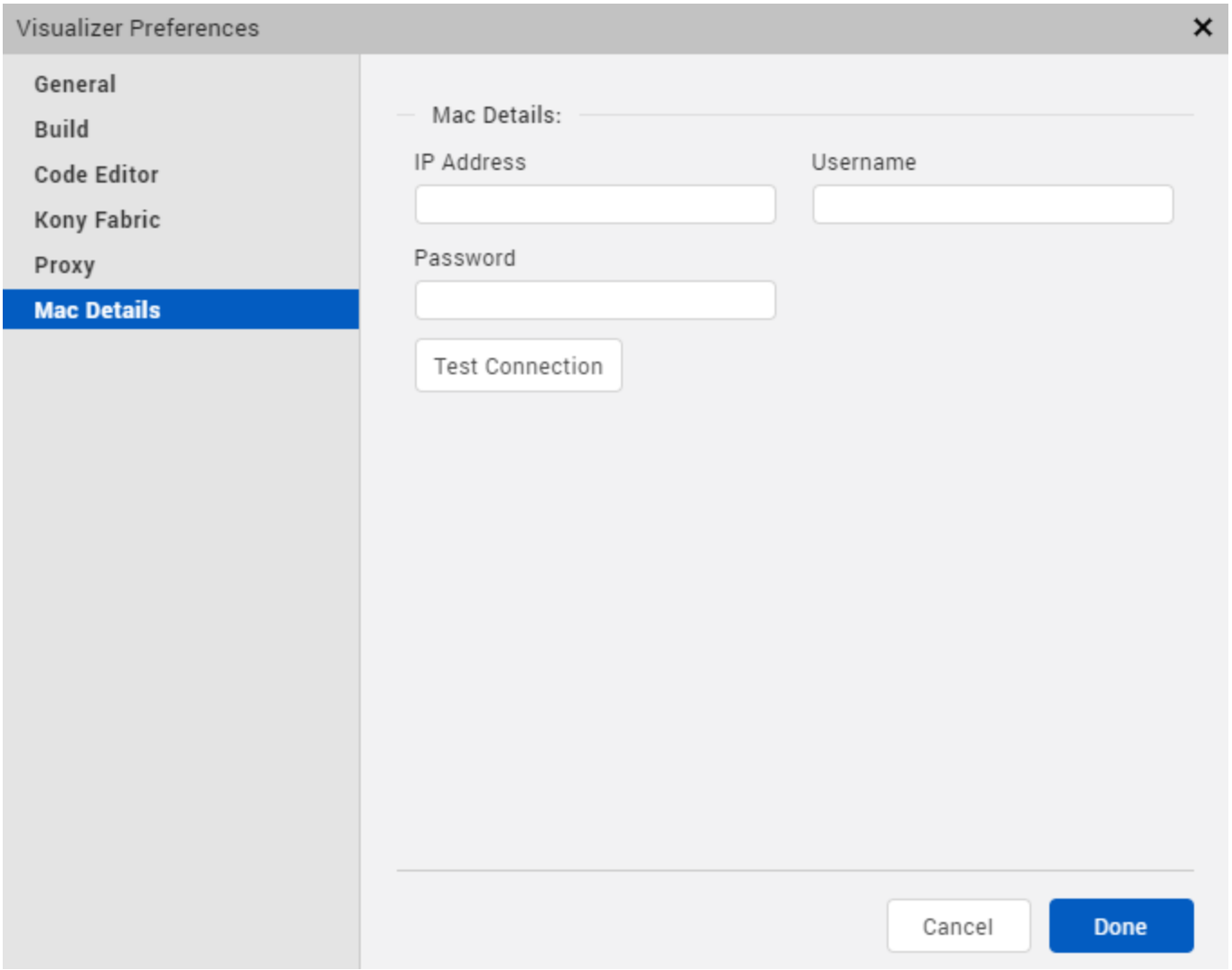
To configure Kony Visualizer to make a remote connection to a Mac machine, follow these steps:

1. In Kony Visualizer, click **Edit** > **Preferences**. The **Visualizer Preferences** window appears.



2. Click the **Mac Details** tab.
3. Under the Mac Details section, enter the following details:
    - **IP Address**: The IP address or domain name of the Mac system in which you want to remotely build and view your application.

- **Username**: The user name of the Mac system you want to remotely access to build your application.
- **Password**: The password of the Mac system you want to remotely access to build your application.



4. Click **Test Connection** to validate the connection to the Mac device.
5. Click **Done**.

Open Webapps and Build Folders

Applies to *Kony Visualizer Classic*.

To locate your Webapps and build resources, you can open their respective folders for any of the channels.

To open a Webapps or build folder for a channel, do the following.

1. In the Project Explorer, click the context menu arrow of any of the available channels, namely Mobile, Tablet, Desktop, or Watch.
2. From the context menu, click one of the available options:
    - Open Build Folder
    - Webapps Native Folder
    - Webapps Mobile Web Folder

Debug an Application

Applies to *Kony Visualizer Classic*.

Kony Visualizer provides you with a debugger to detect and diagnose errors in your applications.

From **V8 SP4** onwards, you can debug apps in Kony Visualizer.

The debugger allows you to control the execution of your application by:

- Setting breakpoints.
- Suspending launched applications.
- Stepping through your code.
- Examining the contents of the variables and view them on an emulator.

The debugger has a client and server design. This design allows you to debug applications that are running remotely on the network as well as the applications running locally on your workstation.

Debugging an application involves building the application in debug mode (iOS or Android) and launching the application in an iOS or Android emulator or device. You can then use the Google Chrome debugger to debug the application. For information on using the Chrome debugger, see Get Started with Debugging JavaScript in Chrome DevTools on the Google Chrome website.

See the following for information on debugging an application:

- Build and Launch an iOS Application
- Build and Launch an Android Application
- Debug JavaScript for iOS in Kony Visualizer
- Debug JavaScript for Android in Kony Visualizer

Build and Launch an iOS Application

You must build and debug an iOS app in the Mac environment.

To build and launch an iOS application for debugging, do the following:

1. In Kony Visualizer, build the application in debug mode. To do so:
    - In Kony Visualizer Classic, from the **Product** menu , click **Build**. In the Build dialog box, select Debug from the Build Mode drop-down list.
    - In Kony Visualizer, from the **Build** menu, click **Build and Publish Native**. In the Build dialog box, select Debug from the Build Mode drop-down list.
2. After the build is complete, extract the built *.kar* file.
3. Open an Xcode project.
4. Select the target as **Debug**, choose an emulator, and click **Run**. The emulator launches and the following dialog appears:

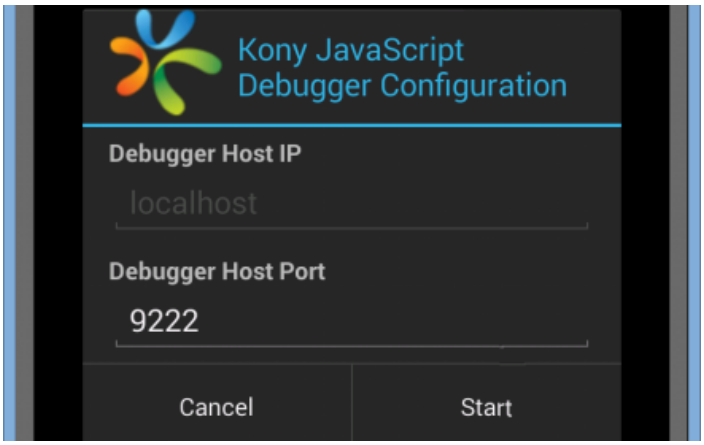
5. Select **Start** in the emulator. The waiting for debugger dialog appears.

6. Follow the steps in Debug JavaScript for iOS in Kony Visualizer.

Build and Launch an Android Application

To build and launch an Android application for debugging, do the following:

1. In Kony Visualizer, build the application in debug mode. To do so:
   - For Kony Visualizer Classic, from the **Product** menu , click **Build**. In the Build dialog box, select Debug from the Build Mode drop-down list.
   - In Kony Visualizer, from the **Build** menu, click **Build and Publish Native**. In the Build dialog box, select Debug from the Build Mode drop-down list.
2. Launch the application in the emulator or device.
   The device must connected using a USB connection. If you have specified a port other than 9222 as your Android debugger port in:
   - Kony Visualizer Classic: In the **Window** menu, click **Preferences**. From the Preferences dialog box, click **Kony Visualizer** and update the Android Debugger Ports setting to use the same port.
   - Kony Visualizer: In the **Edit** menu, click **Preferences**. From the Preferences dialog box, click **Build** and update the **Debugger ports** to use the same port.



3. Click **Start** in the emulator. A Waiting for the Debugger message appears.
4. Follow the steps in Debug JavaScript for Android in Kony Visualizer.

Debug JavaScript for iOS in Kony Visualizer

Kony Visualizer Classic

With Kony Visualizer, you can perform JavaScript debugging on connected Apple computers and devices. This procedure assumes that you have the Google Chrome web browser installed on your computer. You also need to install and run on your Mac the iOS WebKit Debug Proxy (ios_webkit_debug_proxy). It must be version 1.6 or greater. To install this utility, from your Mac, open a terminal and enter the following command:

```
brew install ios-webkit-debug-proxy
```

Important: You can debug JavaScript applications in iOS using the Kony Visualizer preview app on your device or on the simulator. The option to debug applications in iOS is available from **Preview** > **Debug** in Kony Visualizer Classic.

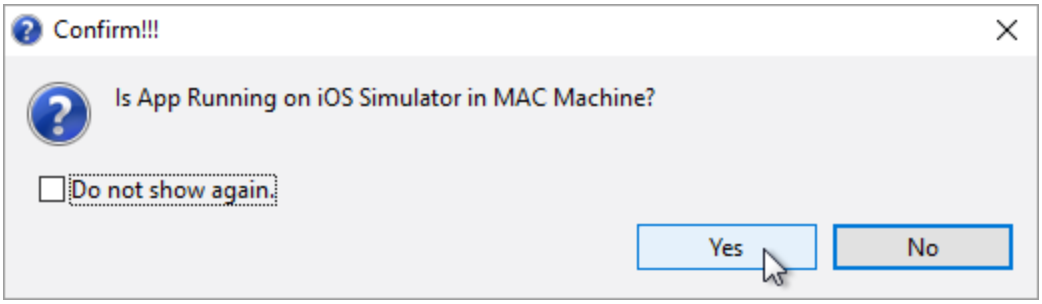
Important: If you are using a Windows machine to connect to a Mac machine for debugging, for iOS applications Functional Preview debugging, a URL will be generated in the console of Kony Visualizer. Ensure that you copy and paste the generated URL in your Chrome browser to start the debugging process.

Important: Ensure that you deactivate and activate the breakpoint once your debugging starts in your Chrome browser. This is applicable for Windows and Mac machines.

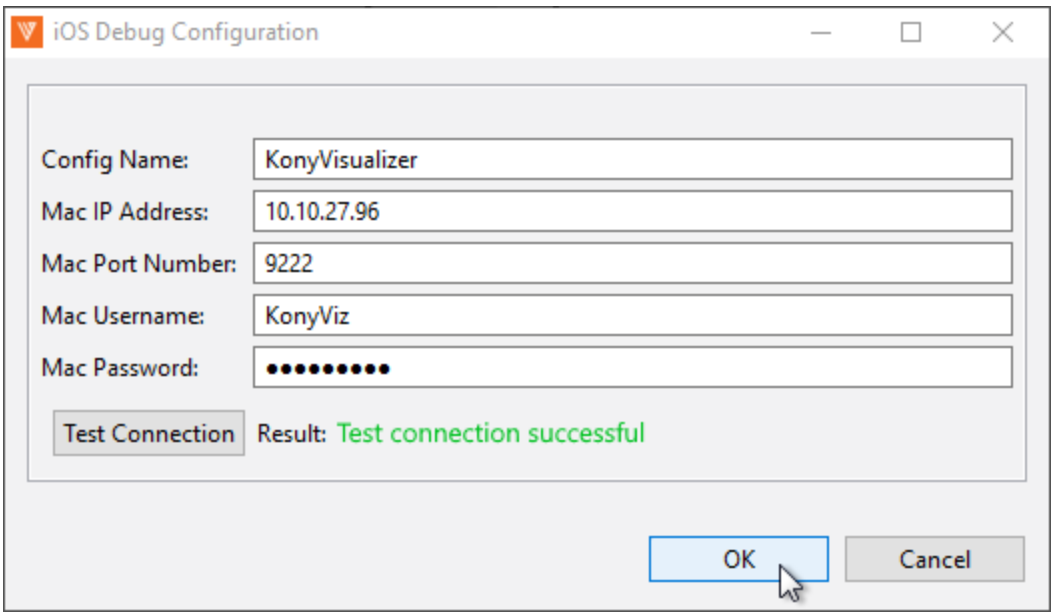
To debug JavaScript applications for iOS on a simulator or device:

1. In Kony Visualizer, build your application for the iOS platform in debug mode. To do so,
    - In Kony Visualizer Classic, from the **Product** menu , click **Build**. In the Build dialog box, select Debug from the Build Mode drop-down list.
    - In Kony Visualizer, from the **Build** menu, click **Build and Publish Native**. In the Build dialog box, select Debug from the Build Mode drop-down list.

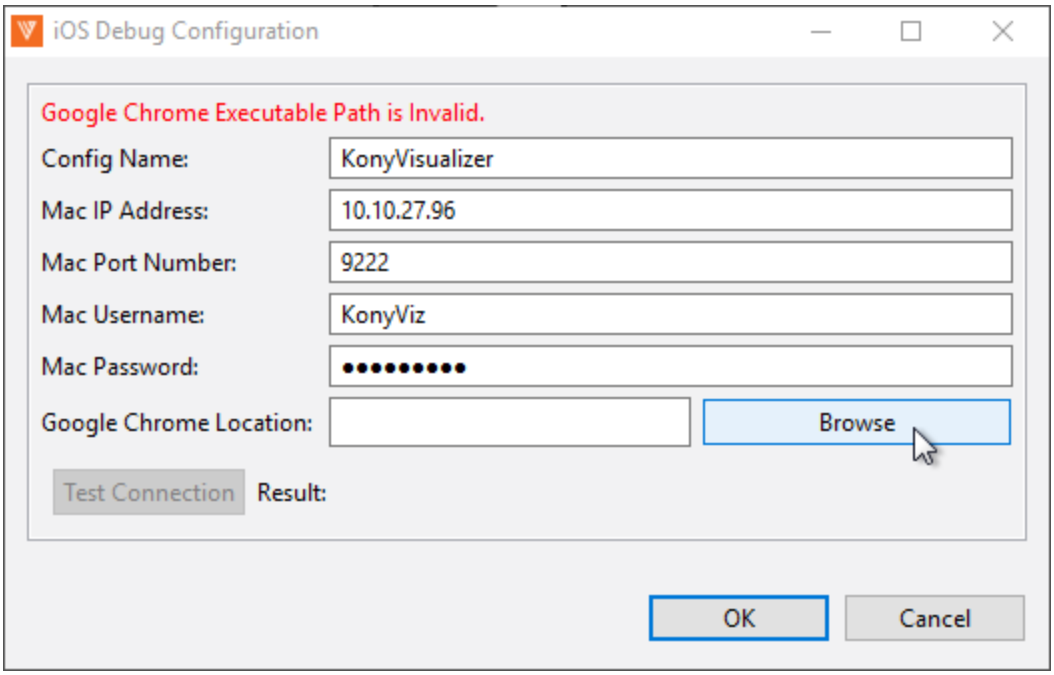
2. Launch the application in a simulator on your Mac computer, or on the device.
3.
   - For Kony Visualizer Classic, from the **Product** menu , click **Debug As**, and then select **Debug iOS application on a simulator** or **Debug iOS application on a device**.
   - For Kony Visualizer, from the **Debug** menu, click **Debug Native App**. From the **Debug Native App** drop-down list, select either **iOS Devices** or **iOS Simulators**.
4. Kony Visualizer displays a dialog box confirming whether the app is running in an iOS simulator on a Mac computer. .



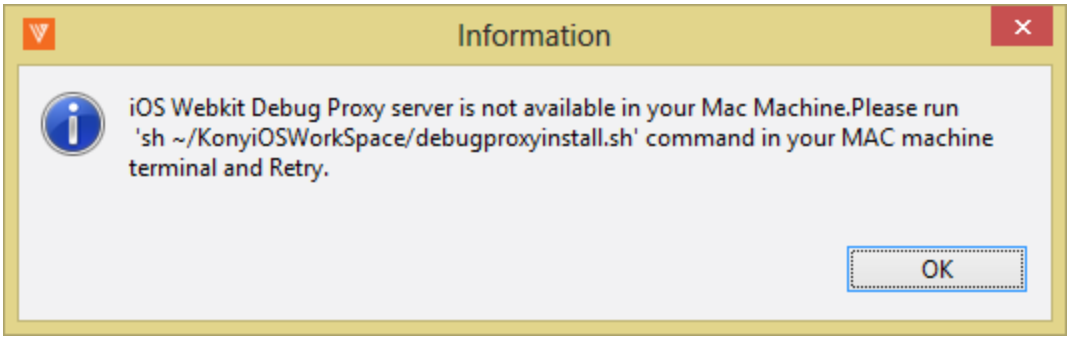
5. If you click **Yes** in the dialog box and you're running Kony Visualizer on a Windows computer, Kony Visualizer displays a dialog box prompting you for information so that the debugger can connect to the Mac computer running the iOS simulator. Once you have entered the needed information, click Test Connection. If the connection is configured properly, the Result reads **Test connection successful**. Click **OK**.



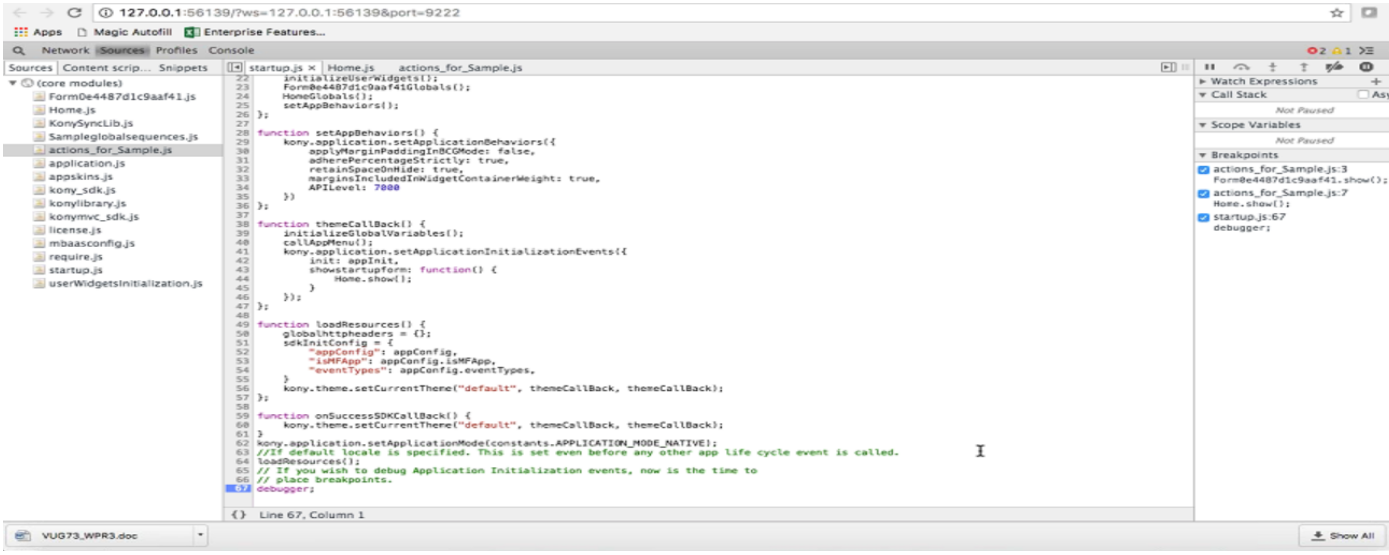
6. Kony Visualizer tests to see if it can locate Google Chrome on your computer. If it does not find Chrome in the Program Files folder, a Google Chrome Location field appears in the above dialog box. Click its corresponding **Browse** button to locate the Chrome executable.



7. Once the connection to the Mac computer running the simulator is established and Kony Visualizer has verified the location of Google Chrome, Kony Visualizer launches the iOS WebKit Debug Proxy tool. If Kony Visualizer does not find the iOS WebKit Debug Proxy tool, it prompts you to manually run the necessary script file on your Mac computer.



8. Upon the successful start of the iOS WebKit Debug Proxy tool, Kony Visualizer opens a new Google Chrome debugger window. JavaScript files that you open appear on the Sources panel of the window, and you can add breakpoints and step through the code using the Debugger panel.



You can then use the Google Chrome debugger to debug the application. For information on using the Chrome debugger, see Get Started with Debugging JavaScript in Chrome DevTools on the Google Chrome website.

Kony Visualizer

Kony Visualizer supports JavaScript debugging for iOS native applications with Safari web browser and Web Inspector. Ensure that you have the latest version of Safari web browser installed in your system. Ensure that you enable developer settings in your iOS device.

To get started with debugging iOS native applications, you must enable Developer Settings on your iOS device. If your Mac has an OS earlier than Mojave, the Developer settings are automatically configured. If your Mac has Mojave OS or later, configure Mojave OS Specific settings.

Once you are done configuring the settings, you can Build iOS Native Application in the Debug mode and then further, Debug the iOS Native Application using Safari Browser.

iOS Device Developer Settings

For the debugging to work, you must enable the Web Inspector developer tool on your device before debugging an application. To enable Web Inspector tool, follow these steps:

1. On your iOS device, open **Settings**.
2. Go to **Safari** > **Advanced**.
3. Toggle the button next to **JavaScript** to enable JavaScript in the Safari browser.

   Toggle the button next to **Web Inspector**to debug iOS native applications.

Mac Developer Settings

For the Mac OS versions earlier than Mojave, Visualizer automatically connects to the Safari JavaScript debugger, provided that the developer settings in your iOS device are configured.

Mojave-specific Settings

For the Mac OS versions Mojave and later, full disk access and accessibility must be provided to the Terminal due to enhanced security of newer Mac OS versions.

For the debugger to run on your Mac, follow these steps:

1. In your Mac, go to **System Preferences** > **Security and Privacy**.
2. Click the **Privacy** tab, and then click the lock icon at the bottom of the screen. You may receive a password prompt.
3. If the password prompt appears, enter your password.
4. Once you can access the settings, from the left pane, select **Full Disk Access** .
5. Click the **+** icon to add the **Terminal** to the list of applications with **Full Disk Access**.
6. From the left pane, select **Accessibility**.
7. Click the **+** icon, and then add **Terminal** to the list of applications for accessibility.

> ***Note:*** If Visualizer fails to automatically communicate with Safari to launch Web-inspector, you must debug the application manually.

Build and Debug iOS Native Application

Build iOS Native Application in the Debug Mode

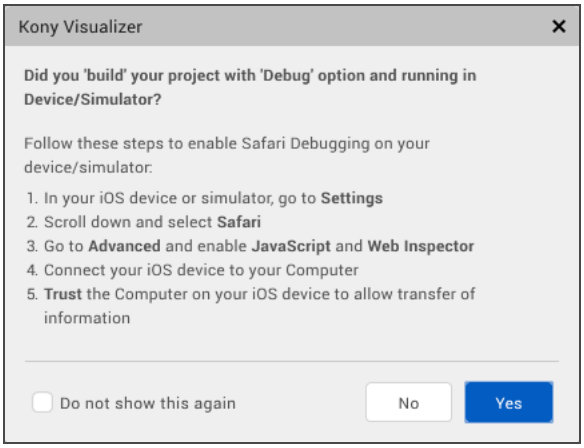
To debug an application, you must first build the application in the Debug mode. To do so, follow these steps:

1. In Kony Visualizer, from the **Build** menu, click **Build and Publish Native**.
2. In the **Build** dialog box, from the **Build Mode** drop-down list, select **Debug**.
3. Click **Build**.
4. If you haven't configured the P12 password, .P12 certificate, Mobile Provisioning profile, target iOS version, and development method in the **iPhone** tab of the Project Settings, you will see an error dialog in Visualizer. Click **Settings** to configure the certificates. Refer to iOS automation for more information on configuring the certificates.

Debug iOS Native Application

Follow these steps to debug JavaScript applications for an iOS device:

1. In Kony Visualizer, from the main menu, click **Debug**.
2. Go to **Debug Native App** > **Connected iOS Devices**.
3. From the list of connected devices, select your device.
4. If you haven't enabled the Developer settings in your iOS device already, follow the steps in the popup and click **Yes**.

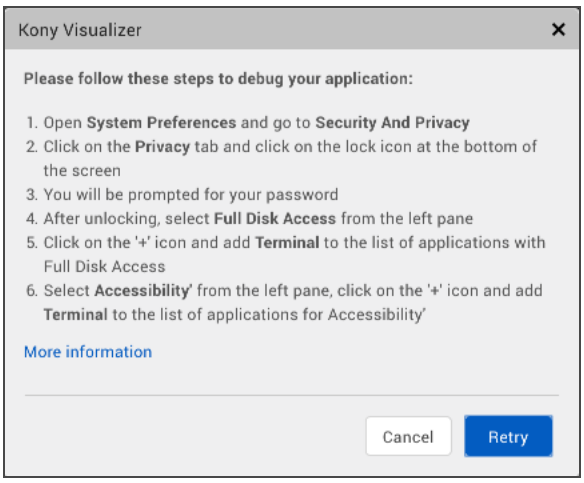


   You can select the **Do not show this again** checkbox to not see the popup again.

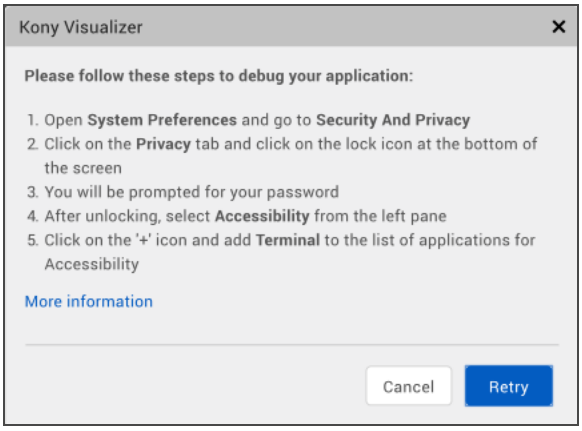
   If the debug process is successful, the Safari web browser with Web Inspector is auto-launched.

5. In case the Safari web browser does not launch automatically, you must debug the application manually.

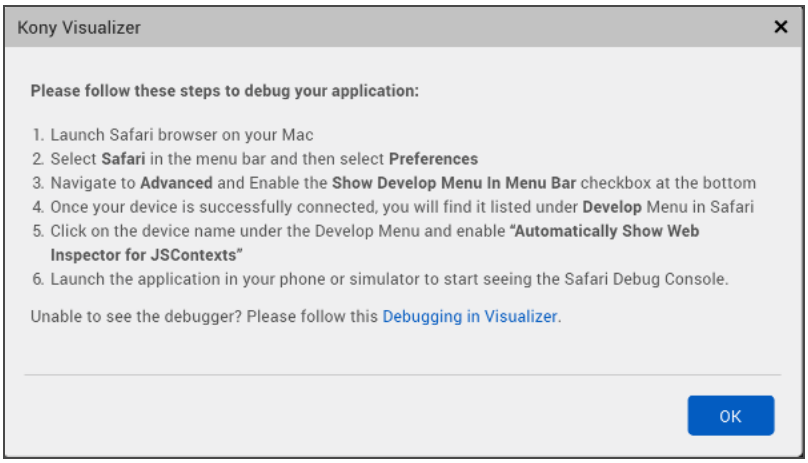
   - If your Mac OS version is Mojave or later, you must configure the Mojave-specific settings or follow the steps in the following popup.



- If your Mac OS version is earlier than Mojave, ensure that you have provided the accessibility to the terminal by following the steps in the popup.



- If you have configured the Security and Privacy settings and the Safari web browser still does not launch, you must debug the application manually or follow the steps in the popup.



Manual Debugging

If the application debugging fails and the Safari web browser does not launch automatically, you must configure the following steps manually:

1. Launch Safari web browser.
2. From the menu bar, go to **Safari** and click **Preferences**.
3. In the **Preferences** menu, click **Advanced**.
4. Select the **Show Develop Menu In Menu Bar** check box.
5. Connect your device to your Mac system using a USB.
6. From the menu bar, click the **Develop** menu, you will see the device name that is connected to your Mac system.
7. Click on the device name and enable **Automatically Show Web Inspector for JS Contexts**.
8. Launch the application on your device to see the Safari Debug Console.

Debug JavaScript for Android in Kony Visualizer

With Kony Visualizer, you can perform JavaScript debugging on connected Android computers and devices. This procedure assumes that you have the Google Chrome web browser installed on your computer.

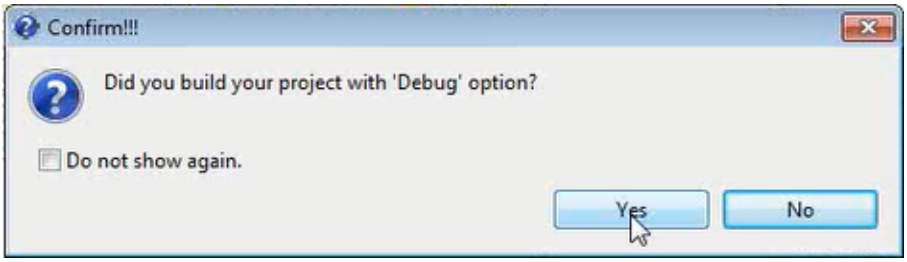
> ***Note:*** By default, the Android debugger uses port 9222. If there is a conflict, you can change the port that the debugger uses.
> In Kony Visualizer Classic, in the **Window** menu, click **Preferences**. From the Preferences dialog box, click **Kony Visualizer** and update the **Android Debugger Port** settings.
> In Kony Visualizer in the **Edit** menu, click **Preferences**. From the Preferences dialog box, click **Build** and update the **Debugger ports** settings.

> ***Note:*** You can now use Android FP debugger from the Preview menu. This uses the Kony Quantum App app to debug the app. By default, the Android FP Debugger Port feature uses 9333. If there is a conflict, you can change the port that the debugger uses.
> In Kony Visualizer Classic, in the **Window** menu, click **Preferences**. From the Preferences dialog box, click **Kony Visualizer** and update the **Android FP Debugger Port** settings.
> In Kony Visualizer in the **Edit** menu, click **Preferences**. From the Preferences dialog box, click **Build** and update the **Debugger ports** settings.

To debug JavaScript applications for Android in Kony Visualizer:

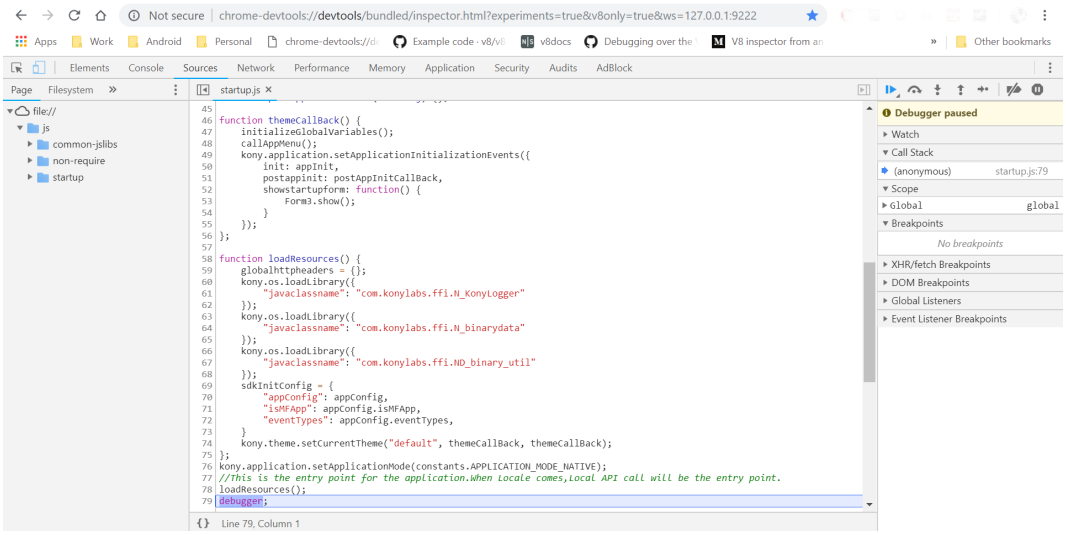
1. In Kony Visualizer, build your application for the Android platform in debug mode. To do so,
   - In Kony Visualizer Classic, from the **Product** menu , click **Build**. In the Build dialog box, select Debug from the Build Mode drop-down list.
   - In Kony Visualizer, from the **Build** menu, click **Build and Publish Native**. In the Build dialog box, select Debug from the Build Mode drop-down list.
2. Launch the application in an emulator or device. The device must be connected using a USB connection.

3.
   - For Kony Visualizer Classic, from the **Product** menu, point to **Debug As**, and then select **Debug android application**.
   - For Kony Visualizer, from the **Debug** menu, click **Debug Native App**. From the **Debug Native App** drop-down list, select **Android Devices/Simulators**.
4. A confirmation dialog box appears asking if the project was built with the Debug option. If you know that it was, click **Yes**. Otherwise, click **No**, which ends the debug process.



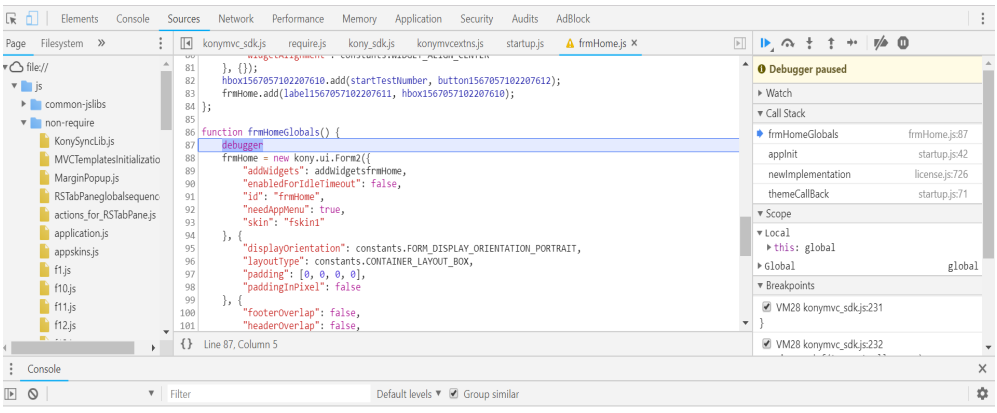
> ***Note:*** To debug applications in Kony Visualizer, you must have built the application using the Build Mode **Debug**. To do so, in the Build dialog box, select Debug from the Build Mode drop-down list.

5. Kony Visualizer displays a popup that the debugging has started. Open your Google Chrome browser and enter the URL displayed in the popup into the address bar of the browser.
   The URL will be as follows, **chrome-devtools://devtools/bundled/inspector.html?experiments=true&v8only=true&ws=127.0.0.1:<port_number>**
   For example,
   chrome-devtools://devtools/bundled/inspector.html?experiments=true&v8only=true&ws=127.0.0.1:9222
   JavaScript files that you open appear in **Sources** panel. You can add breakpoints and step through the code using the Debugger panel.



You can then use the Google Chrome debugger to debug the application. For information on using the Chrome debugger, see Get Started with Debugging JavaScript in Chrome DevTools on the Google Chrome website.

> ***Important:*** In Google Chrome version 66 and higher, the breakpoints do not get applied. You can write the keyword **debugger** in the JS files where the debugger needs to be stopped. This **debugger** keyword acts as a breakpoint.



Note the following behavior when debugging an Android application:

- When the debugger connects, the focus changes to the DevTools console. However, the debugger is paused in *startup.js*, which can be found on the **Pages** tab which is under the **Sources** tab. You must manually switch to the **Sources** tab to debug your JavaScript code.
- Your application stops working when you try to perform a JSBinding operation after the JSDebbuging is complete.

Expose Widget IDs for Test Automation

Applies to *Kony Visualizer Classic*.

Kony Visualizer can expose the IDs of widgets so that they can be used by a test automation framework. This option is available for iOS, Android, and Windows, and is disabled by default.

This section covers the following topics:

Expose the Widget IDs

Limitations by Platform

Expose the Widget IDs

To expose the widget IDs, do the following:

1. On the File menu, click Settings. The Project Settings dialog box displays.
2. Click the **Native** tab.
3. On the Common sub-tab, click **Expose Widget IDs for Test Automation**, and then click **Finish**.

For the Android platform, when you select to expose widget IDs, an xml file is generated during the build containing the widget IDs of the app. This is explained in detail under Android Limitations.

Limitations by Platform

Each supported platform has its own set of limitations with respect to using exposed widget IDs for test automation.

iOS Limitations

The iOS platform has the following limitations:

- No two widgets within the same form can have same widget ID.
- The content of the following widgets cannot be accessed using the exposed widget IDs:

  List, PickerView ,OnScreenWheel, SegementedUI, ToggleView
- Only widgets that are visible within the screen area of the canvas have their widget IDs exposed.
- For widgets that are partially exposed within the screen area of the canvas, such as a ComboBox, CheckBoxGroup, RadioButtonGroup, and SegmentedUI–only the child elements that are visible, such as rows and tabs, have their IDs exposed.
- Annotations in a Map widget cannot be accessed.
- The IDs of custom widgets are not exposed.

Android Limitations

The Android platform has a number of limitations that you want to be mindful of when it comes to exposing widget IDs for test automation.

XML Files of Static and Dynamic Widgets

To make use of the exposed widget IDs in automated testing for the Android platform, Android relies on an xml file that captures all the widget IDs.

For static Kony Visualizer widgets–those that you can drag and drop using the user interface–the IDs of these widgets are automatically collected at build time into an xml file called `widgetids.xml`. The paths are as follows, depending on whether the app is built for the Mobile or Tablet channels:

Android Mobile

`<WorkSpaceName>\<ProjectName>\resources\mobile\native\android\values\`

Android Tablet

`<WorkSpaceName>\<ProjectName>\resources\tablet\native\androidtab\values\`

For dynamic widgets–those defined in a JavaScript module–if you want them to be accessed by automated tests, you need to create a separate xml file listing them called `dynamicwidgetids.xml`. The path and file name are as follows, depending on whether the app is built for the Mobile or Tablet channels:

Android Mobile

`<WorkSpaceName>\<ProjectName>\resources\mobile\native\android\values\`

Android Tablet

`<WorkSpaceName>\<ProjectName>\resources\tablet\native\androidtab\values\`

The xml file `dynamicwidgetids.xml` needs to be structured in the following way:

- For every widget ID, there must be one *Item* tag entry.
- Each widget ID represented by an *Item* tag must have a *Type* property of *id*.
- All the *Item* tags must be enclosed in a *Resources* block.
- All widget IDs must be unique. No duplicates.
- No dynamic widget IDs should conflict with the IDs of the static widgets.

The following is an example of how the file `dynamicwidgetids.xml` should be structured:

```
<? xml version = "1.0"
encoding = "utf-8" ?> < resources > < item name = "widgetid1"
type = "id" / > < item name = "widgetid2"
```

```
type = "id" / > < item name = "widgetid3"
type = "id" / > < item name = "widgetid4"
type = "id" / > < /resources>
```

Other Android Limitations

- Unique IDs for the child elements of the following widgets cannot be exposed for test automation:

  Browser, Calendar, CheckBoxGroup, ComboBox, DataGrid, ImageStrip, ListBox, PickerView, RadioButtonGroup, Segment, Tab

  However, you can use the following snippet to retrieve the child object:

  ```
  UiObject radiobutton1 = new UiObject(new UiSelector().resourceId
  ("com.konylabs.android:id/radiobuttongroup103994549623788")).getC
  hild(new UiSelector().index(1));
  ```

- The child elements and content of a RichText widget cannot be accessed. This is native behavior for Android.

Windows Limitations

The Windows platform, specifically for the Tablet channel, has the following limitations with respect to using exposed widget IDs for test automation:

- Only Winium and CodedUI is supported.
- IDs cannot be given to a header in a Segment.
- Using Winium, within a Popup, ListBox widgets and Message dialogs cannot be automated.

Android USB Debugging on Windows 10

You can debug your Android app on an Android device using the USB. To debug an Android app on Windows 10, do the following:

- Enable USB debugging on your Android phone
- Install Java SE SDK 8 update 152
- Configure Android SDK
- Configure Google Chrome for debugging

Enable USB debugging on your Android phone

On Android 4.1 and lower, the Developer options screen is available by default. On Android 4.2 and higher, do the following:

1. Open the **Settings** app.
2. Select **System**.
3. Scroll to the bottom and select **About phone**.
4. Scroll to the bottom and tap **Build number** 7 times.
5. Return to the previous screen to find **Developer options** near the bottom.
6. Scroll down and enable **USB debugging**.

Install Java SE SDK 8 update 152

Install Java SE SDK 8 update 152, click Java SE SDK 8 Update 152 and follow the instructions on your Windows 10 machine.

Configure Android SDK

To download and configure Android SDK on your Windows 10 computer, do the following:

Ensure that the same Android SDK version is used in Visualizer.

1. Download the Android Studio .
    i. Click sdk-tools-windows-3859397.zip to download the Android SDK Command line tool for Windows.
    ii. Extract the downloaded content to **C:\Users\USERNAME\AppData\Local\Android**.
    iii. Navigate to **C:\Users\USERNAME\.android** and create an empty file named **repositories.cfg**.

2. Install the Android SDK platform tool.
    i. Navigate to **C:\Users\USERNAME\AppData\Local\Android\tools\bin**
    ii. Open a command window (Shift-Right click, Open Powershell window here, type `cmd` in the Powershell window).
    iii. Run the command `sdkmanager platform-tools` to install the Android SDK platform tools.

3. List the Android devices connected to the Windows 10 PC.
    i. Navigate to **C:\Users\USERNAME\AppData\Local\Android\platform-tools**.
    ii. Open a command window.
    iii. Run `adb devices -l` to list the Android devices connected to the Windows 10 PC.

4. Connect to the Android device.
    i. Navigate to **C:\Users\USERNAME\AppData\Local\Android\tools**
    ii. Run `monitor.bat` and click on the connected device.

Ensure that your environment variables are set correctly from here.

Now your Android device is successfully connected to your Windows 10 PC and can be used to debug your application in Kony Visualizer Classic.

Important: Before you do start debugging your Android app on the Windows 10 machine, ensure that Google Chrome is version 62 or earlier. Debugging will not work with version 63 and later.

Configure Google Chrome for debugging

To configure Google Chrome for Android USB debugging, do the following:

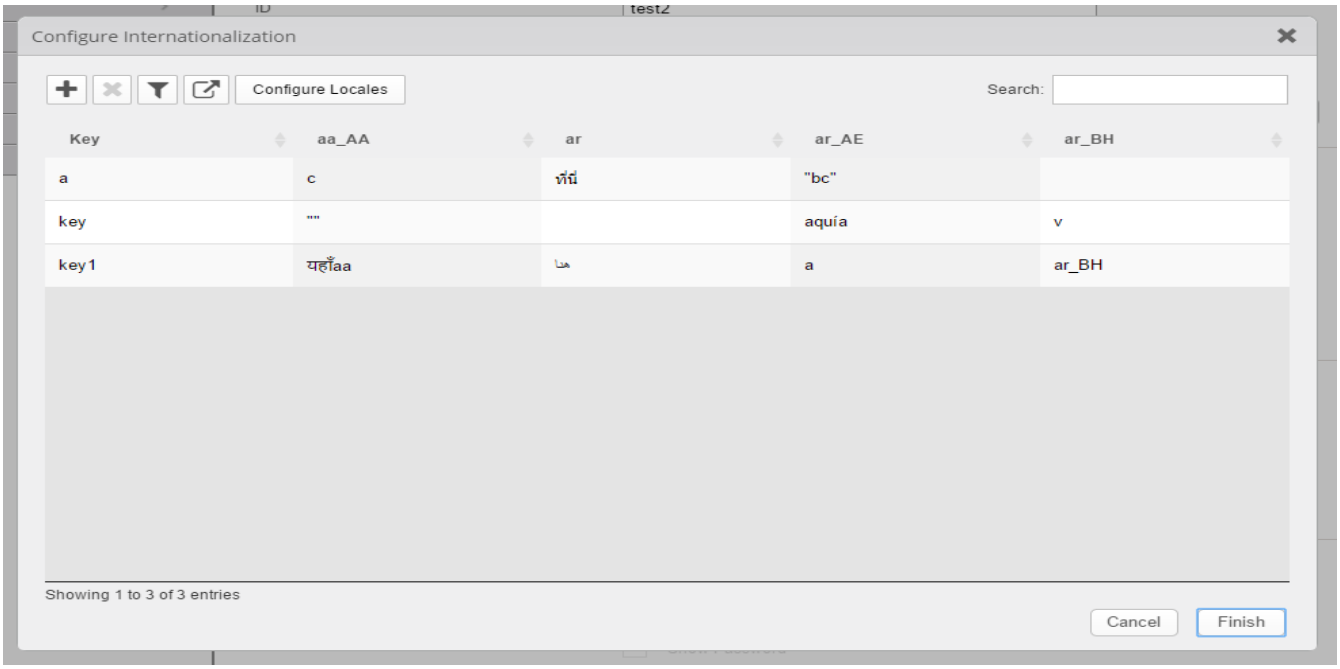
1. Download Chrome version 61 from https://www.slimjet.com/chrome/google-chrome-old-version.php
2. Uninstall the current version of Chrome.
3. Stop all Google Processes in Task Manager.
4. Disable Google Update from the Windows Startup.
5. Stop the Google Update Services.
6. Erase the directory **Update in C:\Program Files (x86)\Google**.
7. Install Chrome 61 from the **Downloads** directory.
8. Quit Google Chrome 61 as it starts.
9. If Google Chrome has already started, it will begin the auto-update. In that case:
    i. Exit Google Chrome
    ii. Navigate to **C:\Program Files (x86)\Google\Chrome\Application** and delete the file **update_chrome.exe**.
    iii. Erase the directory Update in **C:\Program Files (x86)\Google**.
10. Start Google Chrome 61 and check the version.
11. To disable Google Chrome automatic updates if they are re-enabled, change the following registry key: Set **HKEY_LOCAL_**

MACHINE\SOFTWARE\Policies\Google\Update\AutoUpdateCheckPeriodMinutes to the **REG_DWORD** value to **0**.

12. See https://support.google.com/chrome/a/answer/6350036#Policies for more information about disabling Google Chrome automatic updates.

Internationalizing (i18n) Application Content

The process of designing or developing an application in such a way that it supports various languages and regions is referred to as *Internationalization*, also known as *i18n*. In Kony Visualizer, the i18n features make it possible to build your app for various language locales without making changes to the application code or logic. By setting up the locales that you want to support and then adding what amounts to a vocabulary list for each locale, your app can become functional in multiple languages. Each entry that you make is called a key, and defines a given term in each of the locales that you want your application to support.



The following topics lead you through the process of implementing i18n functionality in an app:

Create Locales

Add i18n Content for Each Locale

Assign an i18n Key to a Widget

Search for An i18n Key

View a Locale Using the Preview App

In addition, you can also export a project's i18n settings and import them into another project.

Export Internationalization Settings

Import Internationalization Settings

Updating i18N Keys on Android Applications

Create Locales

To create locales, do the following:

1. In Kony Visualizer, on the Project (Edit menu for Kony Visualizer Classic) menu, click **Internationalization (i18n)**. The Configure Internationalization window appears.
2. Click **Open Locales**.
3. On the Predefined tab, select the checkboxes of the locales that you want your app to support. If you do not see the locale you want, click the Custom tab, and then add your own locale, giving it values for Language, Country, and Locale. For these values, use the patterns found in the predefined locales as examples to guide you.

   > ***Notes:***
   > - The Locale field must use the format aa_AA or aa. For example, jp_JP, or jp would be a valid locale name.
   > - The Country field can only contain letters.
   > - The Language field must begin with a letter.

4. Once you have created your locales, click **OK**. The Configure Internationalization dialog box now has a column for each locale you created. For every locale you configure, a corresponding empty folder is created in your workspace using the following path:
   `<workspace>\<application>\resources\common`
   You can add images to this folder which you can use in the project for corresponding locale.
5. Set the default locale by selecting the locale you want from the Default Locale drop-down list.

6. Click **Finish**.
   With your locales chosen, you can now define i18n keys for each locale.

Add i18n Content for Each Locale

When it comes to adding i18n content to your application, you have a couple of options:

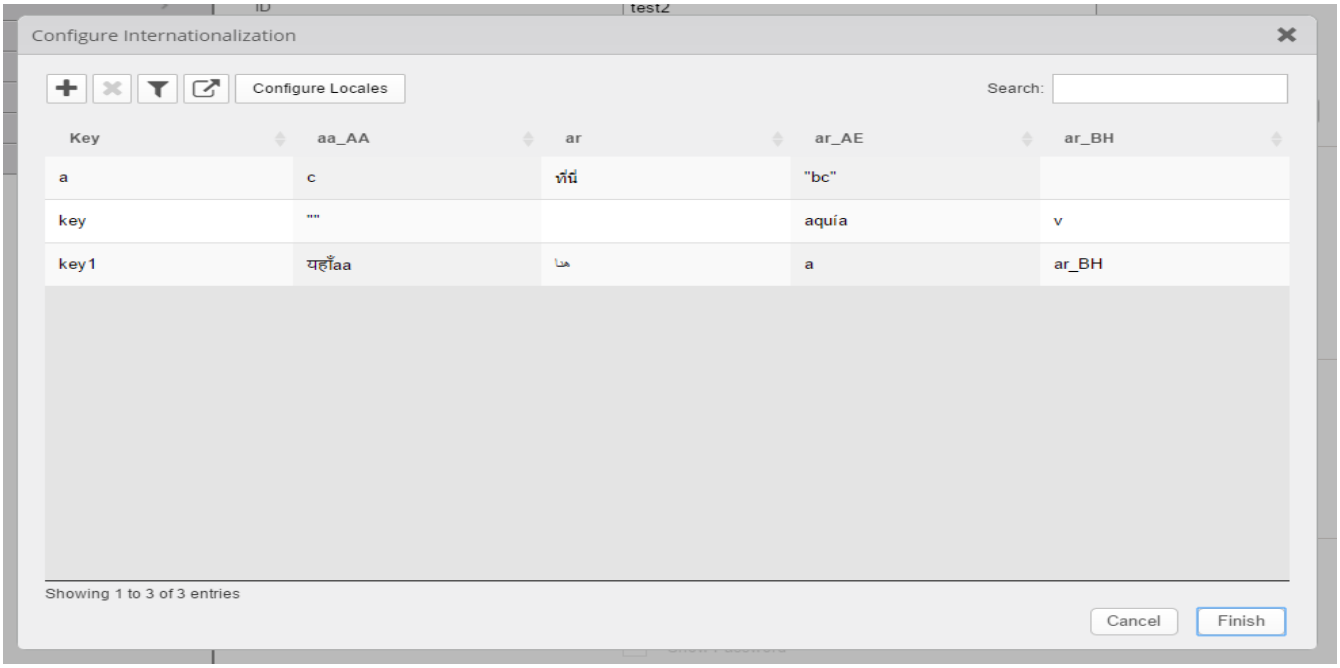
- You can add i18n content within Kony Visualizer
  - By adding i18n keys one after the other manually, or
  - By adding all the i18n keys at once from a resource bundle
- You can add i18n content programmatically from the code using APIs.

Add i18n Keys Manually

This procedure assumes that you have already created the locales that you want your app to support. For more information, see Create Locales.

To add i18n keys manually, do the following:

1. In Kony Visualizer, on the **Edit** menu (the **Project** menu for Kony Visualizer), click **Internationalization (i18n)**. The Configure Internationalization dialog box displays.
2. In the Configure Internationalization dialog box, an initial row for an i18n key and the locales you want your app to support is displayed. In the Key field, type a name for your key, such as *key001*.
3. In each locale's field for the key you just named, enter the word or phrase for the term you're defining. For instance, if you're entering the Spanish equivalent for the English phrase *Sign In*, you would enter *Iniciar Sesión*.
4. Add a new row for each key you create by clicking the green plus sign button.
5. Repeat steps 3 and 4 for each new key you want to create.



6. Click **Finish**.

Add i18n Programmatically Using APIs

For information on how to use APIs to programmatically add i18n functionality to an app, see the Kony Visualizer API Developer's Guide.

Assign an i18n Key to a Widget

To assign an i18n key to a widget, do the following:

1. Using either the Project Explorer or the Visualizer Canvas, select a widget.

   The text of a widget uses the default language of the project.

2. On the Look tab of the Properties pane, select the key you want from the **Text i18n Key** drop-down list, or using the search icon, search and filter for the key you want.

> ***Note:*** For Group Widgets (such as ComboBox, DataGrid, and CheckBoxGroup) and Segment Widget, you assign i18n keys in the Master Data.

Search for An i18n Key

When adding editing, or deleting i18n keys, you can make it easier to locate particular keys by searching for them.

To search for an i18n key, do the following:

1. In Kony Visualizer, on the **Edit** menu (the **Project** menu for Kony Visualizer), click **Internationalization (i18n)**. The Configure Internationalization dialog box displays.
2. In the **Search** text box, type text that matches the key you're looking for. Keys that contain the text you type are listed in the table of i18n keys.

View a Locale Using the Preview App

To get a sense for what the user will experience when viewing an app in a particular locale, you can use the Kony Visualizer preview feature. The preview feature builds a preview version of the app, posts it to the cloud, and then gives you a publish code that you—or anyone you give the code to—can use to view the app on a device, such as a smartphone. The device simply needs to have the Kony Visualizer Preview App installed, which is available free from Kony on your device's app store.

To view a locale using the Preview App, do the following:

1. Set the default locale of the app to the locale that you want to view using Functional Preview. To do so, on the Project (Edit menu for Kony Visualizer Classic) menu, click **Internationalization (i18n)**, and then select the locale you want from the Default Locale drop-down list.
2. Follow the instructions provided in the topic, Preview an App on a Device.

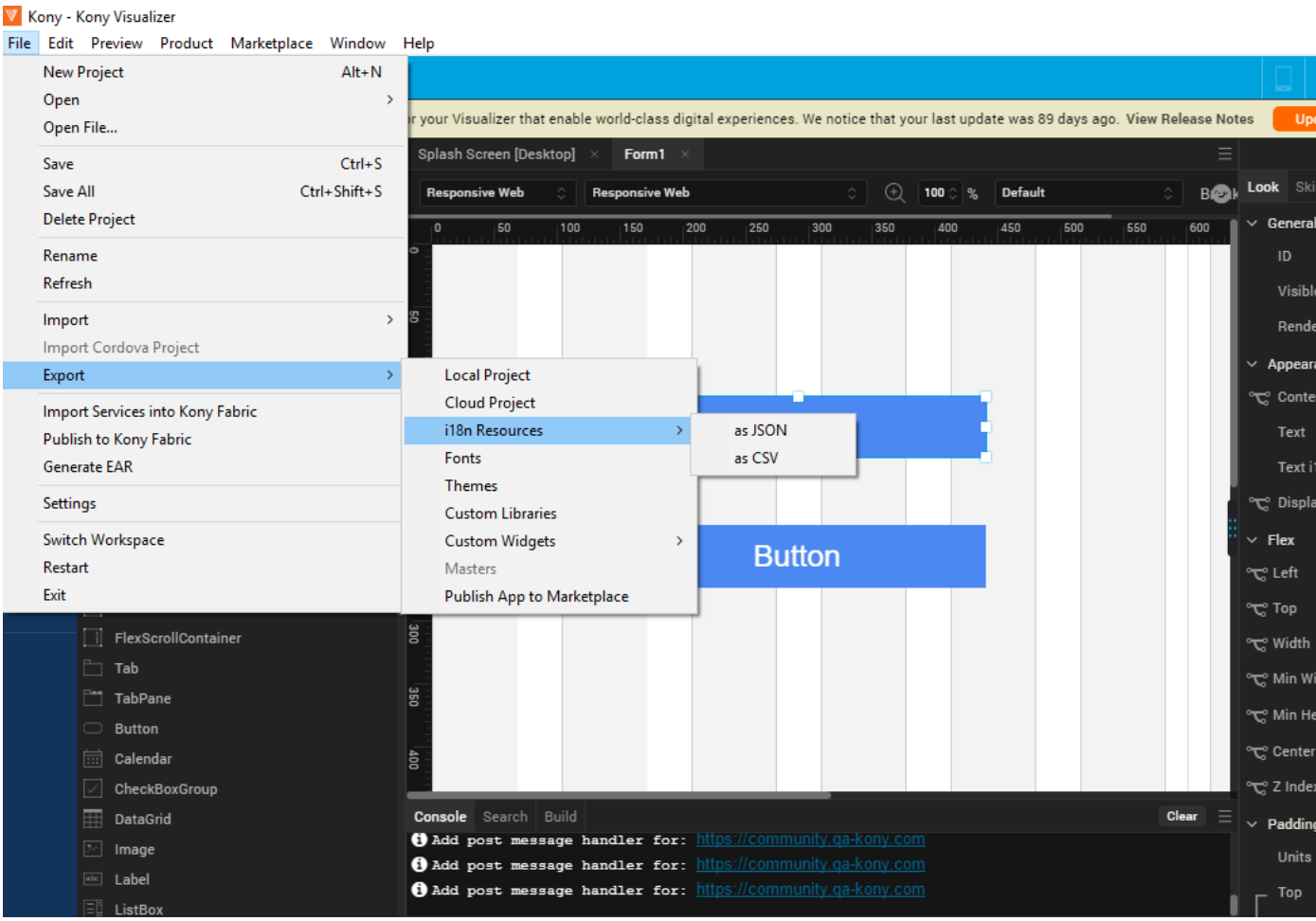
Export Internationalization Settings

You can export your i18n settings so that you can import them for use in a different Kony Visualizer project. From V8 SP4 onwards, you can export your i18n settings either as a JSON or as a CSV file format.

To export i18n settings, you can perform any of the following actions.

From the File Menu

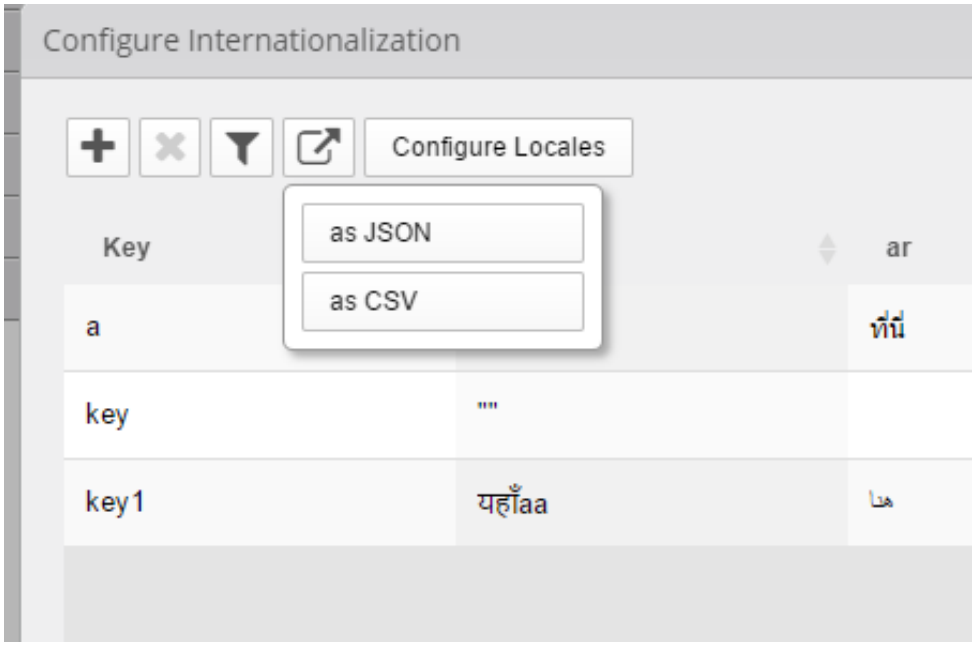
1. In Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer) , point to **Export**, and then click **i18n resources**.



2. Select the file format as either **JSON** or **CSV**. The Save As dialog box appears.
3. Navigate to the folder where you want to save the i18n settings, and then click **Save**. Kony Visualizer exports the i18n resources as a zipped file.

From the Edit Menu

1. In Kony Visualizer, on the Edit menu (the Edit menu in Kony Visualizer) menu, click **Internationalization (i18n)**. The Configure Internationalization window appears.
2. Click the **export** icon beside the Configure Locales button.
3. Select the file format as either **JSON** or **CSV**. The Save As dialog box appears.
4. Navigate to the folder where you want to save the i18n settings, and then click **Save**. Kony Visualizer exports the i18n resources as a zipped file.



Import Internationalization Settings

If you have the exported i18n settings from another Kony Visualizer project, you can import them into the project you're working on. From V8 SP4 onwards, you can import i18n settings either as a JSON or as a CSV file format.

To import i18n settings, do the following:

1. In Kony Visualizer, on the **File** menu (the **Project** menu in Kony Visualizer) , point to **Import**, and then click **i18n resources**.
2. In the Open dialog box, navigate to the zipped file that contains the i18n settings you want to import, select the file, and then click **Open**.

Important Considerations

The following are important considerations when importing i18n keys:

- If the current project contains i18n keys, the imported keys are added to the existing i18n file.
- If the name of an existing key is the same as an imported key, then the imported key values overwrite the existing key values.
- If the name of an existing key is the same as an imported key but the locales are different, then only those locales that are common are overwritten.
- If the imported i18n keys have locales that are not defined in the current project, then Kony Visualizer activates these locales in the current project, and then assigns the imported i18n values.

Resource Bundle

A resource bundle is a properties file that contains all the i18n keys, along with their values. It is locale-specific, that is, you will have one resource bundle per locale. To use resource bundles, you design your application to access the locale of choice at run-time, depending on the language the user chooses to display the app in. When the user chooses a different language, the resource bundle for

that locale is implemented. A resource bundle follows the naming convention of `<language_Country>.properties`. For example,

- For the United States English locale, the resource bundle is `en_US.properties`
- For the French Canadian locale, the resource bundle is `fr_CA.properties`.

Updating i18N Keys on Android Applications

If your Android app is published even once, when you modify the app (including i18N locales, values), you must ensure to change the version of the app from project settings. Modifying the Version Number and Version Code are mandatory steps for the changes to reflect in the app.

To update the Version Number, do the following:

1. From the **File** menu, navigate to **Project Settings**.
   The **Project Settings** window opens.
2. In the **Application** tab, update the version number to the next version. For example, if the **Version** is 1.0.0, modify it to 1.0.1
3. Click **Finish**.

To update the Version Code, do the following:

1. From the **File** menu, navigate to **Project Settings**.
   The **Project Settings** window opens.
2. Navigate to **Native** > **Android**.
   The Android tab contents appear. By default, the Mobile/Tablet tab displays.
3. Update the entry in the **Version Code**. For example, if the **Version Code** is 1.0, modify it to 1.1
4. Click **Finish**.

Supporting Right-to-Left Languages

Kony Visualizer V8 SP1 supports Right-to-Left (RTL) languages. When RTL layout is enabled, the user interface is mirrored. The text direction changes from right-to-left.

The Internationalization section of Visualizer has a new i18N property (Project Settings > Application in Kony Visualizer). When a user configures locales, the user can define each one of the following:

- **Flex Position properties**: When enabled, layout properties are reversed from Left to Right (LTR) and Right to Left(RTL).
- **Content Alignment**: When enabled, the flex content is aligned from right to left.

From V8 SP4 onwards, we also have:

- **Flow Horizontal Alignment**: When enabled, flow horizontal left alignment is converted to flow horizontal right alignment.

By default, these options are disabled.

- In Kony Visualizer, after the Enable i18n feature is enabled in Project Settings, a new field, **Enable i18n Layout Config** appears beside it. For Kony Visualizer Classic, the Enable i18n Layout Config always appears by default.
- At form level, for those widgets on the form on which i18n is applicable, a new **Replace i18n Layout** is enabled in the **Properties** > **Look** section. Using these fields, you can allow exceptions for this right to left layout configured at the locale level.
- The exception toggle is enabled when you set **Custom** to **On** from under the Replace i18n Layout.
- The toggles for all the properties are only visible when Custom is configured as On. The default value for these is Off.
- Once Custom is set to On, all locales respect only the custom configuration.

- For any widget, once the **Custom** button is set to **On** and if one or more of the toggles (Flex Position Properties / Content Alignment / Flow Horizontal Alignement) are set to **On**, then the widget ignores the configurations set at locale level and renders the layout as it is designed.

Click here to watch a video on locale based layout.

Important Considerations

- For any widget which has an exception toggle set, there won't be any changes to the layout.
- For any widget where **Left** is set, and **Width** is set, but **Right** is empty, the **Left** value will be replaced into the **Right** value, and the Left value will be changed to NULL
- For any widget where **Right** is set and **Width** is set, but **Left** is empty, the **Right** value will be replaced into the **Left** value, and the Right value will be changed to NULL.
- For any widget which has both Right and Left Values set, the values are swapped.
- For any widget with content alignment is configured to **Left**, content alignment is configured to **Right**
- For any widget where content alignment is configured to **Right**, content alignment is configured to **Left**.
- For any widget, if the **Padding** has a Left value, the Left value will be replaced into the Right value.
- For any widget, if the **Padding** has a Right value, the Right value will be replaced into the Left value.
- From V8 SP4 onwards,
    - For any Container widgets, if the layout type is set to Flow Horizontal, the layout type changes from **Flow Right** to **Flow Left**.
    - For any Container widgets, if the layout type is Free form, the Left layout property is converted to Right and Right layout property is converted to Left. Also, the widget order is reversed.

Supported Widgets

- Flex Container
- Flex Scroll Container
- Label
- Text Box
- Button
- Text Area
- Switch
- Segment
- Image
- Camera
- Map
- Rich text widget
- Browser widget
- Slider
- Video
- ComboBox
- Phone
- PickerView
- Tab Pane
- Tab

Configure Right-to-Left Layout

To configure internationalization, do the following:

1. On the **Edit** menu (**Project Settings**> **Application** in Kony Visualizer), click **Internationalization (i18n)**(click **Configure** in Kony Visualizer). The Configure Internationalization page opens.
2. Check the **Enable i18n Layout Config** check box.
   For Kony Visualizer, on the Application tab, check the **Enable i18n** checkbox. When you select the Enable i18n checkbox, the **Enable i18n Layout Config** checkbox is enabled. Select Enable i18n Layout Config checkbox.
3. Click **Configure Locales**. The Configure Locales page opens. The page displays predefined languages that support Right-to-Left layout.
4. For any of the language, you want to enable RTL, select the locale. You can choose to select the checkboxes for check Flex Position Properties, Content Alignment, and Flow Horizontal Alignement.

Exceptions for Right-to-Left Layout from the Properties Pane

Every widget has an exception toggle for the supported properties. You can enable this toggle by selecting **Custom** under **Replace i18n Layout**.

To configure exceptions for Right-to-Left layout from the properties pane, do the following:

1. In the Project Explorer, on the Project tab, expand the forms. Select the form for which you want to configure Right-to-Left layout.
2. Click the widget for which you want to change the property.
3. On the **Properties** pane, in the **Look** tab, the **Replace i18n Layout** is visible.
4. If this is turned off, select the **On** radio button. The properties appear.

5. Select On or Off radio button as required for the following:
    a. **Flex Positions Properties**: Reverses layout properties for Left and Right
    b. **Content Alignment**: Content alignment changes from Right to Left and or from Left to Right.
    c. **Flow Horizontal Alignment**: Flow horizontal alignment changes from Left to Right.
6. Save the changes.

Known Limitations

- Support for Right to Left languages is not added for widgets in the VBox layout.
- Support for Right to Left languages is not added for Watch channel.
- For Components and Masters, the user will be able to set the properties under **Replace i18n Layout** at source only but not on instance widgets. Also, these are not pass through properties for Components with the contract.
- When the user provides values for below fields and switches to Right-to-Left locale, the layout in Visualizer Canvas is not in sync with the layout in app runtime.
    - Center X and Left/Right
    - Width, Left and Right
- The layout does not mirror on switching to Right-to-Left locale when the values for Center X and Width are given.
- For a Segment widget, if the content is set using the data property, setData API, and setDataAt API, the content does not get mirrored.

Android Build Environment and Configurations

Android developers can customize and configure Android build environments using the following information:

1. Access the Generated Android Project for Kony Application
2. Main Activity and its Life Cycle Methods
3. Android Pre-compile and Post-compile Ant Tasks Support
4. Support for Integrating Android Third-Party Libraries With Kony Project

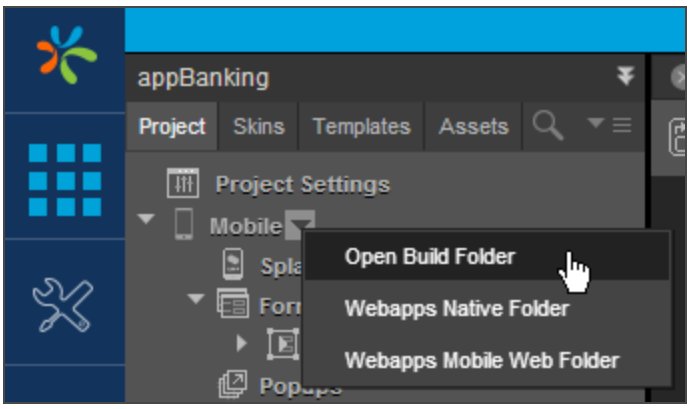
Access the Generated Android Project for Kony Application

If an application is built in Kony Visualizer for the Android platform, a corresponding native Android project is generated.

Changes made to the project can be compiled in the command line using `gradle assembleDebug`, `gradle assembleRelease` by navigating to the native Android application folder in the command line.

To access the generated Android project, do the following:

1. In Kony Visualizer, in the Project Explorer, on the Project tab, click the context menu arrow for your platform of choice (e.g. Mobile), and then click Open Build Folder.
2. Launch Kony Visualizer.
3. In Kony Visualizer, in the Project Explorer, on the **Project** tab, click the context menu arrow for your platform of choice (e.g. Mobile), and then click **Open Build Folder**. Open the build folder. A folder opens with the directory structure `<WorkSpace>\temp\<AppID>`.



4. Navigate to the respective folder:
   - **Mobile**: `<WorkSpace>\temp\<AppID>\build\luaandroid\dist\<AppID>`.
   - **Tablet** : `<WorkSpace>\temp\<AppID>\build\luatabrcandroid\dist\<AppID>`.

Main Activity and its Life Cycle Methods

If the application package name is **com.xyz.sample** and the Application ID is **SampleApplication**, the Kony Visualizer build process generates a Main Activity Java source file with Application ID as its name in the following path:

- **Mobile**: `<WorkSpace>\temp\SampleApplication\build\luaandroid\dist\SampleApplication\src\com\xyz\sample\SampleApplication.java`
- **Tablet**: `<WorkSpace>\temp\SampleApplication\build\luatabrcandroid\dist\SampleApplication\src\com\xyz\sample\SampleApplication.java`

A developer can add custom code and add/modify overridden Android activity life cycle methods such as onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy(), onNewIntent (Intent), and so on. A developer can also invoke any third-party APIs directly from overridden life cycle methods of the class.

While enhancing the Java file, a developer needs to remember the following points:

1. You can enhance the auto-generated methods in this class, but you must not remove any auto-generated code.
2. While overriding life cycle methods, you must call the superclass implementation of each life cycle method.
3. While using the pre-compile task feature(described in next section), to overwrite the generated Main Activity file with the modified file, a developer may need to maintain separate Main Activity Java source files for debug and release modes, because they differ in certain functions or parameters.

Android Pre-compile and Post-compile Ant Tasks Support

A developer can perform custom tasks before and after compiling the generated native Android application using **androidprecompiletask.xml** and **androidpostcompiletask.xml**.

For example, a developer can perform the following pre-compile tasks using androidprecompiletask.xml:

1. Overwrite the generated Main Activity Java file with modified file in `<AppID>\src\<packagepath>\` folder .
2. Copy the modified Android application build XML file to the native Android folder.
3. Copy any custom libs, assets, res, and other files/folders into the native Android hierarchy.
4. Append custom properties to project.properties or local.properties, etc.

For example, you can perform the following post-compile tasks using androidpostcompiletask.xml:

1. Automating signing of an application with a release key.
2. Trigger a security scan on a generated APK for security flaws.

3. Trigger automation on a compiled APK file and publish results.
4. Check-in code in GIT repositories.

A developer can copy these XMLs into Kony project workspace base directory with same names (**androidprecompiletask.xml** and **androidpostcompiletask.xml**).

A template **androidprecompiletask.xml** file is available at `<WorkSpace>\temp\<AppID>\build\luaandroid\extres`. This XML is available under `extres` folder after building any sample project. This XML file contains build parameter information useful to perform the tasks explained in the example above.

> ***Note:*** The pre- and post- compile Ant tasks can support integration with external tools such as binary protection tools, static analyzer tools, and so on.

Support for Integrating Android Third-Party Libraries With Kony Project

Since Kony's build system is based on Gradle, integrate third party Android library format (.aar) files into the project by copying .aar files to the required path.

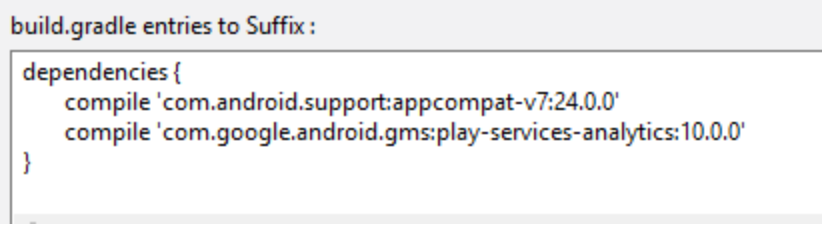
Similarly, third-party java class (.jar) files can also be integrated into the project by copying .jar files to the required path.

- For mobile - `<WorkSpace>\<Application>\resources\customlibs\lib\android`
- For tablet - `<WorkSpace>\<Application>\resources\customlibs\lib\tabrcandroid`

If the library format's .aar or .jar file depends on other libraries, add them to your project.

> ***Important:*** If the included dependencies have an associated order among them, they must be added in same order. For more information on dependency order, see Dependency order.

For Kony Visualizer version 7.3, use the **build.gradle entries to Suffix** Gradle property to add dependencies. For example, if your .aar file or .jar file depends on *Appcompat* and *Play Services Analytics*, you can add the following dependency entries.



For information on Gradle properties, see Set Native App Properties.

For Kony Visualizer version 7.2 and earlier, add dependencies by adding script code to *androidprecompiletask.xml*. For example, the following concatenates the *Appcompat* and *Play Services Analytics* entries to the build.gradle file.

```
< concat destfile = "${app.dir}/build.gradle"
append = "true" > $ {
    line.separator
}
dependencies.compile 'com.android.support:appcompat-v7:24.0.0' <
/concat>
```

```
< concat destfile = "${app.dir}/build.gradle"
append = "true" > $ {
    line.separator
}
dependencies.compile 'com.google.android.gms:play-services-
analytics:10.0.0' < /concat>
```

Generating and Configuring Map API Keys

You will need Maps API Key for Maps to work in the applications you develop. You need to generate the Maps API key and specify it in the properties of the Map widget.

You need to configure the following map keys for your application:

- Application Level Map widget key
- Google Map key for Android (exclusive key for Android platform).
- Bing Map key for Windows

Application Level Map Widget Key

The map widget key needs to be generated based on a domain name. The key generated for a single domain can be used for all sub-domains, URLs on hosts in those domains, and all ports on those hosts. For more information about the domain to be used for generating a map APIs key, see http://code.google.com/apis/maps/faq.html#keysystem.

The generated map key is configured at an application level. This map key is used by all the Map widgets within an application. This map key enables the application to display Google Maps through the Map widgets within the application.

To configure map widget key, do the following:

1. Go to https://code.google.com/apis/console and log in with your Google Account.
    i. Click the Services link from the left-hand menu.
    ii. Activate the Google Maps API v3 service.
    iii. Note the API Key that you generate.
    iv. Click the API Access link from the left-hand menu. Your API key is available on the API Access page, in the Simple API Access section. Maps API applications use the Key for browser apps.

By default, a key can be used on any site. We strongly recommend that you restrict the use of your key to domains that you administer, to prevent use on unauthorized sites. You can specify which domains are allowed to use your API key by clicking the Edit allowed referrers... link for your key.

2. In Kony Visualizer, on the **File** menu (the **Project** menu in *Kony Visualizer*), click **Settings**.
3. On the Application tab, in the Map Widget section, enter the API Key you generated in the **Static map widget key** text box.
4. Enter the API Key you have generated in the **Static map widget key** field under **Map widget**.
5. Click **Finish**. A Build dialog box appears offering to rebuild your app for the platforms that you selected earlier in the **Build Generation** dialog box.

> ***Note:*** When you add a Map widget to a form, the *map widget key* you have entered automatically appears against the *Google Map Key* property of the Map widget. The key generated once is applicable to all the Map widgets within an application.

This configured map widget key is applicable for all platforms except for Android. For Android you need to generate a specific map APIs key and use it in the properties. For more information, see Google Map Key for Android.

Google Map Key for Android

You need the Google Maps API key for Android in order to enable Maps in the applications you develop for Android platform. Maps API keys are linked to specific certificate/package pairs, rather than to users or applications.You only need one key for each certificate, no matter how many users you have for an application. Applications that use the same certificate can use the same API key. For generating maps API key for Android, you need to provide the fingerprint of the signed certificate. For more information see Maps Documentation.

> ***Important:*** Version 1 of the Google Maps Android API as been officially deprecated as of December 3rd, 2012. This means that from March 3rd, 2013 you will no longer be able to request an API key for this version. No new features will be added to Google Maps Android API v1. However, apps using v1 will continue to work on devices. Existing and new developers are encouraged to use Google Maps Android API v2.

To obtain and configure Google Maps API Key v2 for Android, follow these steps:

1. Follow steps for Display your app's certificate information and get an API key from the developers console.
2. Ensure Google Maps Android API is enabled.
3. Navigate to **Properties** > **Application** tab. Under **Map Widget**, add the key generated above in the **Android map widget key 2** field.

   > ***Important:*** Google Play services Revision 13 (Version Name 4.0.30 released on November 2013) and newer versions require Android 2.3 or higher.

4. If you have Google Play Services older to v13, you do not have to perform this step.

   If you have downloaded Google Play Services v13 in Android SDK manager, follow either of the below options:

   - To support Android v2.2 and above devices, do the following:
     i. Download *Google Play services for Froyo* from the Android SDK manager. The Google Play services for Froyo will be available for download only after checking **Obsolete** option in Android SDK manager.
     ii. Rename the folder "`google_play_services_froyo`" to "`google_play_services`", in the Android SDK `path: ..\android-sdk-windows\extras\google\.`

- To use the latest Google Play libraries (Rev 13 and above which supports only Android v2.3 and above devices), do the following:
    i. In Kony Visualizer, On the **Project** menu, click **Settings**.
    ii. Click the **Native** tab, and then click the **Android** sub-tab.
    iii. Scroll down in the dialog box, and under Manifest Properties, click the **Tags** tab.
    iv. In the **Child tag entries under <application> tag** text box, add the following tag:

    <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version" />.

Important Considerations:

- Google Maps V2 does not work in an emulator.
- Multiple Map widgets per Form is not supported for Map V1.(Supported for Map V2)
- If MapV2 key is provided in IDE and if the application is installed on a device with Android Verison 2.2 and above (i.e kony.os.deviceInfo().APILevel >= 8) Google Maps Version 2 would be loaded by default. In rest all cases Google Maps Version 1 would be loaded.
- If the developer uses Google Maps V2, the developer must meet the Google Play attribution requirements as specified at https://developers.google.com/maps/documentation/android/intro#attribution_requirements . The attribution text is available by using *kony.os.deviceInfo ().googleplayServiceSoftwareLicence* in JavaScript projects. This property returns the open source software license information for the Google Play services application, or null when either Google Play services is not available on this device or MapV2 Key is not added in application properties.
- Map does not work on a popup.
- Map preview launches only version 1 map
- Map V2 works for Android 2.2 and above.

- Clickable/Interactive widgets inside a Map callout template will become non-clickable when Android Map V2 version is used. This is the limitation of native Android Map V2 callout. As callouts are displayed as static image snapshot of callout template and only the entire callout is clickable. onSelection event callback is invoked when a callout is clicked.
- You need to update the "Android SDK Tools"," Android SDK Platform-tools" to the latest revision using under SDK Manager (Min revision of Android SDK Tools required is >=15 for Android Google MapV2).
- Android Google MapsV2 requires OpenGL ES version 2 to load.
- You need to download the latest Google Play Services sdk using the SDK Manager. To do so, navigate to **Extras > Google Play Services**. Please refer to the links below for additional information on how Android Google Play and Google MapsV2 work
  - http://developer.android.com/google/play-services/index.html
  - http://developer.android.com/google/play-services/setup.html
  - http://developer.android.com/google/play-services/maps.html

Troubleshooting Android Build Failure with MapV2 Key

If the build fails for MapV2, verify the log if it contains the following messages.

```
"[exec-shell] BUILD FAILED
[exec-shell] ..\sdk\tools\ant\build.xml:601: Invalid file:
..\LibProjects\google-play-services_lib\build.xml"
"
[exec-shell] Error: The project either has no target set or the target
is invalid.
[exec-shell] Please provide a --target to the 'android.bat update'
command.
(or)
google-play-services_lib\build.xml[dependency] Ordered libraries:
```

Follow these steps to resolve the build issue:

1. Verify if the target that is set in the `project.properties` file is already downloaded in the SDK Manager. The `project.properties` file is available at the following location:

   ```
   <SDK-PATH>\extras\google\google_play_services\libproject\google-play-services_lib\project.properties.
   ```

2. Check if the target is available in the SDK Manager by checking the **Obsolete** option, download the corresponding target API.

3. If the target is not available in the SDK Manager, then change the `project.properties` file to point to an appropriate target in the SDK Manager.

   For example, if `project.properties` file contains `target=android-9` and it is not available in the SDK Manager, change the target as `target=android-8` or `target=android-10`, which ever target is available.

Alternative Procedure

1. List the available targets in the system using the following command:

   ```
   android list targets
   ```

2. Set appropriate target from the available targets to `<SDK-PATH>\extras\google\google_play_services\libproject\google-play-services_lib` using the following command:

   ```
   android update lib-project --target <appropriate target ID number from above command> --path <SDK-PATH>\extras\google\google_play_services\libproject\google-play-services_lib/
   ```

Bing Map Key for Windows

To generate a Bing Map Key for Windows platforms, refer http://msdn.microsoft.com/en-us/library/ff428642.aspx.

Navigate to **Application>Properties** and enter the obtained key in the **Bing map widget key field**.

The App Service Event

Using the App Service lifecycle event, you can launch and deep-link Kony native applications from any third-party native application and browser application. This feature allows better integration of Kony applications (all native and hybrid applications) with other native and browser-based applications. The App Service event is available on all platforms.

Characteristics of an App Service Event

An App Service event is characterized by the following:

- An app can have only one App Service event.
- You define the App Service event by creating an action sequence for it in the Properties Editor of a given channel (e.g. Mobile or Tablet).
- The function should always return the form handle that the user will be navigated to, which is the entry point to the target application. The parameters that are part of the URL scheme in the applications are saved in a Lua table and passed to the App Service event.
- The function that you are invoking must be returned manually by adding a snippet. For example, if the function you are invoking is `launchParams`, then the snippet must be defined as:

  ```
  return launchParams(params);
  ```

- The closure function you write in reference to the App Service event is invoked by the platform in a sequence, and this function should also return the form handle.

  > ***Note:*** If both *Post Appinit* and *App Service* events return the form handle, the platforms give more priority to the form returned by the *App Service* event. If both *App Service* and *Postappinit* return a null value, the *showstartupform* event is called.

- Do not invoke `form.show` within the App Service event.

- The App Service event executes every time you access, from your app, a destination outside the app.

Create an App Service Event

To create an App Service event, do the following:

1. In Kony Visualizer, open the project for which you want to set action sequences for the app's lifecycle.
2. From the **Project** tab of the **Project Explorer**, click either **Mobile**, **Tablet**, **Desktop**, or **Watch**. The Visualizer Canvas displays text indicating that you can define app life cycle events from the Properties Editor.
3. On the **App Events** tab of the Properties Editor, click the **Edit** button that corresponds with the App Service event.
4. Using the Action Editor, create the action sequences you want. For more information, see Add Actions.

App Service Event Example

The following is an example of an App Service event.

```
function launchParams(params)
       {
 //The line below displays how an application is launched: Normal,
Push, URL
 kony.print("*************** params is: "+params);
 for (var key in params) {
       kony.print("**********"+params[key]);
       }
 kony.print("*************** launch mode is: " + params.launchmode);
 if(params.launchmode == 1)
{
```

```
  labelText = "You have launched this native app from the device.";
  segData = null;
  frmMain.segParam.removeAll ();
  return frmIntro;
 }else if(params.launchmode == 3){
  labelText = "You have launched this kony native app from a third
party app.";
  //Displays the launchparams. The Launchparams table is a table with
key value pairs specific to the applications needs;

  kony.print("*************** launch params are: " + params.
launchparams);
  if(params.launchparams != null){
   segData = [  ];
   for ( k in params.launchparams ){
    var v = params.launchparams[ kony.decrement( k ) ];
    kony.print("key:" + k + "value:" + v);
    kony.table.insert(segData, { lblkey : "" + k + " :", lblval : ""
+ v } );
   }
   labelText = labelText + "The Parameters are:";
  }else{
   kony.print("*************** launchparams is nil");
  }
  return frmMain;
 }
 return frmIntro;
}
```

The application path in the URL for iPhone and Android is defined by the URL scheme. For more information, see Set App Lifecycle Events and Native Application Properties.

For more information on setting secondary tiles, see the Kony Visualizer API Developer's Guide.

Camera Access in Android Browser

On Android devices, you can access the camera from the Browser widget. The feature is available on devices with Android OS 5.0 and later. For camera access, the Browser widget HTML page must contain the following code.

```
<input   type="file"   accept="image/*"   capture />
```

The capture attribute in the earlier snippet can be of the following types:

```
capture
```

```
capture="true"
```

```
capture="camera"
```

> ***Note:*** For the camera to write the captured image to your application private file system you must expose a file URI using **FileProvider**. For more information on FileProvider, click here.

To enable camera access in the Browser widget, do the following:

- Add the camera permissions
- Add FileProvider support to the application
- Create a fileproviderpaths.xml file and add it to the Native folder.
- Configure the folder to save the captured images. This is an optional step.

Adding Camera Permissions

To add the camera permissions, do the following:

1. In your Kony Visualizer Classic, open the project for which you want to add the camera access to the Browser widget.
2. Navigate to **Project Settings** > **Native** > **Android**. The **Permissions** tab is open by default.

3. In the **The following permissions are set to 'true'** column, ensure that the **CAMERA** option is added. If it is not added, add it.

Adding FileProvider Support to the Application

You must add an entry to the AndroidManifest.xml file in the child tag entries. To add FileProvider support to the application, do the following:

1. In your Kony Visualizer Classic, open the project for which you want to add camera access to the Browser widget.
2. Navigate to **Project Settings** > **Native** > **Android**. The **Permissions** tab is open by default.
3. Under the **Manifest Properties & Gradle Build Entries** section, click **Tags**. The **Tags** tab opens.
4. In the **Child tag entries under <application> tag** text box, enter the following code.

```
< provider
android: name = "android.support.v4.content.FileProvider"
android: authorities = "<application package name >"
android: exported = "false"
android: grantUriPermissions = "true" > < meta - data
android: name = "android.support.FILE_PROVIDER_PATHS"
android: resource = "@xml/fileproviderpaths" / > < /provider>
```

> ***Important:*** In the <provider> entry, replace **<application package name>** with your application package name. For example, if your application package name is **com.orgname.delta**, replace **application package name** with **com.orgname.delta**.

Creating and Adding the fileproviderpaths.xml File

To create and add fileproviderpaths.xml file, do the following:

1. Navigate to your native Android folder in your workspace. **<work space>** > **<Kony app folder> resources** > **mobile** > **native** > **android** > **xml**. This path is for the mobile. For a tablet, the path is **<work space>** > **<Kony app folder> resources** > **tablet** > **native** > **androidtab**.
2. Create a new XML file in the folder and name it **fileproviderpaths.xml**.
3. Open the XML file and paste the following in it.

```
<? xml version = "1.0"
encoding = "utf-8" ?> < paths xmlns: android =
"http://schemas.android.com/apk/res/android" > < files - path
name = "local_file_provider"
path = "/" / > < /paths>
```

Note: This xml file name **fileproviderpaths** (without extension) must match with the value in the manifest <provider> entry for the **android:resource attribute.**

By default, the **path** attribute value is **"/"**. This implies that an image file URI is created under the **files** root folder inside the application private space on the device and a file URI for this file path with write permissions is be shared with the external app camera to save the captured image.

Configuring a New Folder to Save Captured Images

You can configure an alternate folder to save captured images. This is optional. To save the captured images in a different sub folder inside the directory other than the root of the Files folder, do the following:

Modifying an Existing fileproviderpaths.xml File

1. Navigate to the path where you have the **fileproviderpaths.xml** file path.
2. Open the file and change the path attribute to a different value.
   For example, **path="images1"**. Another example is **path="browsercache/images1"**.

> ***Note:*** When you specify a different folder path, ensure that the path value does not begin or end with **"/"**.

Create a New fileproviderattr.xml File

1. For Mobile, navigate to **<work space>** > **<Kony app folder>resources** > **mobile** > **native** > **android** > **values**. For tablet, navigate to **<work space>** > **<Kony app folder>resources** > **tablet** > **native** > **androidtab** > **values**
2. Create a new **fileproviderattr.xml** file with the following content.

```
<? xml version = "1.0"
encoding = "utf-8" ?> < resources > < item type = "string"
name = "file_provider_path" > images1 < /item>
</resources >
```

The path value that you mention in the **<item>** tag of the **fileproviderattr.xml** file must match the **"path"** attribute value in the **fileproviderpaths.xml** file.

Browser Widget Code Changes in JS Layer

Configure the **enableSoftwareRendering** writable property value as **true** for the Browser widget. The property changes the browser rendering from hardware to software.

```
formHome.browser1.enableSoftwareRendering= true
```

In a few devices, after capturing images from the camera, the Browser widget may disappear from the form.

As the camera is a memory hogging application, the Android OS frees most of the resources in the device memory to render the camera application. In such cases, the hardware rendering of the Browser widget may fail in some devices: this may also happen if you open the camera frequently or when the device has low memory. . This is a technical limitation of Android OS.

To handle this issue, you can change the browser rendering from hardware to software that ensures that Browser widget is always displayed properly.

If you configure the **enableSoftwareRendering** property value to **True**, the Browser widget refresh rate decreases. The decrease in the Browser widget refresh rate will impact the refresh rate of animations, GIFs, and videos in the browser.

Appendix F: Accessibility (508 Compliance)

In software application development, 508 Compliance is also referred to as accessibility. For example, people with impaired vision must be able to use software with the help of touch gestures, that is designed to run on a system that has a keyboard. The result of performing a function is read out using the screen reading technology.

> ***Note:*** The screen-reading capabilities of the different assistive technologies may vary and may not produce the same results.

Web Content Access Guidelines (WCAG)

Drafted as a 1986 amendment to the Rehabilitation Act of 1973, Section 508 was originally intended to address the workplace needs of disabled workers in the electronics and information technology fields. That measure was replaced in 1998 by the Federal Electronic and Information Technology Accessibility and Compliance Act, which requires that all electronic and IT products and services offered by federal agencies be accessible to people with disabilities. A “refresh” of Section 508 took effect in 2018 and sought to bring American standards more in line with international accessibility efforts such as the World Wide Web Consortium’s Web Content Access Guidelines 2.0 (WCAG 2.0).

The World Wide Web Consortium (W3C) is an international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards. W3C primarily pursues its mission through the creation of Web standards and guidelines designed to ensure long-term growth for the Web. To learn more about the World Wide Web Consortium, refer About W3C.

W3C’s Web Accessibility Initiative (WAI) brings together individuals and organizations from around the world to develop strategies, guidelines, and resources to help make the Web accessible to people with disabilities. To learn more about Web Accessibility Initiative, refer WAI website.

WCAG Principles

WCAG 2.0 compliance requires the applications to adhere to the following four principles:

- **Perceivable** - Information and user interface components must be presentable to users in ways they can perceive.

  This means that users must be able to perceive the information being presented. (It can't be invisible to all their senses).

- **Operable** - User interface components and navigation must be operable.

  This means that users must be able to operate the interface. (The interface cannot require interaction that a user cannot perform).

- **Understandable** - Information and the operation of user interface must be understandable.

  This means that users must be able to understand the information as well as the operation of the user interface. (The content or operation cannot be beyond their understanding).

- **Robust** - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

  This means that users must be able to access the content as technologies advance. (As technologies and user agents evolve, the content should remain accessible.)

Best Practices

The assistive technology (AT) for iPhone and Android platforms has built-in programs that support accessibility when enabled on devices. Browser-based platforms use the Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA) framework. The WAI-ARIA framework enables you to add attributes to identify features for user interaction, map controls, events, and APIs used in a rich Internet application.

The following table maps different platforms and their assistive technology:

Platform	Assistive Technology
iOS	VoiceOver
Android	TalkBack

Platform	Assistive Technology
SPA	WAI-ARIA

For more information, refer the following links:

- 508 Complaince in Plain English
- W3 Accessibility Guidelines
- Web Accessibility Best Practices

Importance of Accessibility

By integrating accessibility features and services, you can improve the usability of your app, particularly for users with disabilities. People with disabilities depend on accessible apps and services to communicate, learn, and work. By making your app more accessible, you can reach more users.

Accessible design improves overall user experience and satisfaction, especially in a variety of situations, across different devices, and for older users. Accessibility can enhance your brand, drive innovation, and extend your market reach.

There are guidelines, standards, and techniques for web accessibility, such as the Web Content Accessibility Guidelines (WCAG), which is the international standard ISO/IEC 40500. Yet, when designers, developers, and project managers approach accessibility as a checklist to meet these standards, the focus is only on the technical aspects of accessibility. As a result, the human interaction aspect is often lost, and accessibility is not achieved.

Web designers and developers can use usability processes, methods, and techniques, to address the user interface component of accessibility. While the considerations of people with disabilities are not always included in common practices, they can easily be integrated into user experience design.

A key aspect is incorporating **real people** in design:

- Ensuring that everyone involved in web projects understands the basics of how people with disabilities use the Web,

- Involving users with disabilities early and throughout the design process, and
- Involving users in evaluating web accessibility.

Combining accessibility standards and usability processes with real people ensures that web design is technically and functionally usable by people with disabilities. This is referred to as **usable accessibility** or **accessible user experience (UX)**.

Accessibility standards also have an important role in accessible design. For example, understanding the basic Accessibility Principles and using them for developing and evaluating early prototypes helps the development team provide basic accessibility in the earliest stages. Addressing accessibility at later stages becomes increasingly difficult.

Also, usability processes and user involvement alone cannot address all accessibility issues. Even large projects cannot cover the diversity of disabilities, adaptive strategies, and assistive technologies. Accessibility guidelines, standards, and techniques ensure that the wide range of issues are adequately covered.

Web Accessibility

The Web offers the possibility of unprecedented access to information and interaction for many people with disabilities. The Web is an increasingly important resource in many aspects of life: education, employment, government, commerce, health care, recreation, and more. It is essential that the Web be accessible in order to provide equal access and equal opportunity to people with diverse abilities. Access to information and communications technologies, including the Web, is defined as a basic human right in the United Nations Convention on the Rights of Persons with Disabilities (UN CRPD).

WAI-ARIA, the Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. It especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies. Currently, certain functionality used in Web sites is not available to some users with disabilities, especially people who rely on screen readers and people who cannot use a mouse. WAI-ARIA addresses these accessibility challenges, for example, by defining new ways for functionality to be provided to assistive technology. With WAI-ARIA, developers can make advanced Web applications accessible and usable to people with disabilities.

More specifically, WAI-ARIA provides a framework for adding attributes to identify features for user interaction, how they relate to each other, and their current state. WAI-ARIA describes new navigation techniques to mark regions and common Web structures as menus, primary content, secondary content, banner information, and other types of Web structures. For example, with WAI-ARIA, developers can identify regions of pages and enable keyboard users to easily move among regions, rather than having to press Tab many times.

Essential Components of Web Accessibility

It is essential that several different components of web development and interaction work together in order for the web to be accessible to people with disabilities. These components include:

- **Content** - the information in a web page or web application, including:
- **Natural information** such as text, images, and sounds
- **Code or markup** that defines structure, presentation, etc.
- **Web browsers**, **media players**, and other "user agents"
- **Assistive Technology**, in some cases - screen readers, alternative keyboards, switches, scanning software, etc.
- **Users' knowledge**, experiences, and in some cases, adaptive strategies using the web
- **Developers** - designers, coders, authors, etc., including developers with disabilities and users who contribute content
- **Authoring tools** - software that creates websites
- **Evaluation tools** - web accessibility evaluation tools, HTML validators, CSS validators, etc.

When accessibility features are effectively implemented in one component, the other components are more likely to implement them.

Evaluating Web Accessibility

While developing or redesigning a website, evaluate accessibility issues early and throughout the development process, when it is easier to address them. Simple steps, such as changing settings in a browser, can help you evaluate some aspects of accessibility. Comprehensive evaluation to determine if a website meets all accessibility guidelines takes more effort.

There are evaluation tools that help with evaluation. However, no tool alone can determine if a site meets accessibility guidelines. Knowledgeable human evaluation is required to determine if a site is accessible. Refer Evaluation tools for a list of Web Accessibility Evaluation tools approved by WAI.

If you want to get a general sense of how a web page addresses a few accessibility issues, refer Easy Checks. The Easy Checks cover just a few accessibility issues and are designed to be quick and easy, rather than definitive. A web page could seem to pass these checks, yet still have significant accessibility barriers. More robust assessment is needed to evaluate accessibility comprehensively.

Website Accessibility Conformance Evaluation Methodology (WCAG-EM) is an approach for determining how well a website conforms to Web Content Accessibility Guidelines (WCAG).

> ***Note:*** WCAG-EM is a supporting resource for the WCAG standard; it does not define additional WCAG requirements.

Enable Accessibility in Visualizer

Kony Visualizer provides accessibility features to support all the above principles for support to assistive technology, for its widgets and components to be compliant with WCAG 2.0. WCAG 2.0 compliance is not a framework compliance but it applies to the applications designed & developed using the framework. The framework enables (through its features) app design & development to follow this standard. The application design and development requirements would need to adhere to the WCAG 2.0 guidelines while developing the application. This along with the features that the platform exposes would ensure compliance to WCAG 2.0.

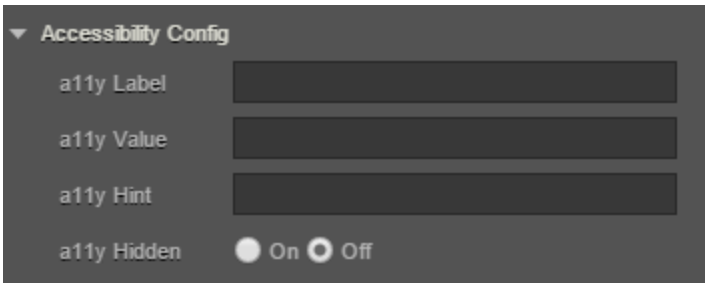
The built-in assistive technology in the iOS and Android platforms reads some basic widgets of Kony Visualizer, such as Button, Label, and TextBox. The assistive technology in iOS and Android platforms read the information differently on other widgets.

Every application that is created using Kony Visualizer is accessible to the built-in assistive technology of the iOS and Android platforms. The way the assistive technology interprets the widget information is left to its individual capability. Developers can enhance the behaviour of assistive technology with the configuration property (explained in the next page) available for each accessibility supported widget.

Configuring Accessibility

To define the Accessibility Config property for a widget from Kony Visualizer, follow these steps:

1. From the Default Library in Kony Visualizer, drag a widget and drop it onto the canvas. For example, a Button widget.
2. Select the Button widget and navigate to the Look tab of the Properties pane.
3. Locate the Accessibility Config section, and enter the following values in the respective fields:
    - **a11yLabel**: Specifies an alternate text to identify the widget. In general, the label must be the text that is displayed on the screen.
    - **a11yValue**: Specifies the current state/value associated with the widget so that the user can perform an action. For example, a checkbox is in selected state or unselected state.
    - **a11yHint**: Specifies the descriptive text that explains the action about the widget.
    - **a11yHidden**: Specifies if the widget must be ignored by assistive technology.



> ***Note:*** The **Accessibility Config** section appears on the **Look** tab only when the **Accessibility Config** check box is selected in the **Project Settings > Application**.

For Camera widget, when the capture mode is set to video, you will get additional options to record the video. The below keys are used to configure accessibility for the additional options:

- **accessibilityConfigCaptureControl**: Provides accessibility support to video capture button.
- **accessibilityConfigTimerControl**: Provides accessibility support to video timer button.
- **accessibilityConfigSettingsControl**: Provides accessibility support to video settings button.
- **accessibilityConfigCancelControl**: Provides accessibility support to video cancel button.
- **accessibilityConfigVideoStartButton**: Provides accessibility support to video start button.
- **accessibilityConfigVideoStopButton**: Provides accessibility support to video stop button.

accessibilityConfig Property

In order to support accessibility in Visualizer, a common property, accessibilityConfig is available in all widgets. This property enables you to control accessibility behavior and alternative text for the widget.

Syntax

```
accessibilityConfig = {

    "a11yLabel" = "string value",
    "a11yValue" = "string value",
    "a11yHint" = "string value",
    "a11yHidden" = true/false
}
```

Type

Object

Read/Write

Read + Write

Remarks

The accessibilityConfig property is enabled for all the widgets which are supported under the Flex Layout.

The accessibilityConfig property is a JavaScript object which can contain the following key-value pairs.

Key	Type	Description	ARIA Equivalent
a11yLabel	String	This is an optional parameter. Specifies alternate text to identify the widget. Generally the label should be the text that is displayed on the screen.	For all widgets, this parameter maps to the **aria-label** property of ARIA in HTML. ***Note:*** For the Image widget, this parameter maps to the **alt** attribute of ARIA in HTML.
a11yValue	String	This is an optional parameter. Specifies the descriptive text that explains the action associated with the widget. On the Android platform, the text specified for a11yValue is prefixed to the a11yHint.	This parameter is similar to the a11yLabel parameter. If the a11yValue is defined, the value of a11yValue is appended to the value of a11yLabel. These values are separated by a space.
a11yHint	String	This is an optional parameter. Specifies the descriptive text that explains the action associated with the widget. On the Android platform, the text specified for a11yValue is prefixed to the a11yHint.	For all widgets, this parameter maps to the **aria-describedby** property of ARIA in HTML.

Key	Type	Description	ARIA Equivalent
a11yHidden	Boolean	This is an optional parameter. Specifies if the widget should be ignored by assistive technology. The default option is set to *false*. This option is supported on iOS 5.0 and above, Android 4.1 and above, and SPA	For all widgets, this parameter maps to the **aria-hidden** property of ARIA in HTML.
a11yARIA	Object	This is an optional parameter. For each widget, the key and value provided in this object are added as the attribute and value of the HTML tags respectively.	This parameter is only available on the DesktopWeb platform.

Android limitations

- If the results of the concatenation of a11y fields result in an empty string, then accessibilityConfig is ignored and the text that is on widget is read out.
- The soft keypad does not gain accessibility focus during the right/left swipe gesture when the keypad appears.

SPA/Desktop Web limitations

- The behavior of accessibility depends on the Web browser, Web browser version, Voice Over Assistant, and Voice Over Assistant version.
- Currently SPA/Desktop web applications support only a few ARIA tags. To achieve more accessibility features, use the attribute a11yARIA. The corresponding tags will be added to the DOM as per these configurations.

Example

This example uses the button widget, but the principle remains the same for all widgets that have an accessibilityConfig property.

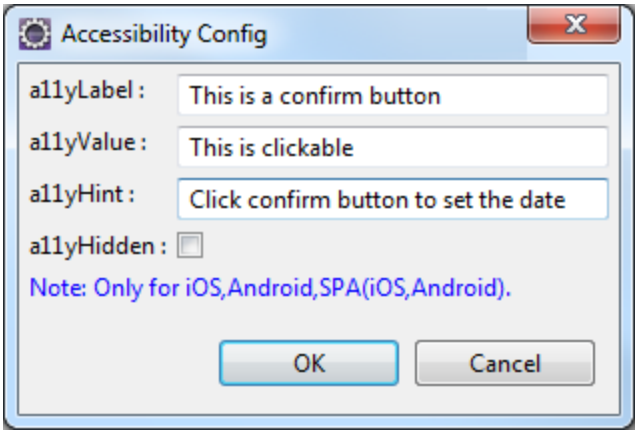
```
//This is a generic property that is applicable for various widgets.
//Here, we have shown how to use the accessibilityConfig Property for
button widget.
/*You need to make a corresponding use of the accessibilityConfig
property for other applicable widgets.*/

Form1.myButton.accessibilityConfig = {
    "a11yLabel": "Label",
    "a11yValue": "Value",
    "a11yHint": "Hint"
     a11yARIA: {
        "tabindex" < all lowercase > : integer // no floating/decimal
numbers,
        "role": 'Prescribed string value as per wcag guidelines',
//not encouraged
        "aria-invalid": true / false,
        "aria-keyshortcuts": "Shift+Space"
    }
};
```

Platform Availability

- Available in the IDE
- iOS
- Android
- SPA
- Desktop Web

Widget Navigation

Here is a sample representation of accessibility on iPhone and Android devices. For example, you have a Confirm button on form frm1, and the accessibilityConfig is defined as below:



Note: In the above snapshot the highlighted text is role, generated by native platforms. The iPhone appended the text ***button*** to the value, and Android appended the text ***button*** to the hint automatically. Kony Visualizer has no control on this behavior. Developers should test the text that is provided for accessibilityConfig.

Note: To focus on widgets in Responsive Web and SPA applications, you can use the Tab key on the keyboard. However, while using the Tab key to focus on non-input widgets (for example, Button widget) with the accessibility On, the focus skin does not always display. To get the focus skin for non-input widgets, use the custom CSS. Here is an example of a custom CSS for a widget:

```
Widget:focus:not([kdisabled='true'])
{ border: 0px solid #ffffff;
 font: normal normal 24.0px Helvetica;
color: #ffffff;
 background: #00ff00; }
```

Container Widgets

Following are the platforms behaviors for the container widgets when the accessibility feature is enabled.

- FlexForm
- Form (Deprecated)
- HBox (Deprecated)
- VBox (Deprecated)
- ScrollBox (Deprecated)
- Popup (Deprecated)

FlexForm

Keyboard/Gesture-based Interactions	• Tab key or equivalent touch gesture moves the focus to the first focusable child widget of the form. • Multiple tabs move the focus to the interactive child widgets of the form. • The title of the form is accessible in the platforms that support native form widget titleBar property using the tab key or equivalent touch gesture along tab order.
Default Behavior	• The a11yLabel overrides the text of the title property. • The a11yValue, a11yHint, and a11yHidden fields are not applicable to form and are ignored. • accessibilityConfig property is supported in iPhone, Android, and SPA-iPhone platforms.

Limitations	iOS: • When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form. • The iOS VoiceOver, focus the first accessible widget available on the form. Android: • onTap gesture on a form, when there are no focusable widgets, the assistive technology reads all non-focusable widgets text available in the form. • In Android OS versions less than 4.2, form does not scroll, although it has content to scroll. You have to enable an option in Android OS versions 4.2 and above in system accessibility settings to auto scroll the content on swipe gesture. • Accessibility capability of the ActionBar is left to the device default behavior. SPA: • SPA-iPhone: When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form. • SPA-Android: accessibilityconfig is not supported

Form (Deprecated)

Keyboard/Gesture-based Interactions	• Tab key or equivalent touch gesture moves the focus to the first focusable child widget of the form. • Multiple tabs move the focus to the interactive child widgets of the form. • The title of the form is accessible in the platforms that support native form widget titleBar property using the tab key or equivalent touch gesture along tab order.
Default Behavior	• The a11yLabel overrides the text of the title property. • The a11yValue, a11yHint, and a11yHidden fields are not applicable to form and are ignored. • accessibilityConfig property is supported in iPhone, Android, and SPA-iPhone platforms.

Limitations	iOS: • When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form. • The iOS VoiceOver, focus the first accessible widget available on the form. Android: • onTap gesture on a form, when there are no focusable widgets, the assistive technology reads all non-focusable widgets text available in the form. • In Android OS versions less than 4.2, form does not scroll, although it has content to scroll. You have to enable an option in Android OS versions 4.2 and above in system accessibility settings to auto scroll the content on swipe gesture. • Accessibility capability of the ActionBar is left to the device default behavior. SPA: • SPA-iPhone: When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form. • SPA-Android: accessibilityconfig is not supported

HBox (Deprecated)

Keyboard/Gesture based Interactions	• Tab key or equivalent touch gesture moves the focus along the tab order when: a. Box is clickable b. accessibilityConfig is defined. • With a focus on the HBox, press Spacebar or Enter or equivalent gesture action to select the HBox when it is clickable. • Multiple tabs or navigation keys help in navigating focus to the child widgets that are actionable.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	iOS: • If the accessibilityConfig is set for an HBox, then the focus never goes to its child widgets and other widgets within the HBox are not accessible to the user. Android: None SPA: • SPA-iPhone: If the accessibilityConfig is set for an HBox, then the focus never goes to its child widgets, and other widgets within the HBox are not accessible to the user. • SPA-Android: If the accessibilityConfig is set for an HBox, then widgets within the HBox are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.

VBox (Deprecated)

Keyboard/Gesture based Interactions	• Tab key or equivalent touch gesture moves the focus along the tab order when: a. Box is clickable. b. accessibilityConfig is defined. • With a focus on the VBox, press Space or Enter or equivalent gesture action to select the VBox when it is clickable. • Multiple tabs or navigation keys help in navigating focus to the child widgets that are actionable.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	iOS: • If the accessibilityConfig is set for a VBox, then the focus never goes to its child widgets and other widgets within the VBox. Android: None SPA: • SPA-iPhone: If the accessibilityConfig is set for a VBox, then the focus will never go to its child widgets, and other widgets within the VBox are not accessible to the user. • SPA-Android: If the accessibilityConfig is set for a VBox, then widgets within the VBox are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.

ScrollBox (Deprecated)

Keyboard/Gesture based Interactions	• Tab key or equivalent touch gesture moves the focus along the tab order when accessibilityConfig is defined. • Multiple tabs or navigation keys help in navigating focus to the child widgets that are actionable. • Page Up / Page Down or equivalent key/gesture allows you to scroll the content of the ScrollBox.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations	iOS: • If the accessibilityConfig is set for a ScrollBox, then the focus never goes to its child widgets, and other widgets within the ScrollBox are not accessible to the user. Android: In Android OS versions less than 4.2, the form does not scroll although it has content to scroll. It depends on the capability of the built-in Accessibility service. You have to enable an option in Android OS versions 4.2 and above in the system accessibility settings to auto-scroll the content on swipe gesture. Similar behavior is observed with native applications as well. When the scrollDirection property is set to SCROLLBOX_SCROLL_BOTH, the behavior is undefined. SPA: When a user scrolls through the Scrollbox, it does not scroll and the widgets are not displayed in the view port, even if you set accessibility. This is due to the inability of the browsers to detect the touch gestures. But the widgets within Scrollbox are navigated and accessibility of the widget is read out. • SPA-iPhone: If the accessibilityConfig is set for a ScrollBox, then widgets within the ScrollBox are not accessible to the user. • SPA-Android: If the accessibilityConfig is set for a ScrollBox, then widgets within the ScrollBox are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.

Popup (Deprecated)

Keyboard/Gesture based Interactions	• Tab key or equivalent touch gesture moves the focus to the first focusable child widget of the popup. • Multiple tabs move the focus to the interactive child widgets of the popup.

Default Behavior	• The a11yLabel overrides the text of the title property. • The a11yValue, a11yHint, and a11yHidden fields are not applicable to popup and are ignored. • accessibilityConfig property is supported in iPhone, Android, and SPA-iPhone platforms.
Limitations	iOS: • When the VoiceOver focus reaches the bottom of the popup, it does not cycle to the top of the popup again, with a right swipe gesture. Similarly with a left swipe gesture, the focus does not cycle to the bottom of the popup when you have reached the top of the popup. Android: • When there are no focusable widgets in a popup, the assistive technology reads all non-focusable widgets text available in the popup. • In Android OS versions less than 4.3, the popup does not scroll although it has more content to scroll. It is the capability of the built-in Accessibility service (TalkBack) that is lacking in versions less than 4.3 OS versions. You have to enable an option in Accessibility settings in 4.3 and 4.4 Android OS versions to auto-scroll the content on a swipe gesture. SPA: • SPA-iPhone: When the VoiceOver focus reaches the bottom of the popup, it does not cycle to the top of the popup again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the popup when you have reached the top of the popup. • SPA-Android: accessibilityConfig is not supported.

Basic Widgets

Below are the platforms behaviors of the basic widgets when accessibility feature is enabled.

- Button
- Calendar
- CheckBox
- ComboBox
- Image
- Label
- Line
- Link
- ListBox
- RadioButton
- RichText
- Slider
- TextArea
- TextBox

Button

Keyboard/Gesture based Interactions	• With a focus on the Button, press the Spacebar or Enter key or equivalent gesture action to select the Button. • Single finger double tap to execute the action. • Accessible by the tab key or equivalent touch gesture along tab order.

Default Behavior	accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.
Limitations	iOS: None Android: None SPA: • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.

Calendar

Description	A Calendar widget accepts dates from the user. Following are the view types that support accessibility in respective platforms: • CALENDAR_VIEW_TYPE_DEFAULT (Android) • CALENDAR_VIEW_TYPE_WHEEL_POPUP (iPhone) • CALENDAR_VIEW_TYPE_ GRID_POPUP (iPhone and SPA)

Keyboard/ Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along tab order. • Before the V8 SP4 release, using a tab key to focus on the calendar icon in the Calendar widget was not supported. Post the Visualizer V8 SP4 release, when you use the Tab key, the focus moves to the calendar Input box. When you use the Tab key again, focus moves to the calendar icon. • For SPA and Desktop Web platforms, when the focus is on the Calendar, press the Spacebar or Enter key or equivalent gesture action to launch the date selector. If you press the Tab key again, the date selector is closed and the focus is moved back to the calendar icon. • You can reach to each available day, month, and year in a calendar through one/ some of the keys or touch gestures.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, Desktop Web platforms.
Limitations	• a11yValue is not applicable. • It is recommended to provide accessibility text to the assistive technology to read the date when the tab focus/gesture makes a selection. • accessibilityConfig is not supported for the viewTypes that are not focused as a whole.

CheckBox

Keyboard/Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along tab order. • Multiple tabs or navigation keys help in navigating the focus to each check button in the group. • With a focus on the CheckBox, press the Spacebar or Enter key or equivalent gesture to change the selection state of the focused check button.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.

<table>
<tr>
<td>Limitations</td>
<td>iOS: Following are the limitations of iOS platform based on the selected viewType:
<table>
<tr><th>viewType</th><th colspan="2">accessibilityConfig</th></tr>
<tr><th></th><th>Widget Level</th><th>Items within widget</th></tr>
<tr><td>CHECKBOX_VIEW_TYPE_LISTVIEW</td><td>Respected</td><td>Ignored</td></tr>
<tr><td>CHECKBOX_VIEW_TYPE_TOGGLEVIEW</td><td>Ignored</td><td>Respected</td></tr>
<tr><td>CHECKBOX_VIEW_TYPE_ONSCREENWHEEL</td><td>Ignored</td><td>Ignored</td></tr>
<tr><td>CHECKBOX_VIEW_TYPE_TABLEVIEW</td><td>Ignored</td><td>Respected</td></tr>
</table>
* accessibilityConfig is ignored when set through masterData or masterDataMap to the items within the widget that pops up as pickerview from the bottom.

Android: None

SPA:

SPA-iPhone: accessibilityConfig for CheckBox as a whole is not respected, but the items within the widget are respected. The CheckBox state and the item text gets the focus separately.
SPA-Android: accessibilityConfig for CheckBox as a whole is not respected, but the items within the widget are respected. The CheckBox state and the item text get the focus separately. The option a11yHint is not supported.

</td>
</tr>
<tr>
<td>Example code</td>
<td>Click here to view the sample code</td>
</tr>
</table>

ComboBox

Keyboard/Gesture based Interactions	• Tab key or equivalent touch gesture along tab order. • With a focus on the ComboBox, press the Spacebar or Enter key or equivalent gesture action to open the drop-down list. • With drop-down list in an expanded state, press the Spacebar or Enter key or equivalent gesture to select the focused item. • Multiple tabs or navigation keys help in navigating the focus to each item in the ComboBox.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.

Limitations

iOS: Following are the limitations of iOS platform based on the selected viewType:

viewType	accessibilityConfig	
	Widget Level	Items Within Widget
COMBOBOX_VIEW_TYPE_LISTVIEW	Respected	Ignored *
COMBOBOX_VIEW_TYPE_TABLEVIEW	Ignored	Respected
COMBOBOX_VIEW_TYPE_TOGGLEVIEW	Ignored	Respected
COMBOBOX_VIEW_TYPE_ONSCREENWHEEL	Ignored	Ignored

* accessibilityConfig is ignored when set through *masterData* or *masterDataMap* to the items within the widget that pops up as pickerview from the bottom.

Android: If the accessibilityConfig is set, it will override the selected item.

SPA:

- SPA-iPhone: The ComboBox widget is mapped to the HTML ComboBox, and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ComboBox widget.
- SPA-Android:The ComboBox widget is mapped to the HTML ComboBox, and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ComboBox widget.

Example code	Click here to view the sample code

Image

Keyboard/Gesture based Interactions	Tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.
Limitations	On all platforms, except SPA, if the accessibility is not configured, the image widget is not accessible to the user. SPA-Android: The option a11yHint is not supported.

Label

Description	A Label widget is used to display non-editable text to the user.
Keyboard/Gesture based Interactions	Tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA-Android platforms.
Limitations	iOS: None Android: None SPA: • SPA-iPhone: accessibilityConfig property is not supported. • SPA-Android:The option a11yHint is not supported

Line

Keyboard/Gesture based Interactions	Not accessible by the tab key or equivalent touch gesture.
Default Behavior	accessibilityConfig property is not supported.
Limitations	None

Link

Keyboard/Gesture based Interactions	• With a focus on the link, press the Spacebar or Enter key or equivalent gesture action to select the link. • Single finger double tap to execute the action. • Tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	iOS: None Android: None SPA: • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.

ListBox

Keyboard/Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the ListBox, press the Spacebar or Enter key or equivalent gesture to open the drop-down list. • With drop-down list in an expanded state, press the Spacebar or Enter key or equivalent gesture to select the focused item. • Multiple tabs or navigation keys help in navigating the focus to each item in the ListBox.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.

<table>
<tr>
<td>Limitations</td>
<td>iOS: Following are the limitations of iOS platform based on the selected viewType:
<table>
<tr><th>viewType</th><th colspan="2">accessibilityConfig</th></tr>
<tr><th></th><th>Widget Level</th><th>Items Within Widget</th></tr>
<tr><td>LISTBOX_VIEW_TYPE_LISTVIEW</td><td>Supported</td><td>Ignored *</td></tr>
<tr><td>LISTBOX_VIEW_TYPE_TABLEVIEW</td><td>Ignored</td><td>Supported</td></tr>
<tr><td>LISTBOX_VIEW_TYPE_TOGGLEVIEW</td><td>Ignored</td><td>Supported</td></tr>
<tr><td>LISTBOX_VIEW_TYPE_ONSCREENWHEEL</td><td>Ignored</td><td>Ignored</td></tr>
</table>
* accessibilityConfig is ignored when set through masterData or masterDataMap to the items within the widget that pops up as pickerview from the bottom.

Android: If the accessibilityConfig is set, it will override the selected item.

SPA:

SPA-iPhone: The ListBox widget is mapped to the HTML ListBox, and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ListBox widget.
SPA-Android:The ListBox widget is mapped to the HTML ListBox and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ListBox widget. The option a11yHint is not supported.

</td>
</tr>
<tr>
<td>Example code</td>
<td>Click here to view the sample code</td>
</tr>
</table>

RadioButton

Keyboard/Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the RadioButton, press the Spacebar or Enter key or equivalent gesture to change the selection state of the focused item. • Multiple tabs or navigation keys help in navigating the focus to each item in the RadioButton.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.

<table>
<tr>
<td>Limitations</td>
<td>iOS: Following are the limitations of the iOS platform based on the selected viewType:
<table>
<tr><th>viewType</th><th colspan="2">accessibilityConfig</th></tr>
<tr><th></th><th>Widget Level</th><th>Items Within Widget</th></tr>
<tr><td>RADIOGROUP_VIEW_TYPE_LISTVIEW</td><td>Supported</td><td>Ignored *</td></tr>
<tr><td>RADIOGROUP_VIEW_TYPE_TABLEVIEW</td><td>Ignored</td><td>Supported</td></tr>
<tr><td>RADIOGROUP_VIEW_TYPE_TOGGLEVIEW</td><td>Ignored</td><td>Supported</td></tr>
<tr><td>RADIOGROUP_VIEW_TYPE_ONSCREENWHEEL</td><td>Ignored</td><td>Ignored</td></tr>
</table>
* accessibilityConfig is ignored when set through masterData or masterDataMap to the items within the widget that pops up as pickerview from the bottom is ignored.

Android: None

SPA:

SPA-iPhone: accessibilityConfig for RadioButton as a whole is not supported, but the items within the widget are supported.
SPA-Android: accessibilityConfig for RadioButton as a whole is not supported, but the items within the widget are supported. The option a11yHint is not supported.

</td>
</tr>
<tr>
<td>Example code</td>
<td>Click here to view the sample code</td>
</tr>
</table>

RichText

Keyboard/Gesture based Interactions	Accessible by the tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.
Limitations	On all platforms, links inside a RichText widget are not accessible. SPA-Android: The option a11yHint is not supported.

Slider

Keyboard/Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the Slider, press the Right / Up key or equivalent gesture to increase the value of the slider. Press the Left / Down key or equivalent gesture to decrease the value of the slider.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.
Limitations	Android: Android OS cannot change the slider value when accessibility is set. SPA: • SPA-iPhone: Browsers cannot change the slider value when accessibility is set. • SPA-Android: Browsers cannot change the slider value when accessibility is set.

TextArea

Keyboard/Gesture based Interactions	• With a focus on the TextArea, press the Spacebar or equivalent gesture to open the soft keypad for touch devices. • Single finger double tap to execute the action. • Accessible by the tab key or equivalent touch gesture along tab order. • Soft keypad gains focus on explicit touch on the soft keypad.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.
Limitations	iOS: None Android: When the accessibilityConfig is defined for placeholder or entered text, then behavior is left to the device. SPA: • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.

TextBox

Keyboard/Gesture based Interactions	• With a focus on the TextBox, press the Spacebar or equivalent gesture to open the soft keypad for touch devices. • Single finger double tap to execute the action. • Accessible by the tab key or equivalent touch gesture along tab order. • Soft keypad gains focus on explicit touch on the soft keypad.

Default Behavior	accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms. ***Note:*** To configure the clear text button, use the property accessibilityConfigForClearButton and the keys are same as that of accessibilityConfig. This is applicable to iOS platform only
Limitations	iOS: None Android: When the accessibilityConfig is defined for placeholder or entered text, then behavior is left to the device. SPA: • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.

Advanced Widgets

Below are the behaviors of the advanced widgets when the accessibility feature is enabled.

- Alert
- Camera
- Hz Image Strip
- PickerView
- SegmentedUI - TABLEVIEW
- SegmentedUI - PAGEVIEW
- Switch

Alert

Description	An Alert is a dialog displayed to show an alert message.
Keyboard/Gesture based Interactions	• Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture to navigate and focus on the buttons and messages of the Alert dialog box. • With a focus on the Alert button, press the Enter or Spacebar or equivalent gesture to select the button. • By default, Alerts should gain focus as Alert displays.
Default Behavior	accessibilityConfig is not supported.
Limitations	On all platforms, the buttons within the Alert dialog are not configurable.

Camera

Keyboard/Gesture based Interactions	• Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the camera, press the Enter or Spacebar or equivalent gesture to launch the camera.
Default Behavior	• accessibilityConfig property is supported in iPhone and Android platforms.
Limitations	accessibilityConfig property is not supported in SPA platform

Hz Image Strip

Keyboard/Gesture based Interactions	• Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture along tab order. • On multiple tabs or equivalent touch gesture, move the focus to the images where accessibility is configured and visible on the screen. • Images that are not visible are scrolled automatically to a visible region on a tab or equivalent gesture.
Default Behavior	• accessibilityConfig property is supported in iPhone, Android, SPA, and Desktop Web platforms.

Limitations	If the entire image strip widget is not focused as a whole, then the accessibilityConfig is not respected in any of the platform. The accessibilityConfig is supported only when the viewType is set to HORIZONTAL_IMAGESTRIP_VIEW_TYPE_STRIPVIEW. iOS:For the viewType when set to HORIZONTAL_IMAGESTRIP_VIEW_TYPE_STRIPVIEW, accessibility is ignored. But the accessibility configured for each image is supported. Accessibility is not available for all other viewtypes. Android: In Android OS versions less than 4.2, Horizontal Image Strip does not scroll though it has content to scroll. It depends on the capability of the built-in accessibility service. You must enable an option in Android OS versions greater than equal to 4.2 in system accessibility settings to auto scroll the content on swipe gesture. You will observe similar behavior with native applications as well. SPA: • SPA-iPhone: If the accessibilityConfig is set for a Horizontal Image Strip, then widgets within the Horizontal Image Strip are not accessible to the user. • SPA-Android: If the accessibilityConfig is set for a Horizontal Image Strip, then widgets within the Horizontal Image Strip are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.
Example code	sample code

PickerView

Keyboard/Gesture based Interactions	• Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture along tab order. • Every column in the PickerView is reachable by the tab key or equivalent touch gesture. • With a focus on the PickerView, press the Right / Up key or Left / Down key or equivalent gesture to allow navigation between items in the focused column. • With a focus on the PickerView, press the Enter or Spacebar or equivalent gesture to select the focused item in the PickerView.
Default Behavior	• accessibilityConfig property is supported in Android platforms.
Limitations	iOS: accessibilityConfig is not supported Android: In Android OS versions less than 4.2, SegmentedUI does not scroll though it has content to scroll. It depends on the capability of the built-in accessibility service. You must enable an option in Android OS versions greater than equal to 4.2 in system accessibility settings to auto-scroll the content on a swipe gesture. You will observe similar behavior with native applications as well.
Example code	sample code

SegmentedUI - TABLEVIEW

Description	A SegmentedUI is a container widget to display multiple rows of information in vertical order.

Keyboard/Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along the tab order moves the focus to the first row. • If the first row is a section header, then the subsequent tabs move the focus to the interactive child widgets. If there are no child widgets or interactive widgets, or all child widgets are reached, the tab moves the focus to the next row until it reaches the last visible row. • In SINGLE_SELECT_MODE or MULTI_SELECT_MODE, as the row gets the focus through the tab, underlying accessibility technology conveys the user as either selected/unselected.
Default Behavior	• At widget level, accessibilityConfig property is not supported in iPhone, Android, and SPA platforms because it is not focusable completely. Accessibility is supported only for the individual rows because they are focused completely. • accessibilityConfig is applied to the row template, and then accessibility is applied to each row, unless overridden by the row data. • Row template's accessibility configurations can be modified before setting the row data to the SegmentedUI and should not be modified after data are set.

Limitations	iOS: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. Android: In Android OS versions less than 4.2, SegmentedUI does not scroll though it has content to scroll. You must enable an option in Android OS versions 4.2 and above system accessibility settings to auto-scroll the content on a swipe gesture. You will observe similar behavior with native applications. SPA: • SPA-iPhone: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. • SPA-Android: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. But when touched explicitly the widgets gain focus with a swipe gesture. The option a11yHint is not supported.
Example code	sample code

SegmentedUI - PAGEVIEW

Description	A SegmentedUI is a container widget to display multiple rows of information in horizontal layout with a single row appearing on the widget.

Keyboard/Gesture based Interactions	• Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture along tab order. • If the first row is a section header, then the subsequent tabs move the focus to the interactive child widgets. If there are no child widgets or interactive widgets, or all child widgets are reached, the tab moves the focus out of the widget. • If there are page indicators at the bottom of the PAGEVIEW and the page indicators are interactive, the tab focus each page and pass the index of the total page information to the assistive technology. • In SINGLE_SELECT_MODE or MULTI_SELECT_MODE, as the row gets the focus through the tab, underlying assistive technology conveys the user as either selected/unselected.
Default Behavior	• At widget level accessibilityConfig property is not respected. It is respected only for the page level in iPhone, Android, and SPA platforms. • accessibilityConfig applied to the row template and its internal widgets are applied to each row, unless overridden by the row data. • Row template's accessibility configurations can be modified before setting the row data to the SegmentedUI and should not be modified after data are set.

Limitations	On all platforms, accessibilityConfig is not supported for page indicators. iOS: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. Android: In Android OS versions less than 4.2, SegmentedUI does not scroll though it has content to scroll. It depends on the capability of the built-in accessibility service. You must enable an option in Android OS versions 4.2 and above, in system accessibility settings to auto-scroll the content on a swipe gesture. You will observe similar behavior with native applications as well. SPA: The event onRowClick is fired when any child widget is explicitly selected or clicked. • SPA-iPhone: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. • SPA-Android: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. But when touched explicitly the widget's gain focus. The option a11yHint is not supported.

Switch

Keyboard/Gesture based Interactions	• Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the Switch, press the Enter key or Spacebar or equivalent gesture to toggle the state and initiate the action.
Default Behavior	• accessibilityConfig property is supported in iPhone and SPA platforms.

Limitations	iOS: The option a11yValue is not supported. SPA: • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.

Platform-specific Accessibility features

This topic provides information about various accessibility compliance features and guidelines provided by different OS providers.

An accessible app supports personalization by design and gives everyone a great user experience, regardless of their capabilities or how they use their devices. This allows individuals with a range of disabilities, such as visual, hearing, physical, or speech impairments, to enhance their ability to access and interact with web pages and apps.

iOS

On the iOS platform, Apple provides Accessibility features that are broadly classified into four categories: Vision, Physical and Motor, Hearing, and General.

The **Vision** section contains VoiceOver, Zoom, Display & Text Size and Audio Descriptions configurations. The **Physical and Motor** section contains options such as Face ID, Switch Control, Apple TV Remote and Keyboard. The **Hearing** section provides configuration options for Hearing Devices, Audio/Visual, and Subtitles & Captioning. The **General** category contains Guided Access, Siri, and Accessibility Shortcut options.

For detailed information on all the features supported, refer Accessibility Features on iPhone. For information on how to make your app accessible, refer Human interface Guidelines.

Android

The accessibility features available on the Android platform vary depending on the OEM (original equipment manufacturer).

On the Android platform, options that a user can configure for Accessibility include settings for Screen Readers, Display options, Audio and On-screen Text options, Interaction controls, and Experimental. The **Screen Readers** section contains the configuration for the Text-to-speech output. The **Display** section contains the settings for Font Size, Display size, Magnification, Color correction, Color inversion. The **Interaction Controls** section contains the settings for Dwell timing, Power button ends call, Auto-rotate screen, Touch and hold delay, and Vibration. The **Audio and On-screen text** section has settings for Mono Audio and Captions. The **Experimental** section has the option to enable high contrast text.

For more information on the features supported on the Android platform and how to make your app accessible, refer Development resources.

Gestures

When a device enables assistive technology, visually impaired users typically navigate through the UI controls such as tab/enter/arrow keys/page up/down keys. On various touch-only devices, a few of these key actions are mapped to touchscreen finger gestures.

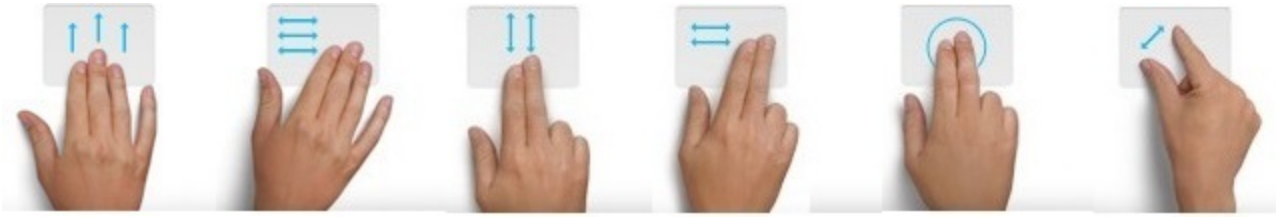
The following table depicts how keyboard-based navigation keys are mapped to gestures on mobile platforms:

Keyboard based devices (desktop)	Purpose	Android touch (Android 4.1 and higher) (talkback)	iOS touch
Tab	To move focus in forward direction	One finger right/down flick gesture	One finger right flick gesture
Shift+Tab	To move focus in reverse direction	One finger left/up flick gesture	One finger left flick gesture
Enter /Space	To take action on the focused widget	One finger double tap	One finger double tap

Keyboard based devices (desktop)	Purpose	Android touch (Android 4.1 and higher) (talkback)	iOS touch
Right Arrow/Up Arrow	To increase the value selection on specific widgets like slider/picker		One finger up flick gesture
Left Arrow/Up Arrow	To decrease the value selection on specific widgets like slider/picker		One finger up flick gesture
Page Up	To scroll up/left of the content in a scroll container.	Two/three finger up/left flick gesture	Three fingers up/right flick gesture
Page Down	To scroll down/right the content in a scroll container	Two/three finger down/right flick gesture	Three fingers down/right flick gesture
	Starts reading from the beginning of the page		Two fingers up flick gesture
	Starts reading from the current focused item		Two fingers down flick gesture

Note: A gesture may function differently in different assistive technologies. Refer to the respective assistive technology documentation for more information on gestures.

Following are some of the gestures explained in the above table:



For more information on accessibility gestures, refer to:

- Android: https://support.google.com/nexus/answer/2926960?hl=en
- iOS: https://www.apple.com/in/accessibility/osx/voiceover/
- SPA: http://www.w3.org/WAI/mobile/

Platform-specific Limitations

The following is the list of platform-specific limitations.

SPA

This section lists the accessibility limitations of SPA platforms.

1. Scrolling a form and SegmentedUI through a swipe with three fingers and tab gestures (custom scroll) is not supported in SPA platform.
2. When a Popup is loaded, the default focus goes to the form on which the Popup is loaded, and then it comes to Popup.
3. A Label widget without any text is not focusable with tab gestures.
4. When a form is loaded, the default focus can be any where in the form.
5. For the widgets such as ScrollBox, Horizontal Image Strip, Slider, and Segment (PAGEVIEW), a swipe or tab gesture will not bring the focused item into a view area.
6. On the SPA Android platform, accessibiltiyConfig for form and Popup is not supported.
7. On the SPA Android platform, before loading a form, some random text is read by assistive technology. The random text is read from the script tag that may be present in JavaScript.

8. For container widgets, if only a11yHint is configured, then accessibility first reads the text of the child widgets and then a11yHint text that is configured for a container widget.
9. When the accessibilityConfig is set for any container widget (HBox, VBox, Scrollbox, FlexContainer and FlexScrollContainer), the widgets inside the container will not get focused while navigating in iOS Safari.
10. When the accessibilityConfig is set for any container widget (HBox, VBox, Scrollbox, FlexContainer and FlexScrollContainer), the container widget will not get focused while navigating in Android browsers.

Support for iPhone X

iPhone X comes with a super retina edge to edge display with 458 PPI. It comes with a device height of 812, width of 375 and a screen resolution of 2436×1125 (@3X). In iPhone X, Face ID replaces the Touch ID authentication.

> ***Note:*** For information about how to detect whether an iPhone device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

Prerequisites

- You must have Xcode 9.0.1 to test the applications on iPhone X simulator.

Requirements

To provide support for iPhone X on Kony Visualizer, do the following:

- **Provide a 3X splash image**: Current Apps which do not have a 3X splash image, will not be able to fill the complete screen, from edge to edge in iPhone X. Splash screen with name splashscreen-812h@3x.png with a resolution of 1125 By 2436 should be provided for iPhone X.
- **Provide a 2×1 splash video**: Current Apps which do not have a splash video of the resolution 2×1, will not fit-to-screen in iPhone X. Splash video with name splashvideo-2x1.mp4 should be provided for iPhone X. This ensures that the video will fit-to-screen and will not extend in the AVKit safe area. For details on AVKit Safe Area, refer to Human Interface Guidelines for iPhone X.
- **Add NSFaceIDUsageDescription in the infoplist_configuration.json**: You must add the **NSFaceIDUsageDescription** (Privacy - Face ID Usage Description) key in the configuration file for the Face ID to work. Navigation path: `projectname/resources/common/infoplist_configuration.json`. If you do not perform the above step, the app will crash. See Apple's documentation for more information.

Limitations

- When using Face ID on Xcode 9.0.1 simulator, authentication will fail every other time, and an error message Lost connection to coreauthd appears. This is a limitation from Apple. For more information, refer https://openradar.appspot.com/34769844 and https://github.com/lionheart/openradar-mirror/issues/18536

Related API Documentation

- **Home Indicator Auto Hide**: See **prefersHomeIndicatorAutoHidden** in setApplicationBehaviour function.
- Get Biometric Type

Appendix: Universal Links

Kony Studio supports the universal links feature provided by Apple. The universal links feature enables you to set connectivity between your website and mobile app. When an end-user taps a link to your website from any iOS device, the framework directly opens the app (connected with your website) installed on the iOS device. If the app relative to your website is not installed on the iOS device, the URL is opened on the Safari browser. The universal links feature is supported from iOS 9 onwards.

To use the universal links feature, you must configure the following at the app server side, and also in the mobile app.

App server side configuration

1. Create a file with file name, `apple-app-site-association`. The file should contain JSON data about URLs that your app should handle.
2. Upload the file to the root directory or to the `.well-known` subdirectory at your HTTPS web server. Here is a sample apple-app-site-association file.

```
{
"applinks": {
"apps": [],
"details": [
{
"appID": "9JA89QQLNQ.com.apple.wwdc",
"paths": [ "/wwdc/news/", "/videos/wwdc/2015/*"]
},
{
"appID": "ABCD1234.com.apple.wwdc",
"paths": [ "*" ]
}
]
}
```

```
}
```

In the sample, the applinks element defines the apps that are associated with your website. The value of the apps key is an empty array. The `details` key defines an array of dictionaries to link appIDs and URL paths. You can define one dictionary per app that your website supports.

The `appID` is the combination of teamID followed by app's bundle ID (for example, if appID is 9JA89QQLNQ.com.apple.wwdc, where 9JA89QQLNQ is the teamID and com.apple.wwdc is the bundle ID). The `paths` key is an array of strings that contains parts of the website that should be associated with the app, and also the parts that should not be supported by the app. To specify the parts of the website that are not to be supported by the app, prefix the string with "NOT " (including a space after 'T') as shown below.

```
"paths": [ "/wwdc/news/", "NOT /videos/wwdc/2015/*"]
```

> ***Note:*** The **apps** key in the apple-app-site-association file is mandatory and the value should be an empty array.

Mobile app level configuration

1. In Kony Studio, add domains that your app supports in the app **Application Properties** dialog (**Application Properties**>**Native**>**iPhone/iPad**). For more information, see iPhone/iPad tab.

When iOS launches the app, the App Service callback is invoked by the framework and shows the form returned by the callback. If no form is specified in the callback, the home form of the app is shown. The App Service receives an object as an argument, which gives the information about the URL that triggers the universal link.

Here is a sample for handling universal links in App Service.

```
function callback_UniversalLinks(params) {
	if (params.launchmode == 4) {
		kony.print("Universal links CallBack : launchMode : " +
params.launchmode + "Launch Params :  " + JSON.stringify
(params.launchparams));
		return frmCallBack;
	}
}
```

Frequently Asked Questions

Click the required topic to view the corresponding information.

Android SDK and Emulator Setup - FAQs

Below are some potential areas where you may encounter questions while setting up the Android SDK and Emulators.

In case you already have SDK and emulators configured for Android, you may not have to go through the setup process again. This section covers issues to assist with new Android SDK and Emulators setup.

This section is not intended to be used in isolation but must be used along with the Kony Setup documentation.

What do I need for setting up Android with my Kony Visualizer?

To be able to build (APK) and run applications for Android, at a minimum, you need the following setup:

- Android SDK

In the case where you would like to set up and use a GUI version of the AVD Manager (for an Android Emulator) and SDK Manager (to download missing SDK components. Support and Google library components and emulator images), you will also need:

- Android Studio

Why do I need Android Studio?

With version 25.3.0 of the Android SDK, Android had deprecated and removed the standalone AVD (Android Virtual Device) Manager and SDK Manager GUI tools from the Android SDK package. These tools are now only available as part of the Android Studio with GUI or through the command line. To use GUI to create Android Emulators and manage versions of SDK, Android expects its users to install Android Studio.

What is an AVD Manager?

The AVD (Android Virtual Device) Manager is an Android SDK tool that helps you create and manage AVDs which allow Android developers to emulate an Android Device.

I have downloaded the Android Studio, but I see errors in its console. For example, Failed to find the target with hash string android-26 in: <location>. Should I try to resolve these?

You do not need to resolve errors you see in Android Studio console. The Android Studio is only used for having a GUI interface to AVD Manager and SDK Manager. You may need to install the corresponding android-target which your Kony Visualizer project depends for compilation which is captured as `compileSdkVersion` in `build.gradle`.

What is HAXM? Do I need it?

The Android Emulator supports several hardware acceleration features to improve performance and HAXM is one of them. HAXM or Hardware Accelerated Execution Manager is a Virtual Machine acceleration software to speed up Android Emulators.

HAXM can be used to speed up X86 based Emulators. AVDs that use an ARM or MIPS-based system images cannot be accelerated using HAXM.

You can always use ARM-based emulators however if your environment supports HAXM, it is recommended to set it up to speed up your emulators.

How do I Setup HAXM?

You can read more about HAXM and steps on how to setup HAXM at https://developer.android.com/studio/run/emulator-acceleration.html#accel-graphics.

Will HAXM based emulator work on my machine?

HAXM based emulators do not work in all environments. You may need to make changes to your environment configuration and BIOS settings while setting up HAXM. HAXM does not work on a Parallels VM.

For detailed steps on how to setup HAXM, click https://developer.android.com/studio/run/emulator-acceleration.html#accel-graphics

How can I use an X86 Emulator with Visualizer?

For creating x86 based emulator you need to

- Install HAXM
- Enable Virtualization extensions in your computer BIOS

https://developer.android.com/studio/run/emulator-acceleration.html#accel-vm

To use an x86 Emulator with Visualizer, you need to build your Visualizer with X86 device support.

You can do this by following the below steps.

1. Navigate to **Project Settings**.
2. Select the **Native** tab and then Select the **Android** tab.
3. Check the box for **Support x86 device** and then click **Finish**.

Now you can build your projects and test them using an x86 Emulator.

Support 64-bit Devices option usage

If a user has checked the Support 64-bit Devices option and the built binary has 64 bit libraries, such apks would only install and run on emulators created on 64 bit architectures, i.e arm64 and x86_64. There are some cases where the apk may contain only 32 bit libs, exception cases are captured in documentation.

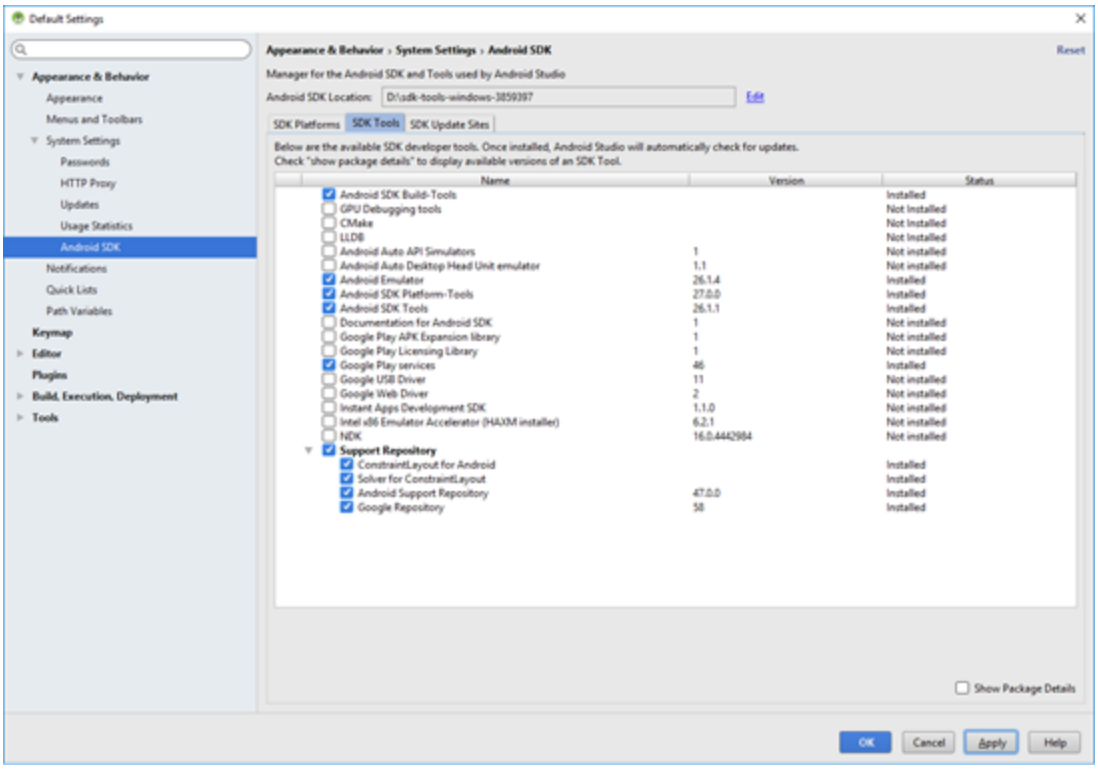
What if I have downloaded the Android SDK package (zipped) separate from Android Studio? How do I use that to create an AVD?

If a developer has downloaded just SDK tools package and wants to set up the full SDK out of it from Android studio from https://developer.android.com/studio/index.html, follow the below steps:

1. In the downloaded and unzipped SDK folder, create folder platforms in parallel to the tools folder.
2. Open Android Studio and navigate to **File** > **Project Structure**> **SDK Location**> **Android SDK Location**.
3. Set the **Android SDK Location** path.
4. Navigate to **File** > **Settings** > **Appearance & Behavior** > **System Settings** > **Android SDK**. The Android SDK settings window appears with the SDK Platforms tab open by default.
5. Click the **SDK Tools** tab.
6. Select the following components.
    i. Android SDK Build Tools
    ii. Android Emulator
    iii. Android SDK Platform Tools
    iv. Android SDK Tools

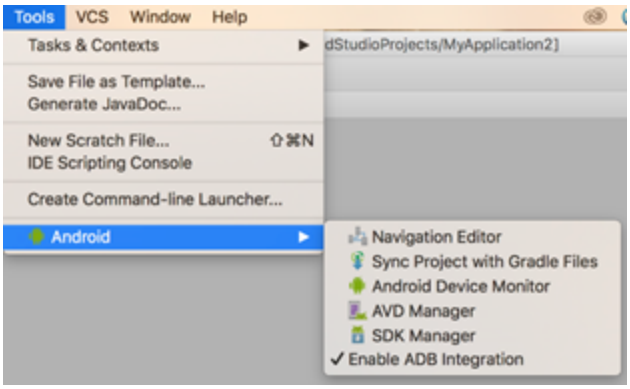
v. Google Play Services

vi. Support Repository (all the items under it)

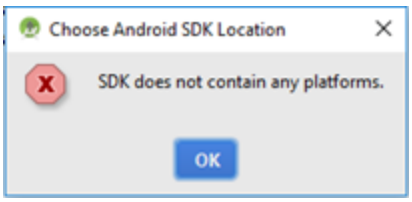


7. Download at least one system Image from **SDK Platforms** tab and click **Apply** to download.

   Once the download is complete, under the Tools Menu in Android Studio, you should be able to see an Android option, which would allow you to create an AVD using AVD Manager.



Visualizer shows an error message (could not find platform folder) while trying to add the Android SDK location. What do I do?



To fix the platform folder issue, do the following:

1. In the downloaded and unzipped SDK folder, create a folder **platforms** parallel to the **tools** folder. Once you create the folder, you will be able to select the Android SDK location.
2. Build your project for Android, and the build process will determine the required tools which are missing and will automatically download the required tools to update the Android SDK.
3. Once the required tools are downloaded, you can set up your emulator and run your application.

I have an Android SDK, but I do not have Android Studio installed. Will I still be able to Build & Run my Apps?

You will be able to build your apps using the Android SDK, but since you do not have an Android Studio, you will not have GUI to create AVDs to run your app on an emulator. However you can explore command line options to create AVD's using the avdmanager tool.

For more information, refer: https://developer.android.com/studio/command-line/avdmanager.html

My Emulator (AVD) does not launch. How do I validate that my emulator is working?

To launch the AVD you created, do the following:

1. In your computer, add the following entries to the user path variable as part of the environment variables in the order of preference specified below. Prefix the entries to path.
    i. AndroidSDKPath
    ii. AndroidSDKPath\emulator\
    iii. AndroidSDKPath\tools\
    iv. AndroidSDKPath\platform-tools
2. Set the ANDROID_HOME variable to **AndroidSDKPath**.

3. Restart the machine and try the command if the emulator configured is ARM based.
   `emulator.exe -avd <avdname> -port 5554`

4. Cross check if the right emulator is being picked from right SDK folder by using the below command in command prompt.
   Windows:
   `where emulator`
   MAC:
   `which emulator`
   If the **AndroidSDKPath/emulator** folder and the **AndroidSDKPath/tools** folder contains the emulator.exe in the newly downloaded SDK, make sure the one under **emulator** is picked up.

5. If the right path is not being picked up or if the emulator command fails to launch the AVD, clean up unnecessary **PATH** environment variable entries and check if it resolves the issue.

> ***Note:*** If the emulator does not launch after performing all the above steps, uninstall Kony Visualizer and install Kony Visualizer again.

> ***Important:*** It is recommended to configure Android SDK path entries before installing Kony Visualizer.

When I ran my app on the emulator, the emulator launched, but my console continues to display waiting for device or stops at installing package. What should I do?

Sometimes the ARM-based emulators run extremely slow in some environments and may take upwards of 10 - 15 minutes for the emulator to launch the first time. You may want to wait and observe if this is the case on your environment. X86 Emulators usually launch much faster.

Do I need to run the app every time I need to launch the emulator?

No, you can use the **Launch Emulator** option under the build menu to directly launch the project in the emulator.

Jasmine Test Automation - FAQs

Can we automate an app which is built in the Release mode?

No, an app which is built in the **Release** mode cannot be automated. The preliminary condition to record or playback an application is that it must be built in the **Test** mode.

Does connecting the device mean connecting via USB cable?

The connection happens over the network. So, the device need not be connected via USB to the system on

which the Kony Automator is launched.

Why does the “Device connected successfully” message not appear even after launching the app?

- Ensure the **build mode** is set as **Test** before you build and publish your application. If you are previewing the application using the live preview feature, ensure the **preview mode** is set as **Test**.
- Check if the system on which Kony Automator is running and the device are on the same network.
- Although the connected network is same for the system and the device, the app might be using inaccessible IP (Default gateway). To modify the IP, follow these steps:

    1. From the main menu, navigate to **Edit** > **Preferences**.
       The **Visualizer Preferences** window appears.
    2. Uncheck the **Auto-detect System IP Address** check box.
    3. For **Static IP**, enter the IP address to be used.

- The port on which the Kony Automator is running might be unavailable. To modify the port number, follow these steps:

    1. From the main menu, navigate to **Edit** > **Preferences**.
        - For windows machine, navigate to **Edit** > **Preferences**.
        - For mac machine, navigate to **Kony Visualizer** > **Preferences**.

       The **Visualizer Preferences** window appears.
    2. From the left pane, select the **Test Recorder** section.
    3. In the **General** section, modify the port number.
    4. Click **Validate** to check the availability of the port.
       If the port is available, an alert message **Port xxxx is available** appears.
    5. Click **Done**.
    6. Restart Visualizer.

- If you are using Android Pie device, refer here.

Why am I not able to start Visualizer after making the addon changes?

The addOn server runs on port 9111 by default. Make sure the port is not occupied.

If you want to use a different port, follow these steps:

1. From the main menu, navigate to **Edit** > **Preferences**.
    - For windows machine, navigate to **Edit** > **Preferences**.
    - For mac machine, navigate to **Kony Visualizer** > **Preferences**.

    The **Visualizer Preferences** window appears.
2. From the left pane, select the **Test Recorder** section.
3. In the **General** section, modify the port number.
4. Click **Validate** to check the availability of the port.
   If the port is available, an alert message **Port xxxx is available** appears.
5. Click **Done**.
6. Restart Visualizer.

Is await necessary for some APIs like wait, waitFor?

Await is necessary for the APIs that are asynchronous to ensure that the APIs function synchronously.

Why am I unable to see any recordable actions for a widget?

The recording capability may not be available for a specified widget. However, you can add a code snippet manually by using the automation API. If an API is not available for a required action, please raise an ideation ticket on Base Camp.

Why am I getting an exception and the app keeps crashing on playback?

Make sure the APIs like wait/waitFor are properly used so that the test script waits before performing actions on the widgets.

Why isn't automation running on Android Pie and the above device?

Since Android has blocked http calls in Pie, you must make certain changes as mentioned here, for the automation to work.

How can I capture an event in a recording?

Please raise an ideation ticket on basecamp.